UNIVERSITY OF CALIFORNIA, SAN DIEGO

Towards Grammaticality and Fluency: Characterizing and Correcting ESL Errors Using Dictionary Random Walks and Other Means

A Thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Randy West

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my adviser Roger Levy and colleague Y. Albert Park for their invaluable mentorship and for giving me this project in the first place. I owe a debt of gratitude as well to my committee members Lawrence Saul and Gary Cottrell for taking the time to review my work, to Markus Dreyer for graciously allowing the use of his V-expectation semiring code, and to the San Diego Supercomputer Center for providing time on the DASH system.

Thanks also to my local San Diego coffee shop for hundreds of hours of internet usage for the price of hundreds of cups of coffee, and to Meredith for gracefully bearing all of the late nights and relentless work schedule.

VITA

2007	Bachelor of Science in Computer Science with Honors in the Major, University of California, Santa Cruz
2011	Master of Science in Computer Science, University of California, San Diego

PUBLICATIONS

Randy West, Y. Albert Park, and Roger Levy (2011). Bilingual Random Walk Models for Automated Grammar Correction of ESL Author-Produced Text. In *Proceedings of the NAACL HLT 2011 Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, Stroudsburg, PA, USA. Association for Computational Linguistics.

ABSTRACT OF THE THESIS

Towards Grammaticality and Fluency: Characterizing and Correcting ESL Errors Using Dictionary Random Walks and Other Means

by

Randy West

Master of Science in Computer Science

University of California, San Diego, 2011

Professor Roger Levy, Chair

We present two novel classes of noisy channel models to address verb infinitive/present participle confusion and word choice production errors in text produced by English as a second language (ESL) authors in an extension of Park and Levy (2011). In our word choice model, which is the primary contribution of this work, we model the English word choices made by ESL authors as a random walk across an undirected bipartite dictionary graph composed of edges between English words and associated words in the author's native language. We use cascades of weighted finite-state transducers (wFSTs) to model language model priors, verb form confusion and random walk-induced noise, and observed sentences, and expectation maximization (EM) to learn model parameters. Additionally, we explore the use of online EM for model training. We show that such models can make intelligent verb form and dictionary-based word substitutions to improve grammaticality and fluency in an unsupervised setting.

Chapter 1

Introduction

How do language learners make word choices as they compose text in a language in which they are not fluent? Anyone who has attempted to learn a foreign language can attest to spending a great deal of time leafing through the pages of a bilingual dictionary. However, dictionaries, especially those without a wealth of example sentences or accompanying word sense information, can often lead even the most scrupulous of language learners in the wrong direction. Consider an example: the English noun "head" has several senses, e.g. the physical head and the head of an organization. However, the Japanese *atama* can only mean the physical head or mind, and likewise *shuchou*, meaning "chief," can only map to the second sense of head. A native English speaker and Japanese learner faced with the choice of these two words and no additional explanation of which Japanese word corresponds to which sense is liable to make a mistake on the flip of a coin.

One could of course conceive of more subtle examples where the semantics of a set of choices are not so blatantly orthogonal. "Complete" and "entire" are synonyms, but they are not necessarily interchangeable. "Complete stranger" is a common two-word phrase, but "entire stranger" sounds completely strange, if not entirely ungrammatical, to the native English speaker, who will correct "entire" to "complete" in a surprisingly automatic fashion. Thus, correct word choice in non-native language production is essential not only to the preservation of intended meaning, but also to fluent expression of the correct meaning.

We propose that word choice production errors on the part of the language learner can be modeled as follows. Given an observed word and an undirected bipartite graph with nodes representing words in one of two languages, i.e. English and the sentence author's native tongue, and edges between words in each language and their dictionary translation in the other (see Figure 1.1 for an example), there exists some function $f \mapsto [0, 1]$ that defines the parameters of a random walk along graph edges, conditioned on the source word. By composing this graph with a language model prior such as an n-gram model or probabilistic context-free grammar, we can "correct" an observed sentence by inferring the most likely unobserved sentence from which it originated.

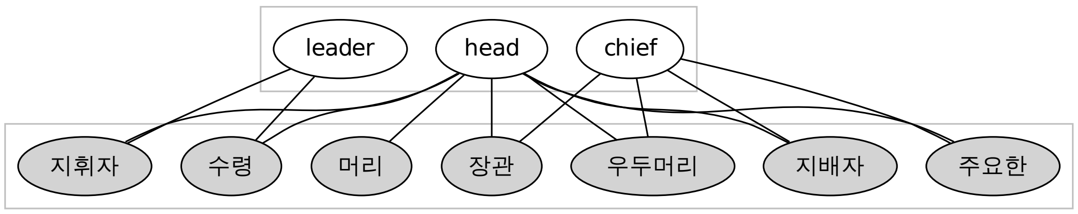

Figure 1.1: Example English-Korean dictionary graph for a subset of the edges out of the English *head*, *leader*, and *chief*.

There are of course many other classes of production errors that ESL authors are wont to commit. Park and Levy (2011) address five such classes, namely spelling, article, preposition, word form, and word insertion errors. We seek in this thesis to extend their work to include generative models of verb infinitive/present participle confusion and the word choice production errors described previously. In addition, we extend their basic framework to employ online instead of batch expectation maximization for training and run a brief results comparison after Liang and Klein (2009).

As a motivational aside, the development of software to correct ESL text is valuable for both learning and communication. A language learner provided instant grammaticality feedback during self-study is less likely to fall into patterns of misuse, and the comprehension difficulties one may encounter when corresponding with non-native speakers would be ameliorated by an automated system to improve text fluency. Additionally, since machine-translated text is often ungrammatical, automated grammar correction algorithms can be deployed as part of a machine translation system to improve the quality of output.

1.1 Related Work

The literature on automated grammar correction is mostly focused on rule-based methods and error identification rather than correction. However, there has been a recent outgrowth in the application of machine translation (MT) techniques to address the problem of single-language grammar correction. Park and Levy (2011) propose a noisy channel model for learning to correct various types of errors, including article and preposition errors, wordform errors, insertion errors and spelling mistakes, to which this thesis is an extension.

Brockett et al. (2006) use phrasal SMT techniques to identify and correct mass noun errors of ESL students with some success, but they correct no other production error classes to our knowledge.

Lee and Seneff (2006) learn a method to aid ESL students in language acquisition by reducing sentences to their canonical form, i.e. a lemmatized form devoid of articles, prepositions, and auxiliaries, and then building an over-specified lattice by reinserting all word inflections and removed word classes. They then score this lattice using a trigram model and PCFG. While this method has many advantages, it does not take into account the full context of the original sentence.

Kok and Brockett (2010) use random walks over bi- and multilingual graphs generated by aligning English sentences with translations in 10 other European languages to learn paraphrases, which they then evaluate in the context of the original sentence. While their approach shares many high-level similarities with our random walk model, both their task, paraphrasing correct sentences, and the details of their methodology are divergent from the present work.

Désilets and Hermet (2009) employ round-trip machine translation from L1 to L2 and back again to correct second language learner text by keeping track of the word alignments between translations. They operate on a very similar hypothesis to that of our random walk model, namely that language learners make overlyliteral translations when the produce text in their second language. However, they go about correcting these errors in a very different way than in the present work, which is novel to the best of our knowledge, and their technique of using error-annotated sentences for evaluation makes a comparison difficult.

1.2 Thesis Organization

The rest of this thesis is arranged as follows. In Chapter 2, we provide the reader with a crash course on the theory and mechanisms employed herein. In Chapter 3, we describe our datasets and the technical details of our software implementation and experimental methodology. In Chapter 4, we detail our noise models and the specifics of training each, and in Chapter 5, we present and analyze our models' results. Finally, we discuss several avenues for future work and conclude in Chapter 6.

