A Distributional and Orthographic Aggregation Model for English Derivational Morphology

Daniel Deutsch*, **John Hewitt*** **and Dan Roth**
Department of Computer and Information Science
University of Pennsylvania
{ddeutsch,johnhew,danroth}@seas.upenn.edu

Abstract

Modeling derivational morphology to generate words with particular semantics is useful in many text generation tasks, such as machine translation or abstractive question answering. In this work, we tackle the task of derived word generation. That is, given the word "run," we attempt to generate the word "runner" for "someone who runs." We identify two key problems in generating derived words from root words and transformations: suffix ambiguity and orthographic irregularity. We contribute a novel aggregation model of derived word generation that learns derivational transformations both as orthographic functions using sequence-to-sequence models and as functions in distributional word embedding space. Our best open-vocabulary model, which can generate novel words, and our best closed-vocabulary model, show 22% and 37% relative error reductions over current state-of-the-art systems on the same dataset.

1 Introduction

The explicit modeling of morphology has been shown to improve a number of tasks (Seeker and Çetinoglu, 2015; Luong et al., 2013). In a large number of the world's languages, many words are composed through morphological operations on subword units. Some languages are rich in *inflectional* morphology, characterized by syntactic transformations like pluralization. Similarly, languages like English are rich in *derivational* morphology, where the semantics of words are composed from smaller parts. The AGENT derivational transformation, for example, answers the question, *what is the word for 'someone who runs'?* with the answer, a *runner*.[1] Here, AGENT is spelled out as suffixing *-ner* onto the root verb *run*.

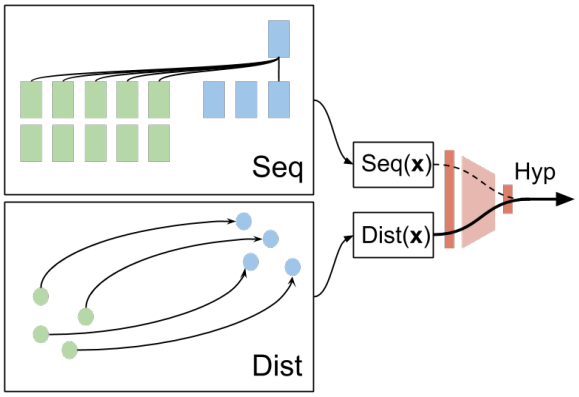

Figure 1: Diagram depicting the flow of our aggregation model. Two models generate a hypothesis according to orthogonal information; then one is chosen as the final model generation. Here, the hypothesis from the distributional model is chosen.

We tackle the task of derived word generation. In this task, a root word $\mathbf{x}$ and a derivational transformation t are given to the learner. The learner's job is to produce the result of the transformation on the root word, called the derived word $\mathbf{y}$. Table 1 gives examples of these transformations.

Previous approaches to derived word generation model the task as a character-level sequence-to-sequence (seq2seq) problem (Cotterell et al., 2017b). The letters from the root word and some encoding of the transformation are given as input to a neural encoder, and the decoder is trained to produce the derived word, one letter at a time. We identify the following problems with these approaches:

First, because these models are unconstrained, they can generate sequences of characters that do

*These authors contributed equally; listed alphabetically.

[1]We use the verb *run* as a demonstrative example; the transformation can be applied to most verbs.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1938–1947
Melbourne, Australia, July 15 - 20, 2018.

$\mathbf{x}$	t		$\mathbf{y}$
wise	ADVERB	$\rightarrow$	wise*ly*
simulate	RESULT	$\rightarrow$	simula*tion*
approve	RESULT	$\rightarrow$	approv*al*
overstate	RESULT	$\rightarrow$	overstate*ment*
yodel	AGENT	$\rightarrow$	yodel*er*
survive	AGENT	$\rightarrow$	surviv*or*
intense	NOMINAL	$\rightarrow$	intens*ity*
effective	NOMINAL	$\rightarrow$	effective*ness*
pessimistic	NOMINAL	$\rightarrow$	pessim*ism*

Table 1: The goal of derived word generation is to produce the derived word, $\mathbf{y}$, given both the root word, $\mathbf{x}$, and the transformation t, as demonstrated here with examples from the dataset.

not form actual words. We argue that requiring the model to generate a known word is a reasonable constraint in the special case of English derivational morphology, and doing so avoids a large number of common errors.