Chapter 2

Background

In this chapter, we hope to provide readers with the background necessary to understand the proceeding chapters. In particular, we will provide a brief review of noisy channels (Shannon, 1948) as they apply to statistical machine translation (SMT) (Brown et al., 1993), the expectation maximization (EM) algorithm (Dempster et al., 1977), and weighted finite-state transducers (wFST), especially with regard to their parsing using semirings (Mohri, 2004; Goodman, 1999).

2.1 Noisy Channel Models in SMT

The fundamental problem of statistical machine translation is to determine the most likely translation given an observed sentence, i.e. to determine

$$\hat{\boldsymbol{w}}' = \operatorname*{argmax}_{\boldsymbol{w}'} \mathrm{p}(\boldsymbol{w}'|\boldsymbol{w}), \tag{2.1}$$

where $\boldsymbol{w}$ is a vector of observed words and $\boldsymbol{w}'$ is an associated translation, $\hat{\boldsymbol{w}}'$ being the optimal such vector (Brown et al., 1993). Although the vast majority of the SMT literature focuses on translation between natural languages, grammar correction can similarly be viewed as a translation task from an erroneous "language" to a correct one (Park and Levy, 2011). In a naive implementation, sentences in L1 and L2 would simply be stored in a vast lookup table and associated with numbers indicating the likelihood of translating between them. Such a system would of course be infeasible in practice, both due to the impossibility of enumerating the infinite variety of valid sentences in each language and the impracticality of storing a weight for each such pair, but it serves as a useful starting point for thinking about the translation task from a computational perspective. As a next step, we wish to decompose Equation 2.1 into more manageable components, which we can accomplish using Bayes theorem, as in

$$\mathrm{p}(\boldsymbol{w'}|\boldsymbol{w}) = \frac{\mathrm{p}(\boldsymbol{w'})\,\mathrm{p}(\boldsymbol{w}|\boldsymbol{w'})}{\mathrm{p}(\boldsymbol{w})}. \tag{2.2}$$

The denominator in Equation 2.2 is unimportant in determining $\mathrm{argmax}_{\boldsymbol{w'}}$, and so we can drop it to arrive at the so-called Fundamental Equation of Machine Translation (Brown et al., 1993),

$$\hat{\boldsymbol{w}}' = \operatorname*{argmax}_{\boldsymbol{w'}} \mathrm{p}(\boldsymbol{w'})\,\mathrm{p}(\boldsymbol{w}|\boldsymbol{w'}). \tag{2.3}$$

We have now broken Equation 2.1 into two logical parts, a prior belief about the validity of each translation $\boldsymbol{w'}$, and the probability of $\boldsymbol{w'}$ given the observed sentence $\boldsymbol{w}$. Together, these constitute a noisy channel model from information theory (Shannon, 1948). Information theorists employ such models to reconstruct signals subjected to noise during transmission, and we find that the grammar correction task fits neatly into this paradigm. More concretely, we take the view that the pre-language or native language production of a sentence is subjected to some noise process that perturbs the sentence author's intended form and/or meaning during translation and outputs the observed sentence $\boldsymbol{w}$. Our task, then, is to recover the intended sentence $\boldsymbol{w'}$, and we do so by choosing the most probable such sentence according to a formulation that considers the stand-alone probability of producing each possible correction in conjunction with the probability of the original sentence having been perturbed to $\boldsymbol{w'}$. The former probability is formalized in a language model learned from a large corpus of text, e.g. an n-gram model or probabilistic context-free grammar (PCFG), and the latter in a noise process which may be hard-coded by experts, learned from training data, or some combination of the two.

2.2 Expectation Maximization

The expectation maximization or EM algorithm is a broadly applicable iterative method for computing maximum likelihood or maximum a posteriori estimates[1] from incomplete data (Dempster et al., 1977). In order to motivate EM, we imagine that we have a set of observed data $\boldsymbol{X}$, a set of latent data $\boldsymbol{Z}$, a vector of unknown model parameters $\boldsymbol{\theta}$, and a likelihood function $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{Z}) = \mathrm{p}(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$. We would like use maximum likelihood estimation (MLE) to determine $\hat{\boldsymbol{\theta}}_{MLE}$, the parameter vector which maximizes the marginal likelihood $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{X}) = \mathrm{p}(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{\boldsymbol{Z}} \mathrm{p}(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$. However, a direct application of MLE is often intractable, and so we find ourselves in need of a less direct approach, provided that we can maintain acceptable guarantees of convergence.

With this motivation in mind, let us now turn to the algorithm itself. At a high level, EM consists of an expectation step, which fixes model parameters and calculates the expected value of the log likelihood function with respect to the conditional distribution of $\boldsymbol{Z}$ given $\boldsymbol{X}$ under the current estimate of $\boldsymbol{\theta}$,[2] and a maximization step, which determines the parameter vector maximizing the quantity computed in the previous step. These two steps are applied iteratively until some criteria for convergence is satisfied. The algorithm guarantees convergence to a local maximum within the exponential family of distributions and has similar guarantees for any distribution satisfying the appropriate constraints (Wu, 1983). As we confine ourselves to binomial and multinomial likelihood functions throughout this work, we find these guarantees to be sufficient in all respects.

Although the (batch) EM algorithm as described is guaranteed to converge, the number of iterations required to reach convergence can be arbitrarily large. In order to combat this issue, several online flavors of EM have been developed (Sato and Ishii, 2000; Liang and Klein, 2009). We will focus in particular on a technique known as *stepwise-EM* (sEM). Whereas batch EM computes expectations and maximizes over the entire set of training data at each step, sEM takes a stochas-

[1]We will only cover the application of EM to MLE estimation in this work.

[2]When the likelihood function is in the exponential family, the E-step amounts to simply summing the expectations of sufficient statistics.

tic approach that regards each individual example as a sample and interpolates between estimates for each such example over time. sEM has the same convergence guarantees as batch EM, provided that certain conditions are met regarding the interpolation parameter η_k at each step k, namely that $\sum_{k=0}^{\infty} \eta_k = \infty$ and $\sum_{k=0}^{\infty} \eta_k^2 < \infty$. A common strategy is to take $\eta_k = (k+2)^{-\alpha}$ for some $0.5 < \alpha \leq 1$, where a larger value of α indicates a more conservative update strategy. In order to further bolster the stability of sEM, one may wish to perform updates using a small set of examples, referred to as a mini-batch, instead of a single example at a time. This is the strategy that we shall employ whenever we use online EM in this work. Specifically, we set $\alpha = 1$ and use a mini-batch size of 100 examples, which is at most 1% of the total training data used in each experiment.

2.3 Weighted Finite-State Transducers and Semiring Parsing

Weighted finite-state transducers (wFST) are a generalization of finite-state automata (FSA) where state transition arcs are annotated with an output labels and semiring weight elements in addition to the standard FSA input label (Mohri, 2004). wFSTs enjoy a well-developed theory, elegance, and wealth of standard algorithms that make them ideal for developing compact mappings between different information sources, and their embedded weights offer a simple mechanism by which to represent uncertainty or variability in said mapping. Moreover, there exists a robust selection of software packages for implementing and processing wFSTs, notably `OpenFST`, which we make use of in this work (Allauzen et al., 2007).

In order to formally introduce wFSTs, we must begin by describing the elements and operations of a semiring.

Definition 1. *A system* $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ *is a semiring if:* $(\mathbb{K}, \oplus, \bar{0})$ *is a commutative monoid with identity element* $\bar{0}$*;* $(\mathbb{K}, \otimes, \bar{1})$ *is a monoid with identity element* $\bar{1}$*;* $\otimes$ *distributes over* $\oplus$*; and* $\bar{0}$ *is an annihilator for* $\otimes$*:* $\forall a \in \mathbb{K}, a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$ *(Mohri, 2004).*

Table 2.1: Some common semirings. $\oplus_{\log}$ is defined by: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$ (Mohri, 2004).

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0

Some common semirings are listed in Table 2.1. It is useful to imagine the various uses of these semirings in a graph setting. When each edge in a directed graph is labeled with a semiring element, we can perform natural computations on the graph simply by invoking the semiring operators. For example, if the paths through a graph represent a probability distribution, then the first natural operation is to compute the probability mass associated with each such path. This is the product of the arc weights in the path, and we can compute it using $\bigotimes_{\gamma \in \pi} \gamma$, where γ is an arc weight and π is a path, over the probability semiring. Likewise, if we wish to determine the total probability of following several different paths, we can do so using $\bigoplus_{\pi \in \text{paths}} \bigotimes_{\gamma \in \pi} \gamma$ over the same semiring. If we wish instead to compute the shortest path in a graph, then the tropical semiring is well-suited for our purposes. For further reference on semirings and their uses in parsing grammars, we direct the reader to Goodman (1999).

We will now proceed with formally defining a wFST.

Definition 2. *A weighted finite-state transducer T over a semiring $\mathbb{K}$ is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where Σ is the finite input alphabet of the transducer; Δ is the finite output alphabet; Q is a finite set of states; $I \subseteq Q$ the set of initial states; $F \subseteq Q$ the set of final states; $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ a finite set of transitions; $\lambda : I \rightarrow \mathbb{K}$ the initial weight function; and $\rho : F \rightarrow \mathbb{K}$ the final weight function (Mohri, 2004).*

We will further discuss the two fundamental wFST algorithms that we make use of in this work, composition and ϵ-removal (Mohri, 2004, 2001). Composition is an algorithm used to create complex transducers by combining several simpler ones. The composition operation, which we denote $T_1 \circ T_2$ for two transducers T_1 and T_2 with $\Delta_{T_1} = \Sigma_{T_2}$, performs this combination by discovering all paths through T_1 whose output symbols correspond exactly to the input symbols of a path in T_2. The resultant transducer is exactly the description of all such matching paths and is defined for all $(x, y) \in \Sigma^*_{T_1} \times \Delta^*_{T_2}$ by

$$[T_1 \circ T_2](x, y) = \bigoplus_z T_1(x, z) \otimes T_2(z, y),$$

where $T_i(x, y)$ indicates the semiring weight element associated with an $x : y$ input/output string transduction[3] in T_i and $[T_j](\alpha, \beta)$ the cell in a transduction weight matrix associated with T_j and corresponding to the transduction from α to β. In the OpenFST implementation, the complexity of composition is $O(v_1 v_2 d_1 (\log d_2 + m_2))$ in time and $O(v_1 v_2 d_1 m_2)$ in space, where v_i is the number of transducer states, d_i the maximum number out-degree, i.e. number of arcs leaving any given state, and m_i is the maximum multiplicity, i.e. number of times a label is repeated at any given state (Allauzen et al., 2007). This measure of complexity further assumes that the arcs in second transducer are presorted, an operation which itself takes $O(vd \log d)$ time and $O(d)$ space.

The ϵ-removal algorithm transforms a transducer A containing $\epsilon : \epsilon$, i.e. empty string to empty string transitions, into an equivalent transducer B containing no such transitions (Mohri, 2001). In general, ϵ-removal allows for more efficient parsing, and it also plays a role in other fundamental transducer optimization algorithms such as determinization. For our purposes in this work, we will regard it as a method of transferring the combined arc weights of a transducer made up solely of $\epsilon : \epsilon$ transitions into a single initial/final state, which is an essential component of our noise model parameter training process (see Chapter 3 for details). In the OpenFST implementation, the complexity of the ϵ-removal operation for the (acyclic) transducers over which we employ it is $O(v^2 + ve)$ in time and $O(ve)$ in space, where v is the number of transducer states and e is the total number of arcs (Allauzen et al., 2007).

[3]We emphasize *string*, as opposed to symbol transduction, which indicates a string of symbols being processed completely starting from some initial state in I to acceptance in some final state in F.

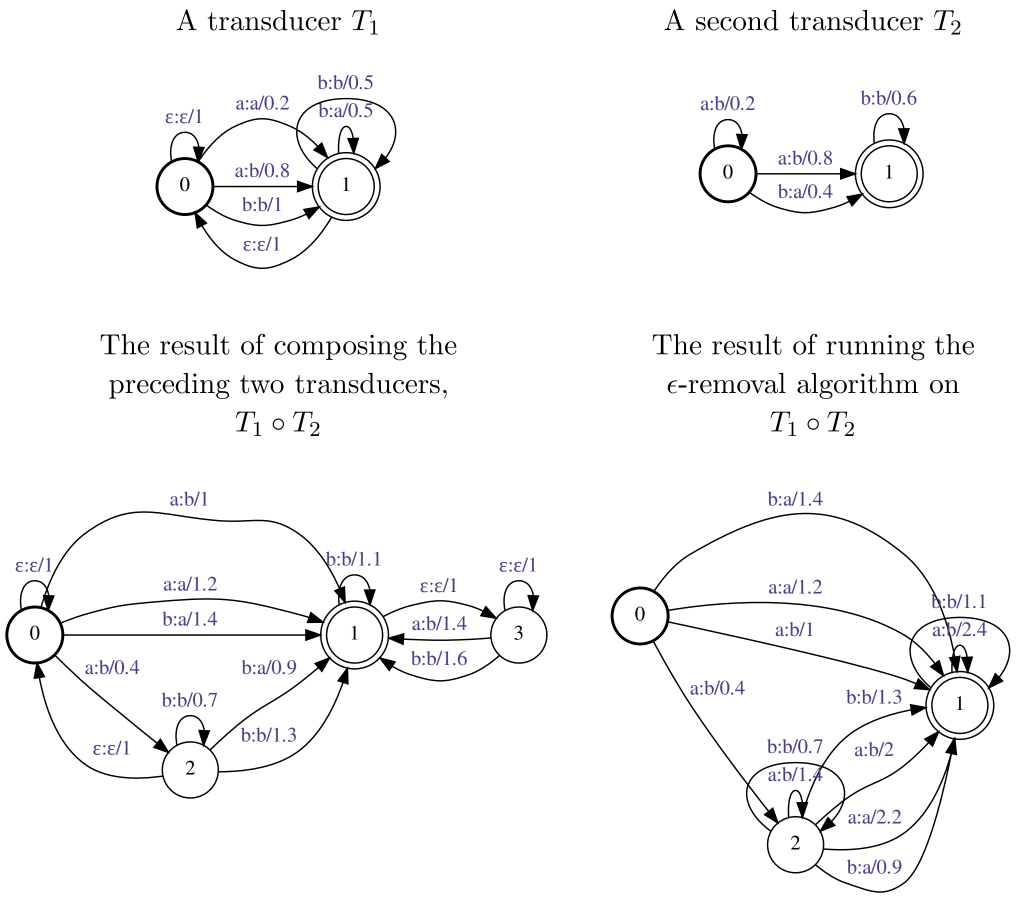

Figure 2.1: Two weighted finite-state transducers over the *tropical* semiring and the result of the composition and ϵ-removal operations on them.

In conclusion, we present the reader with several illustrative graphical representations of weighted finite-state transducers and their operations in Figure 2.1.

Chapter 3

Design and Implementation

In this chapter, we will present the dataset of Park and Levy (2011) along with a technical overview of their training, decoding, and evaluation systems. We will additionally provide a description of our English-Korean dictionary and associated preprocessing, the implementation of the convex optimization process used during training to infer noise model parameters from sufficient statistics for two of our random walk models, our online EM methodology and convergence testing, and the Amazon Mechanical Turk-based judgment task developed for further evaluating our final decoding results (West et al., 2011).

3.1 Datasets

3.1.1 Training, Development, and Evaluation Sets

We use the dataset of Park and Levy (2011), a collection of approximately 25,000 essays comprised of 478,350 sentences scraped from web postings made by Korean ESL students studying for the Test of English as a Foreign Language (TOEFL). Of these, we randomly select $10,000$ sentences for training, 504 as a development set, and 1,007 held out for final model evaluation.