Second, sequence-based models can only generalize string manipulations (such as "add *-ment*") if they appear frequently in the training data. Because of this, they are unable to generate derived words that do not follow typical patterns, such as generating *truth* as the nominative derivation of *true*. We propose to learn a function for each transformation in a low dimensional vector space that corresponds to mapping from representations of the root word to the derived word. This eliminates the reliance on orthographic information, unlike related approaches to distributional semantics, which operate at the suffix level (Gupta et al., 2017).

We contribute an aggregation model of derived word generation that produces hypotheses independently from two separate learned models: one from a seq2seq model with only orthographic information, and one from a feed-forward network using only distributional semantic information in the form of pretrained word vectors. The model learns to choose between the hypotheses according to the relative confidence of each. This system can be interpreted as learning to decide between positing an orthographically regular form or a semantically salient word. See Figure 1 for a diagram of our model.

We show that this model helps with two open problems with current state-of-the-art seq2seq derived word generation systems, suffix ambiguity and orthographic irregularity (Section 2). We also improve the accuracy of seq2seq-only derived word systems by adding external information through constrained decoding and hypothesis rescoring. These methods provide orthogonal gains to our main contribution.

We evaluate models in two categories: open vocabulary models that can generate novel words unattested in a preset vocabulary, and closed-vocabulary models, which cannot. Our best open-vocabulary and closed-vocabulary models demonstrate 22% and 37% relative error reductions over the current state of the art.

2 Background: Derivational Morphology

Derivational transformations generate novel words that are semantically composed from the root word and the transformation. We identify two unsolved problems in derived word transformation, each of which we address in Sections 3 and 4.

First, many plausible choices of suffix for a single pair of root word and transformation. For example, for the verb *ground*, the RESULT transformation could plausibly take as many forms as[2]

$$(\text{ground}, \text{RESULT}) \rightarrow \text{ground}\textit{ing}$$
$$(\text{ground}, \text{RESULT}) \rightarrow \text{*ground}\textit{ation}$$
$$(\text{ground}, \text{RESULT}) \rightarrow \text{*ground}\textit{ment}$$
$$(\text{ground}, \text{RESULT}) \rightarrow \text{*ground}\textit{al}$$

However, only one is correct, even though each suffix appears often in the RESULT transformation of other words. We will refer to this problem as "suffix ambiguity."

Second, many derived words seem to lack a generalizable orthographic relationship to their root words. For example, the RESULT of the verb *speak* is *speech*. It is unlikely, given an orthographically similar verb *creak*, that the RESULT be *creech* instead of, say, *creaking*. Seq2seq models must grapple with the problem of derived words that are the result of unlikely or potentially unseen string transformations. We refer to this problem as "orthographic irregularity."

3 Sequence Models and Corpus Knowledge

In this section, we introduce the prior state-of-the-art model, which serves as our baseline system. Then we build on top of this system by incorporating a dictionary constraint and rescoring the

[2]The * indicates a non-word.

model's hypotheses with token frequency information to address the suffix ambiguity problem.

3.1 Baseline Architecture

We begin by formalizing the problem and defining some notation. For source word $\mathbf{x} = x_1, x_2, \ldots x_m$, a derivational transformation t, and target word $\mathbf{y} = y_1, y_2, \ldots y_n$, our goal is to learn some function from the pair $(\mathbf{x}, t)$ to $\mathbf{y}$. Here, x_i and y_j are the ith and jth characters of the input strings $\mathbf{x}$ and $\mathbf{y}$. We will sometimes use $\mathbf{x}_{1:i}$ to denote $x_1, x_2, \ldots x_i$, and similarly for $\mathbf{y}_{1:j}$.

The current state-of-the-art model for derived-form generation approaches this problem by learning a character-level encoder-decoder neural network with an attention mechanism (Cotterell et al., 2017b; Bahdanau et al., 2014).

The input to the bidirectional LSTM encoder (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) is the sequence #, $x_1, x_2, \ldots x_m$, #, t, where # is a special symbol to denote the start and end of a word, and the encoding of the derivational transformation t is concatenated to the input characters. The model is trained to minimize the cross entropy of the training data. We refer to our reimplementation of this model as SEQ.

For a more detailed treatment of neural sequence-to-sequence models with attention, we direct the reader to Luong et al. (2015).