3.1.2 English-Korean Dictionary

We are unfortunately unaware of any freely available, downloadable English-Korean dictionary databases. As such, we compiled the English-Korean dictionary used throughout this work by scraping the contents of `http://endic2009.naver.com`, a widely-used and trusted online dictionary source in South Korea. Then, conceptualizing the dictionary as a bipartite graph (see Figure 1.1 for an example), we developed a preprocessing algorithm to produce a mapping from English words to other candidate English words reachable from the source word in a random walk of length 2, subject to the constraint that each candidate shares some of the syntactic class and morphological inflectional features of the source word.

Compiling the Dictionary

More pedantically, we cached the `HTML` and `Javascript` contents of every page at `http://endic2009.naver.com/endic.nhn?docid=x` for all $\{x \mid x \bmod 10 = 0 \;\wedge\; 10 \leq x \leq 1307390\}$ and sliced out the contents of the sections labeled `<div id="entryBody333">` and the value of the tag labeled `<input name="query">`. As the text on each page is optimized for display and includes a wide variety of somewhat inconsistent visual cues to distinguish between example sentences, part of speech markers, and actual definitions, we were forced to employ a complex heuristic to parse out the needed information. During development, we incrementally chose a random sample of pages and compared our results with the actual page content. When heuristic revisions were warranted, we made them and performed complete regression tests on previously sampled pages to ensure consistency. Thus, we assert without further clarification that we are highly confident in the correspondence between our gathered dictionary and the underlying source of the `naver.com` dictionary.

Preprocessing the Dictionary

In order to facilitate speedy processing and moreover to exactly bound the number of candidate words reachable from each English source word in a random walk of length 2 along our dictionary graph, we preprocess the dictionary as detailed in Algorithm 1. In addition to speed and bounding concerns, we must handle the non-trivial peculiars of arbitrary lookups in a roughly lemmatized dictionary and preservation of word forms through random walks. We perform this analysis using the `CELEX` database (Baayen et al., 1995), which provides interfaces for mapping arbitrary English words to their lemmas, querying for lemma syntactic (sub)classes, and discovering the morphological inflectional features of arbitrary words, and a standard stop word filter to remove function and other overly common words (see Appendix A for a full list of the stop words used).

Algorithm 1 Preprocess the dictionary graph G to create a mapping of English words $w_i \rightarrow \{w_j, ...\}, i \neq j$ for all w_j reachable from w_i in a random walk of length 2 along G, subject to several constraints.

```
Let $G = (V, E)$ be an undirected dictionary graph, $L$ the set of words in a language model, $B$ a set of stop words, and $C$ the map of words to random walk candidates, initially empty
for words $w_i \in L$ do
  Let $I$ be the set of inflectional features of $w_i$, and $C_i$ the set of random walk candidates for $w_i$, initially $\{\}$
  for lemmas $l$ of $w_i$ do
    if $l \notin B$ then
      Let $S$ be the set of syntactic classes of $l$
      for $l'$ generated from a random walk of length 2 in $G$ from $l$ do
        if $l' \notin B \wedge S \cap \{\text{syntactic classes of } l'\} \neq \{\}$ then
          for words $w'$ related to $l'$ do
            if $I \cap \{\text{inflectional features of } w'\} \neq \{\} \wedge w' \in L$ then
              $C_i \leftarrow C_i \cup \{w'\}$
            end if
          end for
        end if
      end for
    end if
  end for
  $C \leftarrow C \cup \{w_i \rightarrow C_i\}$
end for
return $C$
```

3.2 Language and Sentence Models

Prior to describing our training and decoding processes, it is necessary to first characterize our language model and method of implementing sentences and language priors as wFSTs.

Language Model

For our language model, we use a Kneser-Ney smoothed trigram model learned from a version of the British National Corpus modified to use Americanized spellings (Chen and Goodman, 1996; Burnard, 1995). The implementation of an n-gram model as a wFST requires that each state represent a context, and so one must necessarily instantiate arcs for all words in the alphabet from each state. In order to reduce model size and minimize memory usage, it is standard practice to remove relatively uninformative higher-order n-grams from the model, but under the wFST regime one cannot, for example, remove some trigrams from a bigram context without removing all of them.[1] Instead, we retain only the 1,000 most informative bigram *contexts*, as measured by the Kullback-Leibler divergence between each bigram context and its unigram counterpart. This is in contrast to standard cutoff models, which remove n-grams occurring less than some cutoff number of times in the corpus. See Figure 3.1 for an example language model transducer.

[1]Technically, one could add arcs to a w_{i-2}, w_{i-1} context state with probabilities conditioned only on w_{i-1}, but the point here is that since the number of arcs leaving any given state is predetermined by the size of our symbol alphabet, we do not gain any finite-state machine size reduction without removing states.

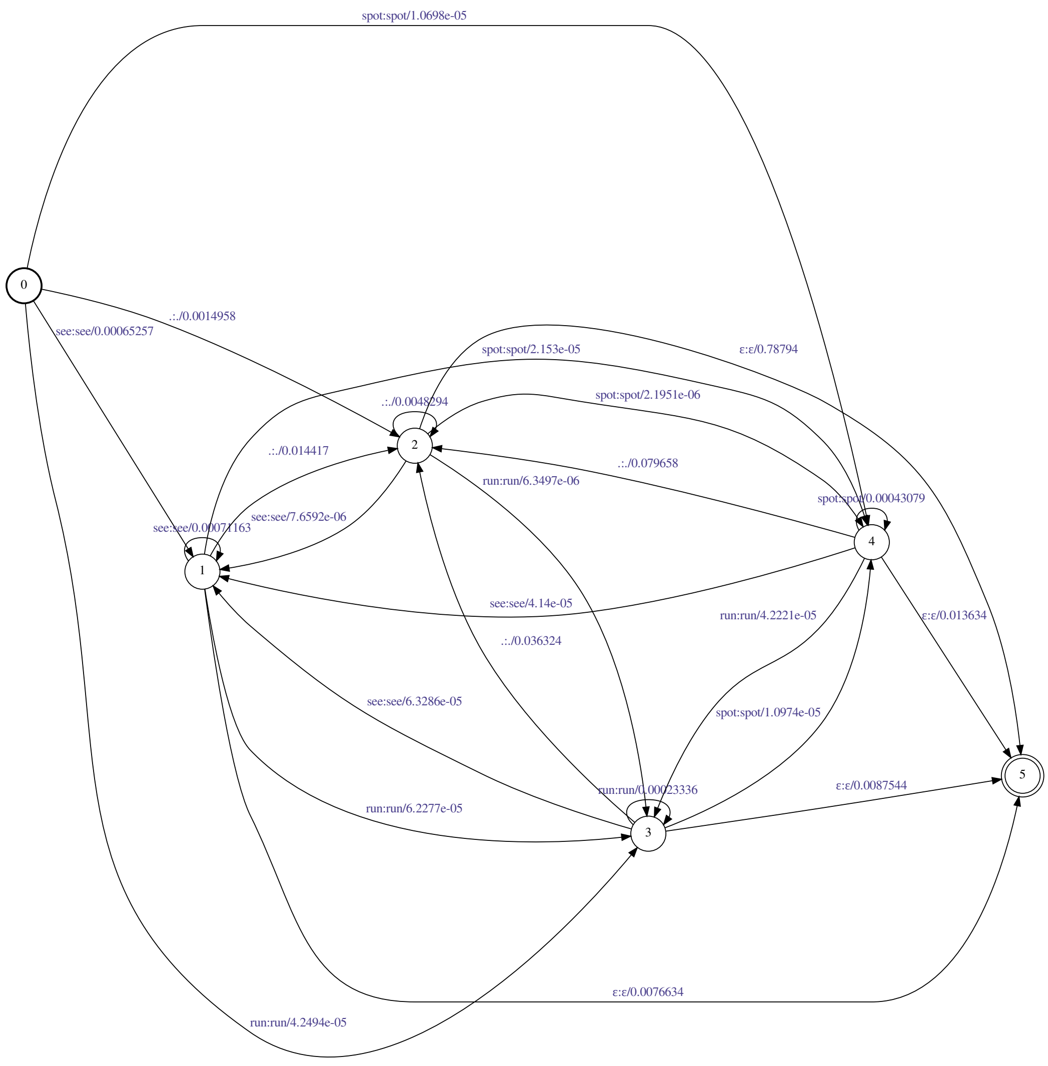

Figure 3.1: An example language model for the sentence "see spot run."

3.2.1 Sentence Models

Sentences are modeled as identity transducers, i.e. wFSTs with $n+1$ states for a sentence of length n and a single arc between each state $0 \leq i < n$ and state $i+1$ labeled with input and output token i from the sentence and weight $\bar{1}$.

3.3 Training

In this section, we describe the noise model parameter training process due to Park and Levy (2011) and Eisner (2002). Additionally we describe our use of the BFGS and L-BFGS-B algorithms (Fletcher, 1970; Byrd et al., 1995; Nocedal and Wright, 1999) to perform quasi-Newton optimization of parameters from sufficient statistics in the M-step.

At a high level, we train our models by holding language model parameters constant and using expectation maximization (Dempster et al., 1977) to learn noise model parameters. This process proceeds as outline in Section 2.2, namely by beginning with initial parameter values in the M-step, building each noise model and gathering expected arc traversal counts over the training set as sufficient statistics for the E-step, and then updating parameter values and repeating the process until convergence. We implement the training system as a client-server model in a combination of `Java`, `R` (R Development Core Team, 2010), `C`, and `C++`, notably using `OpenFST` (Allauzen et al., 2007) and the V-expectation semiring code of Dreyer et al. (2008).

3.3.1 M-step

We begin the training process by initializing all parameters to some appropriate values. In all subsequent iterations, we use the expected arc traversal counts computed during the E-step to recompute the parameters for each noise model by maximizing the conditional probability of the expected counts with respect to the parameters. More concretely, we wish to compute $\operatorname{argmax}_{\boldsymbol{\lambda}} f(\boldsymbol{\gamma}; n, \boldsymbol{p})$, where $\boldsymbol{\gamma}$ is a vector of sufficient statistics, $\boldsymbol{\lambda}$ is a vector of noise model parameters, $\boldsymbol{p}$ is the vector of probabilities which may be functions of $\boldsymbol{\lambda}$ and are associated with each arc counted in $\boldsymbol{\gamma}$, n is the number of training examples considered, and f is the probability mass function of the multinomial distribution, formally defined as

$$f(\boldsymbol{\gamma}; n, \boldsymbol{p}) = \begin{cases} \frac{n!}{\prod_i \gamma_i!} \prod_i p_i^{\gamma_i} & \text{if } \sum_i \gamma_i = 1 \\ 0 & \text{otherwise} \end{cases} . \tag{3.1}$$

The combinatorial term and all terms in the product involving some p_i which is not a function of $\boldsymbol{\lambda}$ in Equation 3.1 are independent of $\boldsymbol{\lambda}$ in the argmax, and so we may disregard them for simplicity.

The maximization process then proceeds in the standard way. Wherever possible, we compute the gradient of $f(\boldsymbol{\gamma}; n, \boldsymbol{p})$ with respect to each λ_j, set it to 0, and solve for λ_j. In a simple coin flip model, i.e. one where $p_i = \lambda_j$ and $p_{i+1} = 1 - \lambda_j$, this amounts to setting $\lambda_j = \frac{\gamma_i}{\gamma_i + \gamma_{i+1}}$, the normalized expected count of the arc labeled with λ_j.

In more complex models where a closed form computation is not possible, however, we must employ an optimization algorithm to compute the argmax described previously. For this, we compute closed-form gradients and plug them into the `optim` function provided by the `R` language (R Development Core Team, 2010), specifically using the BFGS and L-BFGS-B methods (Fletcher, 1970; Byrd et al., 1995; Nocedal and Wright, 1999). For additional speed, we implement all functions and gradients in a `C` language shared library, calls to which we wrap in `R` functions for use with `optim`.

3.3.2 E-step

For our E-step, we begin by reading in a list of training sentences and creating an identity transducer for each. Then, using as our alphabet the set of symbols from all training sentences in this batch, we work backwards to create a transducer for each specified noise model. As each noise model may add additional symbols, its input alphabet is always used as the output alphabet of the proceeding transducer. Throughout this process, we annotate relevant arcs with the V-expectation semiring of Eisner (2002), which we define for reference in Table 3.1, in order to keep track of the arcs traversed in any given transduction. After noise model construction is complete, we use the input alphabet of the final model to create a transducer modeling the prior probability of all possible n-gram combinations from the alphabet. In order to facilitate the computation of sufficient statistics, we additionally replace all language model input symbols and sentence model output symbols with the empty symbol ϵ.

Table 3.1: The V-expectation semiring of Eisner (2002). $\mathrm{val}(\pi) \in V$ is a vector space representing the arcs, features, or coin flips encountered along some path π.

$$\begin{aligned}
\mathbb{K} &\in \mathbb{R}_{\geq 0} \times V \\
(p_1, v_1) \oplus (p_2, v_2) &\overset{\mathrm{def}}{=} (p_1 + p_2, v_1 + v_2) \\
(p_1, v_1) \otimes (p_2, v_2) &\overset{\mathrm{def}}{=} (p_1 p_2, p_1 v_2 + v_1 p_2) \\
\text{if } p^* \text{ defined, } (p, v)^* &\overset{\mathrm{def}}{=} (p^*, p^* v p^*) \\
\bar{0} &\overset{\mathrm{def}}{=} (0, \mathbf{0}) \\
\bar{1} &\overset{\mathrm{def}}{=} (1, \mathbf{0})
\end{aligned}$$

Having completed transducer construction, we compose the language and noise models with each sentence model in turn, which produces a transducer with only ϵ-labeled arcs, and use ϵ-removal to move expectation information into a single state from which we can easily read off the expected traversal count of each arc labeled previously thanks to the V-expectation semiring's bookkeeping (Eisner, 2002; Mohri, 2001). See Figure 3.2 for a graphical depiction of the composition and ϵ-removal processes. We repeat this process over a batch of training sentences and add the results together to yield a final vector of expected counts, which we pass back to the M-step. This process goes back and forth from E- to M-step until the parameters converge within some threshold.