3.2 Dictionary Constraint

The suffix ambiguity problem poses challenges for models which rely exclusively on input characters for information. As previously demonstrated, words derived via the same transformation may take different suffixes, and it is hard to select among them based on character information alone. Here, we describe a process for restricting our inference procedure to only generate known English words, which we call a dictionary constraint. We believe that for English morphology, a large enough corpus will contain the vast majority of derived forms, so while this approach is somewhat restricting, it removes a significant amount of ambiguity from the problem.

To describe how we implemented this dictionary constraint, it is useful first to discuss how decoding in a seq2seq model is equivalent to solving a shortest path problem. The notation is specific to our model, but the argument is applicable to seq2seq models in general.

The goal of decoding is to find the most probable structure $\hat{\mathbf{y}}$ conditioned on some observation $\mathbf{x}$ and transformation t. That is, the problem is to solve

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} \mid \mathbf{x}, t) \tag{1}$$

$$= \arg\min_{\mathbf{y} \in \mathcal{Y}} -\log p(\mathbf{y} \mid \mathbf{x}, t) \tag{2}$$

where $\mathcal{Y}$ is the set of valid structures. Sequential models have a natural ordering $\mathbf{y} = y_1, y_2, \ldots y_n$ over which $-\log p(\mathbf{y} \mid \mathbf{x}, t)$ can be decomposed

$$-\log p(\mathbf{y} \mid \mathbf{x}, t) = \sum_{t=1}^{n} -\log p(y_t \mid \mathbf{y}_{1:t-1}, \mathbf{x}, t) \tag{3}$$

Solving Equation 2 can be viewed as solving a shortest path problem from a special starting state to a special ending state via some path which uniquely represents $\mathbf{y}$. Each vertex in the graph represents some sequence $\mathbf{y}_{1:i}$, and the weight of the edge from $\mathbf{y}_{1:i}$ to $\mathbf{y}_{1:i+1}$ is given by

$$-\log p(y_{i+1} \mid \mathbf{y}_{1:i-1}, \mathbf{x}, t) \tag{4}$$

The weight of the path from the start state to the end state via the unique path that describes $\mathbf{y}$ is exactly equal to Equation 3. When the vocabulary size is too large, the exact shortest path is intractable, and approximate search methods, such as beam search, are used instead.

In derived word generation, $\mathcal{Y}$ is an infinite set of strings. Since $\mathcal{Y}$ is unrestricted, almost all of the strings in $\mathcal{Y}$ are not valid words. Given a dictionary $\mathcal{Y}_\mathcal{D}$, the search space is restricted to only those words in the dictionary by searching over the trie induced from $\mathcal{Y}_\mathcal{D}$, which is a subgraph of the unrestricted graph. By limiting the search space to $\mathcal{Y}_\mathcal{D}$, the decoder is guaranteed to generate some known word. Models which use this dictionary-constrained inference procedure will be labeled with +DICT. Algorithm 1 has the pseudocode for our decoding procedure.

We discuss specific details of the search procedure and interesting observations of the search space in Section 6. Section 5.2 describes how we obtained the dictionary of valid words.

3.3 Word Frequency Knowledge through Rescoring

We also consider the inclusion of explicit word frequency information to help solve suffix ambiguity, using the intuition that "real" derived words

are likely to be frequently attested. This permits a high-recall, potentially noisy dictionary.

We are motivated by very high top-10 accuracy compared to top-1 accuracy, even among dictionary-constrained models. By rescoring the hypotheses of a model using word frequency (a word-global signal) as a feature, attempt to recover a portion of this top-10 accuracy.

When a model has been trained, we query it for its top-10 most likely hypotheses. The union of all hypotheses for a subset of the training observations forms the training set for a classifier that learns to predict whether a hypothesis generated by the model is correct. Each hypothesis is labelled with its correctness, a value in $\{\pm 1\}$. We train a simple combination of two scores: the seq2seq model score for the hypothesis, and the log of the word frequency of the hypothesis.

To permit a nonlinear combination of word frequency and model score, we train a small multilayer perceptron with the model score and the frequency of a derived word hypothesis as features.

At testing time, the 10 hypotheses generated by a single seq2seq model for a single observation are rescored. The new model top-1 hypothesis, then, is the argmax over the 10 hypotheses according to the rescorer. In this way, we are able to incorporate word-global information, e.g. word frequency, that is ill-suited for incorporation at each character prediction step of the seq2seq model. We label models that are rescored in this way +FREQ.