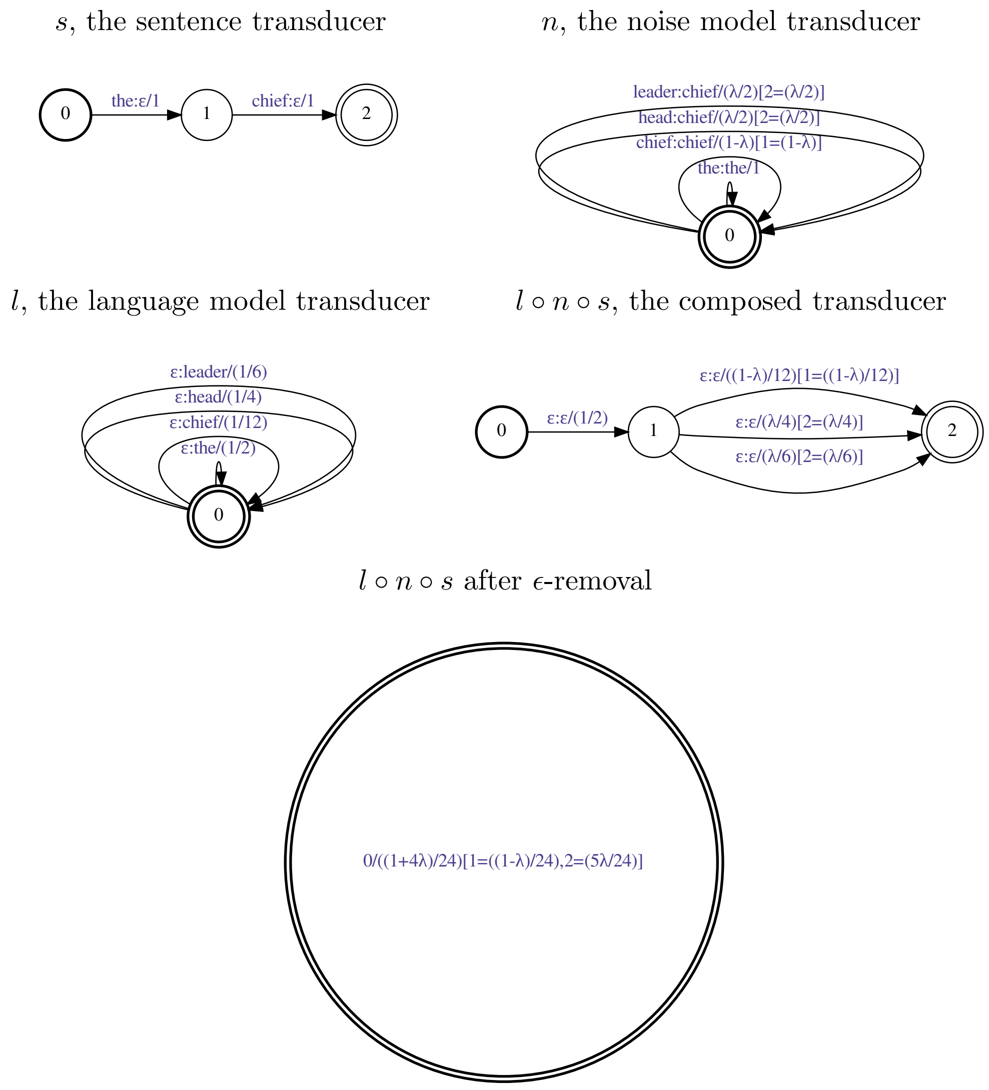
s, the sentence transducer
0
the:ε/1
1
chief:ε/1
2
n, the noise model transducer
leader:chief/(λ/2)[2=(λ/2)]
head:chief/(λ/2)[2=(λ/2)]
chief:chief/(1-λ)[1=(1-λ)]
the:the/1
0
l, the language model transducer
ε:leader/(1/6)
ε:head/(1/4)
ε:chief/(1/12)
ε:the/(1/2)
0
$l \circ n \circ s$, the composed transducer
0
ε:ε/(1/2)
1
ε:ε/((1-λ)/12)[1=((1-λ)/12)]
ε:ε/(λ/4)[2=(λ/4)]
ε:ε/(λ/6)[2=(λ/6)]
2
$l \circ n \circ s$ after ϵ-removal
0/((1+4λ)/24)[1=((1-λ)/24),2=(5λ/24)]

Figure 3.2: Simplified depiction of the E-step for the sentence “the chief” under one of our random walk noise models. The pictured transducers are the observed sentence s, a noise model n with parameter λ, a unigram language model l representing the normalized frequency of each word, the fully composed model $l \circ n \circ s$, and $l \circ n \circ s$ after ϵ-removal.

3.3.3 Online EM Settings and Testing for Convergence

Liang and Klein (2009) report that online expectation maximization not only converges in fewer iterations but also tends to find better parameter choices. We attempt to test their assertion by comparing the performance of batch versus online EM in several of the experiments reported in Chapter 5. We note that this is an improvement over Park and Levy (2011) and West et al. (2011), which use only batch EM. As reported in Section 2.2, we interpolate between parameter values computed in mini-batches of 100 example sentences using $\eta_k = (k+2)^{-\alpha}$ and $\alpha = 1$, where k is the number of mini-batches thus far covered and α is set to interpolate in the most conservative manner possible.

We test for convergence after each complete iteration by computing the amount by which the ratio of previous to current iteration values for each model parameter deviates from 1. If this deviation is less than some threshold Δ, then we say that the model has converged and discontinue training. More concretely, we say that the model has converged if $\forall \boldsymbol{\lambda}, |\frac{\lambda_{i-1}+\epsilon}{\lambda_i+\epsilon} - 1| \leq \Delta$, where ϵ is some small value (`Java`'s `Double.MIN_VALUE`) used to prevent division by 0.

3.4 Decoding

The decoding or inference process is performed in much the same way as the E-step of training. The main difference for decoding is that we use the negative log Viterbi semiring for computing shortest paths instead of the V-expectation semiring. We first build a new noise model for each sentence using the parameter values learned during training. Then, the language, noise, and sentence models (sans the ϵ substitutions performed during training) are composed together, and the shortest, i.e. most probable path is computed. The resulting transducer contains a single path from which we can read off the observed sentence from the output symbols and the corrected sentence from the input symbols.

3.5 Evaluation

The most probable unobserved sentence $\boldsymbol{w'}$ from which the observed sentence $\boldsymbol{w}$ was generated under our model, $\hat{\boldsymbol{w}}' = \text{argmax}_{\boldsymbol{w'}} \, \text{p}(\boldsymbol{w'}) \, \text{p}(\boldsymbol{w}|\boldsymbol{w'})$, can be read off from the input of the transducer produced during the decoding process. In order to evaluate its quality versus the observed ESL sentence, we use the `METEOR`[2] and `BLEU` evaluation metrics for machine translation (Lavie and Agarwal, 2007; Papineni et al., 2002). This evaluation is performed using a set of human-corrected sentences gathered via Amazon Mechanical Turk (AMT), an online service where workers are paid to perform a short task, and further filtered for correctness by an undergraduate research assistant. 8 workers were assigned to correct each sentence from the development and evaluation sets described in Section 3.1, and so after filtering we had 8 or fewer unique corrected versions per sentence available for evaluation. We note that the use of `METEOR` and `BLEU` is justified inasmuch as the process of grammar correction is translation from an ungrammatical "language" to a grammatical one (Park and Levy, 2011). However, it is far from perfect, as we shall see in Chapter 5.

While human evaluation is far too costly to attempt at every step during development, it is worthwhile to examine our corrections through a human eye for final evaluation, especially given the somewhat tenuous suitability of `METEOR` and `BLEU` for our evaluation task. In order to facilitate this, we designed a simple task, again using AMT, where native English speakers are presented with side-by-side ESL and corrected sentences and asked to choose which is more correct. Workers are instructed to "judge whether the corrected sentence improves the grammaticality and/or fluency of the ESL sentence without changing the ESL sentence's basic meaning." They are then presented with two questions per sentence pair:

1. *Question:* "Between the two sentences listed above, which is more correct?" *Answer choices:* "ESL sentence is more correct," "Corrected sentence is more correct," "Both are equally correct," and, "The sentences are identical."

2. *Question:* "Is the meaning of the corrected sentence significantly different from that of the ESL sentence?"

[2] Although the `METEOR` "synonymy" module may initially seem appropriate to our evaluation task for the random walk models, we find that it does little to improve or clarify evaluation results. For that reason, and moreover since we do not wish for differing forms of the same lemma to be given equal weight in a grammar correction task, we instead use the "exact" module for all evaluation in this work.

Answer choices: "Yes, the two sentences do not mean the same thing," and, "No, the two sentences have roughly the same meaning."

Each task is 10 sentences long, 3 of which are identical filler sentences. When a worker mislabels more than one sentence as identical in any single task, the results for that task are thrown out and resubmitted for another worker to complete. We additionally require that each sentence pair be judged by 5 unique, U.S.-based workers. We reproduce the full text of the task in Appendix B.

Chapter 4

Noise Models

Park and Levy (2011) report five different noise models, namely those modeling spelling, article, preposition, word form, and word insertion errors, which they use in a batch EM framework to correct mistakes. In this chapter, we describe two additional noise model classes, a verb infinitive $\leftrightarrow$ present participle model, and several different parametrizations of the bilingual dictionary random walk model first reported in West et al. (2011), which is the primary contribution of this work.

4.1 Infinitive $\leftrightarrow$ Present Participle Model

4.1.1 Motivation

The word form error model of Park and Levy (2011) works by over-generating the observed sentence to include an arc transducing from each form of an observed word to the word itself. This includes the case of verb infinitive to present participle forms and vice versa, e.g. *going* $\leftrightarrow$ *go*, but it misses a somewhat more subtle case. When ESL authors mistake the infinitive for the present participle form, they may still remember to include the preposition *to* before the infinitive. But it is not sufficient to change, for example, *to go* to *to going*; we must also delete the preposition.

We will hereafter refer to our model of infinitive $\leftrightarrow$ present participle production errors as `inf_pp`. A summary of the types of errors handled by Park and Levy (2011)'s word form error model versus those handled by our model is presented in Table 4.1.

Table 4.1: A summary of the types of errors corrected in our `inf_pp` noise model versus those corrected in the word form error model of Park and Levy (2011). The verb *to go* is used as an example to illustrate transductions.

Error Type	**Word Form**	**Infinitive $\leftrightarrow$ Present Participle**
to going $\rightarrow$ to go	Yes	No
go $\rightarrow$ going	Yes	No
to go $\rightarrow$ going	No	Yes
going $\rightarrow$ to go	No	Yes

4.1.2 Implementation

We model errors by assuming that there is a uniform probability σ of erroneously producing *to infinitive* in place of *present participle* and a separate uniform probability λ of committing the opposite production error. Where participle forms map to more than one infinitive form, the transduction probability is appropriately normalized such that $\sum_w \mathrm{p}(w|w') = 1$. We implement our model as a wFST in four states: where the word being parsed is w_i, we have two states representing $w_{i-1} \neq$ *to*, one which transduces *participle* $\rightarrow$ *to infinitive* and the other the identity, and two states representing $w_{i-1} =$ *to*, one which transduces *to infinitive* $\rightarrow$ *participle* and the other the identity. All symbols which are not *to*, *inf*, or *pp* are inserted as identity arcs between the two identity states. See Figure 4.1 for a graphical representation of the transducer.

4.1.3 Training

Four statistics are gathered during the E-step, namely the expected count of arc traversals for each of *to inf* $\rightarrow$ *pp*, *pp* $\rightarrow$ *to inf*, and their identities. In the M-step, we wish to maximize the probability of these statistics $\boldsymbol{\gamma}$ with respect to the parameters σ and λ, which in this case is just the closed form relative frequency

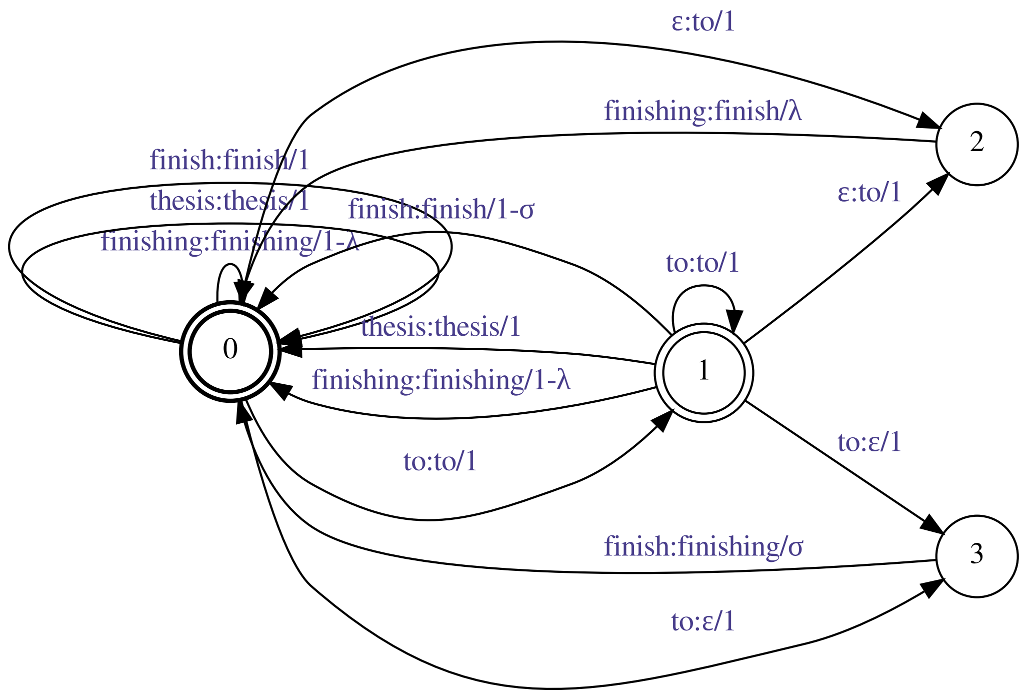

Figure 4.1: A graphical representation of the *to infinitive* ↔ *present participle* noise model transducer as described in Section 4.1.2. We use the verb *finish* and the non-verb *thesis* to illustrate this example.

estimate for multinomials (see Section 3.3.1 for additional details). For clarity, we restate the closed form derivation here, beginning with the maximization equation,

$$\begin{aligned}\hat{\sigma} &= \operatorname*{argmax}_{\sigma} \mathrm{p}(\gamma_\sigma, \gamma_{1-\sigma}|\sigma)\\ &= \operatorname*{argmax}_{\sigma} \sigma^{\gamma_\sigma}(1-\sigma)^{\gamma_{1-\sigma}}.\end{aligned}$$

Next, we convert to log space for ease of manipulation, as in

$$\log(\sigma^{\gamma_\sigma}(1-\sigma)^{\gamma_{1-\sigma}}) = \gamma_\sigma \log(\sigma) + \gamma_{1-\sigma}\log(1-\sigma).$$

We then take the derivative with respect to σ,

$$\frac{\partial}{\partial\sigma} = \frac{\gamma_\sigma}{\sigma} - \frac{\gamma_{1-\sigma}}{1-\sigma},$$

which we set equal to 0 and solve for σ, yielding the update equation

$$\hat{\sigma} = \frac{\gamma_\sigma}{\gamma_\sigma + \gamma_{1-\sigma}}.$$

Noting that any normalization constant disappears in the gradient, a similar process holds for the computation of λ.

4.2 Random Walk Models

Although the models presented in Park and Levy (2011) and Section 4.1 target a wide range of spelling and grammar errors commonly observed in ESL author-produced text, they ignore a much broader and potentially more "difficult" class of production errors, that of word choice. The ability to select the appropriate word or phrase to express one's intent in context is one of the cornerstones of language fluency, and yet the ambiguity inherent in the way natural languages map to one another presents language learners with a significant obstacle in the pursuit of fluent composition.

We propose that word choice production errors can be modeled as follows. Given the dictionary graph described in Section 3.1.2, there exists some function $f \mapsto [0, 1]$ that defines the parameters of a random walk along graph edges, conditioned on the source word. By implementing this graph as a wFST and composing it with our language model prior and sentence transducers, we can "undo" word choice errors by considering a range of candidate words for each observed word consistent with a generative model of word choice and evaluate them in sentence context using the language model. We parametrize the generative model in three different ways, initially with a uniform probability of generating an erroneous word, and subsequently with two different distributions sensitive to observed and unobserved unigram word frequency.