4 Distributional Models

So far, we have presented models that learn derivational transformations as orthographic operations. Such models struggle by construction with the orthographic irregularity problem, as they are trained to generalize orthographic information. However, the semantic relationships between root words and derived words are the same even when the orthography is dissimilar. It is salient, for example, that irregular word *speech* is related to its root *speak* in about the same way as how *exploration* is related to the word *explore*.

We model distributional transformations as functions in dense distributional word embedding spaces, crucially learning a function per derivational transformation, not per suffix pair. In this way, we aim to explicitly model the semantic transformation, not the othographic information.

4.1 Feed-forward derivational transformations

For all source words $\mathbf{x}$ and all target words $\mathbf{y}$, we look up static distributional embeddings $v_{\mathbf{x}}, v_{\mathbf{y}} \in \mathbb{R}^d$. For each derivational transformation t, we learn a function $f_t : \mathbb{R}^d \to \mathbb{R}^d$ that maps $v_{\mathbf{x}}$ to $v_{\mathbf{y}}$. f_t is parametrized as two-layer perceptron, trained using a squared loss,

$$\mathcal{L} = \mathbf{b}^{\mathrm{T}}\mathbf{b} \tag{5}$$

$$\mathbf{b} = f_t(v_{\mathbf{x}}) - v_{\mathbf{y}} \tag{6}$$

We perform inference by nearest neighbor search in the embedding space. This inference strategy requires a subset of strings for our embedding dictionary, $\mathcal{Y}_V$.

Upon receiving $(\mathbf{x}, t)$ at test time, we compute $f_t(v_{\mathbf{x}})$ and find the most similar embeddings in $\mathcal{Y}_V$. Specifically, we find the top-k most similar embeddings, and take the most similar derived word that starts with the same 4 letters as the root word, and is not identical to it. This heuristic filters out highly implausible hypotheses.

We use the single-word subset of the Google News vectors (Mikolov et al., 2013) as $\mathcal{Y}_V$, so the size of the vocabulary is 929k words.

4.2 SEQ and DIST Aggregation

The seq2seq and distributional models we have presented learn with disjoint information to solve separate problems. We leverage this intuition to build a model that chooses, for each observation, whether to generate according to orthographic information via the SEQ model, or produce a potentially irregular form via the DIST model.

To train this model, we use a held-out portion of the training set, and filter it to only observations for which exactly one of the two models produces the correct derived form. Finally, we make the strong assumption that the probability of a derived form being generated correctly according to 1 model as opposed to the other is dependent only on the unnormalized model score from each. We model this as a logistic regression (t is omitted for clarity):

$$P(\cdot|\mathbf{y_D}, \mathbf{y_S}, \mathbf{x}) = \\ \text{softmax}(\mathbf{W}_e\ [\textsc{Dist}(\mathbf{y_D}|\mathbf{x}); \textsc{Seq}(\mathbf{y_S}|\mathbf{x})] + \mathbf{b}_e)$$

where $\mathbf{W_e}$ and $\mathbf{b_e}$ are learned parameters, $\mathbf{y_D}$ and $\mathbf{y_S}$ are the hypotheses of the distributional and seq2seq models, and $\textsc{Dist}(\cdot)$ and $\textsc{Seq}(\cdot)$ are the models' likelihood functions. We denote this aggregate AGGR in our results.

5 Datasets

In this section we describe the derivational morphology dataset used in our experiments and how we collected the dictionary and token frequencies used in the dictionary constraint and rescorer.

5.1 Derivational Morphology

In our experiments, we use the derived word generation derivational morphology dataset released in Cotterell et al. (2017b). The dataset, derived from NomBank (Meyers et al., 2004) , consists of 4,222 training, 905 validation, and 905 test triples of the form $(\mathbf{x}, t, \mathbf{y})$. The transformations are from the following categories: ADVERB (ADJ $\rightarrow$ ADV), RESULT (V $\rightarrow$ N), AGENT (V $\rightarrow$ N), and NOMINAL (ADJ $\rightarrow$ N). Examples from the dataset can be found in Table 1.

5.2 Dictionary and Token Frequency Statistics

The dictionary and token frequency statistics used in the dictionary constraint and frequency reranking come from the Google Books NGram corpus (Michel et al., 2011). The unigram frequency counts were aggregated across years, and any tokens which appear fewer than approximately 2,000 times, do not end in a known possible suffix, or contain a character outside of our vocabulary were removed.

The frequency threshold was determined using development data, optimizing for high recall. We collect a set of known suffixes from the training data by removing the longest common prefix between the source and target words from the target word. The result is a dictionary with frequency information for around 360k words, which covers 98% of the target words in the training data.[3]

6 Inference Procedure Discussion

In many sequence models where the vocabulary size is large, exact inference by finding the true shortest path in the graph discussed in Section 3.2 is intractable. As a result, approximate inference techniques such as beam search are often used, or the size of the search space is reduced, for example, by using a Markov assumption. We, however, observed that exact inference via a shortest path algorithm is not only tractable in our model, but only slightly more expensive than greedy search and significantly less expensive than beam search.

[3] The remaining 2% is mostly words with hyphens or mistakes in the dataset.

Method	Accuracy	Avg. #States
GREEDY	75.9	11.8
BEAM	76.2	101.2
SHORTEST	76.2	11.8
DICT+GREEDY	77.2	11.7
DICT+BEAM	82.6	91.2
DICT+SHORTEST	82.6	12.4

Table 2: The average accuracies and number of states explored in the search graph of 30 runs of the SEQ model with various search procedures. The BEAM models use a beam size of 10.

To quantify this claim, we measured the accuracy and number of states explored by greedy search, beam search, and shortest path with and without a dictionary constraint on the development data. Table 2 shows the results averaged over 30 runs. As expected, beam search and shortest path have higher accuracies than greedy search and explore more of the search space. Surprisingly, beam search and shortest path have nearly identical accuracies, but shortest path explores significantly fewer hypotheses.

At least two factors contribute to the tractability of exact search in our model. First, our character-level sequence model has a vocabulary size of 63, which is significantly smaller than token-level models, in which a vocabulary of 50k words is not uncommon. The search space of sequence models is dependent upon the size of the vocabulary, so the model's search space is dramatically smaller than for a token-level model.

Second, the inherent structure of the task makes it easy to eliminate large subgraphs of the search space. The first several characters of the input word and output word are almost always the same, so the model assigns very low probability to any sequence with different starting characters than the input. Then, the rest of the search procedure is dedicated to deciding between suffixes. Any suffix which does not appear frequently in the training data receives a low score, leaving the search to decide between a handful of possible options. The result is that the learned probability distribution is very spiked; it puts very high probability on just a few output sequences. It is empirically true that the top few most probable sequences have significantly higher scores than the next most probable sequences, which supports this hypothesis.

In our subsequent experiments, we decode using

Algorithm 1 The decoding procedure uses a shortest-path algorithm to find the most probable output sequence. The dictionary constraint is (optionally) implemented on line 9 by only considering prefixes that are contained in some trie T.

```
1:  procedure DECODE(x, t, V, T)
2:      H ← Heap()
3:      H.insert(0, #)
4:      while H is not empty do
5:          y ← H.remove()
6:          if y is a complete word then return y
7:          for y ∈ V do
8:              y' ← y + y
9:              if y' ∈ T then
10:                 s ← FORWARD(x, t, y')
11:                 H.insert(s, y')
```

exact inference by running a shortest path algorithm (see Algorithm 1). For reranking models, instead of typically using a beam of size k, we use the top k most probable sequences.

7 Results

In all of our experiments, we use the training, development, and testing splits provided by Cotterell et al. (2017b) and average over 30 random restarts. Table 3 displays the accuracies and average edit distances on the test set of each of the systems presented in this work and the state-of-the-art model from Cotterell et al. (2017b).

First, we observed that SEQ outperforms the results reported in Cotterell et al. (2017b) by a large margin, despite the fact that the model architectures are the same. We attribute this difference to better hyperparameter settings and improved learning rate annealing.

Then, it is clear that the accuracy of the distributional model, DIST, is significantly lower than any seq2seq model. We believe the orthography-informed models perform better because most observations in the dataset are orthographically regular, providing low-hanging fruit.

Open-vocabulary models Our open-vocabulary aggregation model AGGR improves performance by 3.8 points accuracy over SEQ, indicating that the sequence models and the distributional model are contributing complementary signals. AGGR is an open-vocabulary model like Cotterell et al. (2017b) and improves upon it by 6.3 points, making it our best comparable model. We provide an in-depth analysis of the strengths of SEQ and DIST in Section 7.1.

Model	Accuracy	Edit
Cotterell et al. (2017b)	71.7	0.97
DIST	54.9	3.23
SEQ	74.2	0.88
AGGR	78.0	0.83
SEQ+FREQ	79.3	0.71
DUAL+FREQ	**82.0**	**0.64**
SEQ+DICT	80.4	0.72
AGGR+DICT	81.0	0.78
SEQ+FREQ+DICT	81.2	0.71
AGGR+FREQ+DICT	**82.4**	**0.67**

Table 3: The accuracies and edit distances of the models presented in this paper and prior work. For edit distance, lower is better. The dictionary-constrained models are on the lower half of the table.

Closed-vocabulary models We now consider closed-vocabulary models that improve upon the seq2seq model in AGGR. First, we see that restricting the decoder to only generate known words is extremely useful, with SEQ+DICT improving over SEQ by 6.2 points. Qualitatively, we note that this constraint helps solve the suffix ambiguity problem, since orthographically plausible incorrect hypotheses are pruned as non-words. See Table 6 for examples of this phenomenon. Additionally, we observe that the dictionary-constrained model outperforms the unconstrained model according to top-10 accuracy (see Table 5).

Rescoring (+FREQ) provides further improvement of 0.8 points, showing that the decoding dictionary constraint provides a higher-quality beam that still has room for top-1 improvement. All together, AGGR+FREQ+DICT provides a 4.4 point improvement over the best open-vocabulary model, AGGR. This shows the disambiguating power of assuming a closed vocabulary.

Edit Distance One interesting side effect of the dictionary constraint appears when comparing AGGR+FREQ with and without the dictionary constraint. Although the accuracy of the dictionary-constrained model is better, the average edit distance is worse. The unconstrained model is free to put invalid words which are orthographically similar to the target word in its top-k, however the constrained model can only choose valid words. This means it is easier for the unconstrained model to generate words which have a low edit distance to the ground truth, whereas the constrained model

	Cotterell et al. (2017b)		AGGR		AGGR+FREQ +DICT	
	acc	edit	acc	edit	acc	edit
NOMINAL	35.1	2.67	**68.0**	**1.32**	62.1	1.40
RESULT	52.9	1.86	59.1	1.83	**69.7**	**1.29**
AGENT	65.6	0.78	73.5	0.65	**79.1**	**0.57**
ADVERB	93.3	**0.18**	94.0	**0.18**	**95.0**	0.22

Table 4: The accuracies and edit distances of our best open-vocabulary and closed-vocabulary models, AGGR and AGGR+FREQ+DICT compared to prior work, evaluated at the transformation level. For edit distance, lower is better.

	Cotterell et al. (2017b)	SEQ	SEQ+DICT
	top-10-acc	top-10-acc	top-10-acc
NOMINAL	70.2	73.7	**87.5**
RESULT	72.6	79.9	**90.4**
AGENT	82.2	88.4	**91.6**
ADVERB	96.5	**96.9**	**96.9**

Table 5: The accuracies of the top-10 best outputs for the SEQ, SEQ+DICT, and prior work demonstrate that the dictionary constraint significantly improves the overall candidate quality.

can only do that if such a word exists. The result is a more accurate, yet more orthographically diverse, set of hypotheses.

Results by Transformation Next, we compare our best open vocabulary and closed vocabulary models to previous work across each derivational transformation. These results are in Table 4.

The largest improvement over the baseline system is for NOMINAL transformations, in which the AGGR has a 49% reduction in error. We attribute most of this gain to the difficulty of this particular transformation. NOMINAL is challenging because there are several plausible endings (e.g. *-ity*, *-ness*, *-ence*) which occur at roughly the same rate. Additionally, NOMINAL examples are the least frequent transformation in the dataset, so it is challenging for a sequential model to learn to generalize. The distributional model, which does not rely on suffix information, does not have this same weakness, so the aggregation AGGR model has better results.

The performance of AGGR+FREQ+DICT is worse than AGGR, however. This is surprising because, in all other transformations, adding dictionary information improves the accuracies. We believe this is due to the ambiguity of the ground truth: Many root words have seemingly multiple plausible nominal transformations, such as *rigid* $\rightarrow$ {*rigidness*, *rigidity*} and *equivalent* $\rightarrow$ {*equivalence*, *equivalency*}. The dictionary constraint produces a better set of hypotheses to rescore, as demonstrated in Table 5. Therefore, the dictionary-constrained model is likely to have more of these ambiguous cases, which makes the task more difficult.

7.1 Strengths of SEQ and DIST

In this subsection we explore why AGGR improves consistently over SEQ even though it maintains an open vocabulary. We have argued that DIST is able to correctly produce derived words that are orthographically irregular or infrequent in the training data. Figure 2 quantifies this phenomenon, analyzing the difference in accuracy between the two models, and plotting this in relationship to the frequency of the suffix in the training data. The plot shows that SEQ excels at generating derived words ending in *-ly, -ion*, and other suffixes that appeared frequently in the training data. DIST's improvements over SEQ are generally much less frequent in the training data, or as in the case of *-ment*, are less frequent than other suffixes for the same transformation (like *-ion*.) By producing derived words whose suffixes show up rarely in the training data, DIST helps solve the orthographic irregularity problem.

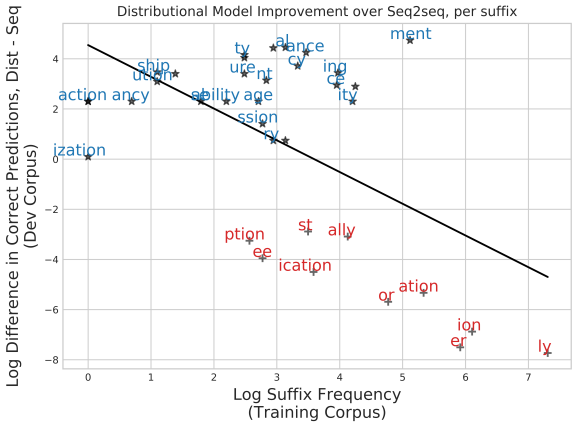

Figure 2: Aggregating across 30 random restarts, we tallied when SEQ and DIST correctly produced derived forms of each suffix. The y-axis shows the logarithm of the difference, per suffix, between the tally for DIST and the tally for SEQ. On the x-axis is the logarithm of the frequency of derived words with each suffix in the training data. A linear regression line is plotted to show the relationship between log suffix frequency and log difference in correct predictions. Suffixes that differ only by the first letter, as with *-ger* and *-er*, have been merged and represented by the more frequent of the two.

8 Prior Work

There has been much work on the related task of inflected word generation (Durrett and DeNero,

x	t	DIST	SEQ	AGGR	AGGR+DICT
approve	RESULT	**approval**	approvation	**approval**	**approval**
bankrupt	NOMINAL	**bankruptcy**	bankruption	**bankruptcy**	**bankruptcy**
irretrievable	ADVERB	irreparably	**irretrievably**	**irretrievably**	**irretrievably**
connect	RESULT	connectivity	**connection**	**connection**	**connection**
stroll	AGENT	strolls	**stroller**	**stroller**	**stroller**
emigrate	SUBJECT	emigre	emigrator	emigrator	**emigrant**
ubiquitous	NOMINAL	**ubiquity**	ubiquit	ubiquit	**ubiquity**
hinder	AGENT	hinderer	hinderer	hinderer	hinderer
vacant	NOMINAL	vacance	vacance	vacance	vacance

Table 6: Sample output from a selection of models. The words in bold mark the correct derivations. "Hindrance" and "vacancy" are the expected derived words for the last two rows.

2013; Rastogi et al., 2016; Hulden et al., 2014). It is a structurally similar task to ours, but does not have the same difficulty of challenges (Cotterell et al., 2017a,b), which we have addressed in our work. The paradigm completion for derivational morphology dataset we use in this work was introduced in Cotterell et al. (2017b). They apply the model that won the 2016 SIGMORPHON shared task on inflectional morphology to derivational morphology (Kann and Schütze, 2016; Cotterell et al., 2016). We use this as our baseline.

Our implementation of the dictionary constraint is an example of a special constraint which can be directly incorporated into the inference algorithm at little additional cost. Roth and Yih (2004, 2007) propose a general inference procedure that naturally incorporates constraints through recasting inference as solving an integer linear program.

Beam or hypothesis rescoring to incorporate an expensive or non-decomposable signal into search has a history in machine translation (Huang and Chiang, 2007). In inflectional morphology, Nicolai et al. (2015) use this idea to rerank hypotheses using orthographic features and Faruqui et al. (2016) use a character-level language model. Our approach is similar to Faruqui et al. (2016) in that we use statistics from a raw corpus, but at the token level.

There have been several attempts to use distributional information in morphological generation and analysis. Soricut and Och (2015) collect pairs of words related by any morphological change in an unsupervised manner, then select a vector offset which best explains their observations. There has been subsequent work exploring the vector offset method, finding it unsuccessful in capturing derivational transformations (Gladkova et al., 2016). However, we use more expressive, nonlinear functions to model derivational transformations and report positive results. Gupta et al. (2017) then learn a linear transformation per orthographic rule to solve a word analogy task. Our distributional model learns a function per derivational transformation, not per orthographic rule, which allows it to generalize to unseen orthography.

9 Implementation Details

Our models are implemented in Python using the DyNet deep learning library (Neubig et al., 2017). The code is freely available for download.[4]

Sequence Model The sequence-to-sequence model uses character embeddings of size 20, which are shared across the encoder and decoder, with a vocabulary size of 63. The hidden states of the LSTMs are of size 40.

For training, we use Adam with an initial learning rate of 0.005, a batch size of 5, and train for a maximum of 30 epochs. If after one epoch of the training data, the loss on the validation set does not decrease, we anneal the learning rate by half and revert to the previous best model.

During decoding, we find the top 1 most probable sequence as discussed in Section 6 unless rescoring is used, in which we use the top 10.

Rescorer The rescorer is a 1-hidden-layer perceptron with a `tanh` nonlinearity and 4 hidden units. It is trained for a maximum of 5 epochs.

Distributional Model The DIST model is a 1-hidden-layer perceptron with a `tanh` nonlinearity

[4] https://github.com/danieldeutsch/acl2018

and 100 hidden units. It is trained for a maximum of 25 epochs.

10 Conclusion

In this work, we present a novel aggregation model for derived word generation. This model learns to choose between the predictions of orthographically- and distributionally-informed models. This ameliorates suffix ambiguity and orthographic irregularity, the salient problems of the generation task. Concurrently, we show that derivational transformations can be usefully modeled as nonlinear functions on distributional word embeddings. The distributional and orthographic models aggregated contribute orthogonal information to the aggregate, as shown by substantial improvements over state-of-the-art results, and qualitative analysis. Two ways of incorporating corpus knowledge – constrained decoding and rescoring – demonstrate further improvements to our main contribution.

Acknowledgements

We would like to thank Shyam Upadhyay, Jordan Kodner, and Ryan Cotterell for insightful discussions about derivational morphology. We would also like to thank our anonymous reviewers for helpful feedback on clarity and presentation.

This work was supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017a. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. Association for Computational Linguistics, Vancouver, pages 1–30. http://www.aclweb.org/anthology/K17-2001.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The sigmorphon 2016 shared taskmorphological reinflection. *ACL 2016* page 10.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017b. Paradigm completion for derivational morphology. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 714–720. https://www.aclweb.org/anthology/D17-1074.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1185–1195. http://www.aclweb.org/anthology/N13-1138.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 634–643. http://www.aclweb.org/anthology/N16-1077.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, San Diego, California, pages 8–15. http://www.aclweb.org/anthology/N16-2002.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Arihant Gupta, Syed Sarfaraz Akhtar, Avijit Vajpayee, Arjit Srivastava, Madan Gopal Jhanwar, and Manish Shrivastava. 2017. Exploiting morphological regularities in distributional word representations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 292–297. https://www.aclweb.org/anthology/D17-1028.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational*

Linguistics. Association for Computational Linguistics, Prague, Czech Republic, pages 144–151. http://www.aclweb.org/anthology/P07-1019.

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 569–578. http://www.aclweb.org/anthology/E14-1060.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 555–560. http://anthology.aclweb.org/P16-2090.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 104–113. http://www.aclweb.org/anthology/W13-3512.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 24–31.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331 6014:176–82.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop Papers*. Scottsdale, Arizona. https://arxiv.org/pdf/1301.3781.pdf.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 922–931. http://www.aclweb.org/anthology/N15-1093.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 623–633. http://www.aclweb.org/anthology/N16-1076.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, pages 1–8. http://cogcomp.org/papers/RothYi04.pdf.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation http://cogcomp.org/papers/RothYi07.pdf.

Wolfgang Seeker and Özlem Çetinoglu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics* 3:359–373.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1627–1637. http://www.aclweb.org/anthology/N15-1186.