4.2.1 Implementation

For our model of word choice error, we are tasked with creating a transducer which obeys the semantics of a random walk of length 2 across our dictionary graph, subject to the constraints detailed in Section 3.1.2. For this, we can use a single initial/final state with arcs labeled with unobserved words as input, observed words as output, and a weight defined by some function f that governs the parameters of a random walk across our dictionary graph. We will present three different parametrization of f in the proceeding sections, but the structure of the transducer implementing each model does not vary. See Figure 4.2 for an example.

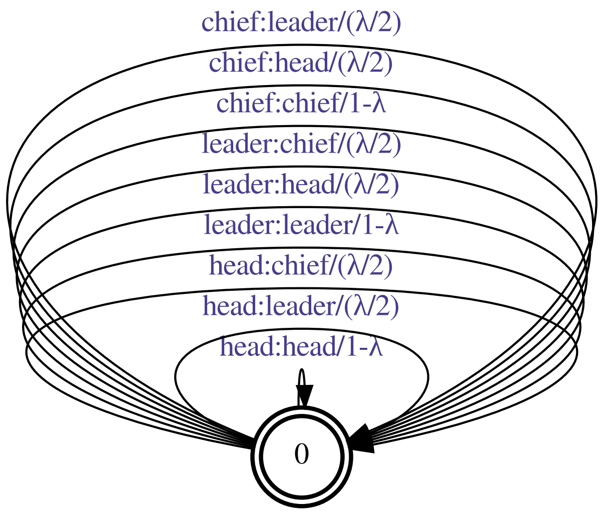

Figure 4.2: An example random walk transducer using the uniform walk probability parametrization described in Section 4.2.2.

4.2.2 Uniform Replacement Model

Motivation

For our first model parametrization, which we will refer to subsequently as `rw_unif`, we assume that the probability of arriving at some word $w' \neq w$ after a random walk of length 2 from an observed word w is uniform across all w. This is perhaps not the most plausible model, but it serves as a baseline by which we can evaluate more complex parametrizations.

Parametrization

More concretely, we use a single parameter λ modeling the probability of walking two steps along the dictionary graph from an observed English word w to its Korean definition(s), and then back to some other English word $w' \neq w$. Since we treat unobserved words as transducer input and observed words as output, λ is normalized by $|\{w|w \neq w' \wedge w \in \text{walk}(w')\}|$, i.e. the number of words reachable along a random walk from w', and $\text{p}(w|w) = 1 - \lambda$ such that $\sum_w \text{p}(w|w') = 1$.

Training

As with `inf_pp`, `rw_unif` is a coin flip model, and so we need only gather two statistics during the E-step, *replace* and *no fix*. We can use these in the same fashion as detailed in Section 4.1.3 to derive a closed form update for our single parameter λ.

4.2.3 Adding One-Way Word Frequency Sensitivity

Motivation

For our second random walk parametrization, which we will hereafter refer to as `rw_freq_unobs`, we hypothesize that there is an inverse relationship between unobserved word frequency and random walk path probability. We motivate this by observing that when a language learner produces a common word, it is likely that she either meant to use that word or used it in place of a rarer word that she did not know. Likewise, when she uses a rare word, it is likely that she chose it above any of the common words that she knows. If the word that she chose was erroneous, then, it is most likely that she did not mean to use a common word but could have meant to use a different rare word with a subtle semantic difference. Hence, we should always prefer to replace observed words, regardless of their frequency, with rare words unless the language model overwhelmingly prefers a common word.

Parametrization

In order to model this hypothesis, we introduce a second parameter α, which we use as in

$$\begin{aligned} \mathrm{p}(w|w) &= 1 - \lambda e^{-\operatorname{freq}(w)^{\alpha}} \\ \mathrm{p}(w|w') &= \frac{\lambda e^{-\operatorname{freq}(w')^{\alpha}}}{|\{w | w \neq w' \wedge w \in \operatorname{walk}(w')\}|}, \end{aligned}$$

where $\operatorname{freq}(w')$ is the unigram frequency of the unobserved word w' from our language model. When $\alpha > 0$, this parametrization will correctly implement the inverse relationship between unobserved word frequency and random walk path probability described previously.

Training

Although `rw_freq_unobs` may be regarded as a coin flip model, we now have a different coin for every unique unobserved word frequency. Thus, we find that we must keep track of a much larger number of sufficient statistics in the E-step in order to successfully update α and λ in the M-step. Specifically, we track γ_{i0} and γ_{i1} for each word w_i, where the subscript $i0$ signifies *no fix* and $i1$ indicates *replace*.

Unfortunately, we are unaware of a closed form procedure for performing parameter updates in this case.[1] As such, we must resort to the convex optimization procedure detailed in Section 3.3.1. We wish as before to maximize the conditional probability of the statistics vector $\boldsymbol{\gamma}$ with respect to the parameters α and λ. This is equivalent to maximizing the log probability of $\boldsymbol{\gamma}$, which we perform using the quasi-Newton BFGS method with `R`'s `optim` function. Although we technically want to perform a non-linearly constrained optimization to ensure that the probability of any given transduction is in $[0, 1]$, we find that unconstrained

[1]One possibility is to bucket words by their unique frequencies and approximate the full function by learning a separate coin flip parameter for each bucket. We took this approach in West et al. (2011) for a similar parametrization, but we found it to be suboptimal due to sparsity issues (the approximation learned was not at all smooth as one would expect from the underlying function).

optimization remains well-behaved. We present the function being maximized and its gradients with respect to each parameter for reference.

$$\begin{aligned}
\mathrm{p}(\boldsymbol{\gamma}|\alpha,\lambda) &= \left(\prod_i \left(1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}\right)^{\gamma_{i0}}\right)\left(\prod_i \left(c_i\lambda e^{-\operatorname{freq}(w_i)^\alpha}\right)^{\gamma_{i1}}\right) \\
\log\left(\mathrm{p}(\boldsymbol{\gamma}|\alpha,\lambda)\right) &= \left(\sum_i \gamma_{i0}\log\left(1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}\right)\right)+\left(\sum_i \gamma_{i1}\log c_i\right)+ \\
&\quad \left(\log\lambda\sum_i \gamma_{i1}\right)-\left(\sum_i \gamma_{i1}\operatorname{freq}(w_i)^\alpha\right) \\
\frac{\partial}{\partial\alpha} &= \left(\sum_i \frac{\gamma_{i0}\lambda e^{-\operatorname{freq}(w_i)^\alpha}\operatorname{freq}(w_i)^\alpha\log\left(\operatorname{freq}(w_i)\right)}{1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}}\right)- \\
&\quad \left(\sum_i \gamma_{i1}\operatorname{freq}(w_i)^\alpha\log\left(\operatorname{freq}(w_i)\right)\right) \\
\frac{\partial}{\partial\lambda} &= -\left(\sum_i \frac{\gamma_{i0}e^{-\operatorname{freq}(w_i)^\alpha}}{1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}}\right)+\left(\frac{1}{\lambda}\sum_i \gamma_{i1}\right),
\end{aligned}$$

where $c_i = |\{w_j | w_j \neq w_i \wedge w_j \in \operatorname{walk}(w_i)\}|$. Notice that c_i is independent of α and λ and therefore drops out of the maximization function.

4.2.4 Adding Two-Way Word Frequency Sensitivity

Motivation

For our final and most complex parametrization, which we will refer to as `rw_freq_both`, we extend the hypothesis behind `rw_freq_unobs` to include sensitivity to observed as well as unobserved word frequency. Specifically, we postulate that in addition to preferring to replace observed words with rare words, we should also place a non-uniform distribution over replacement arcs to prefer those with rarer observed word output, thereby imposing a penalty for replacing common observed words. We justify this in much the same way, namely with the assertion that language learners are likely to be most familiar with common words and therefore more adept at using them than their rarer counterparts.

Parametrization

In order to model this hypothesis, we build in a third parameter β, which we use as follows.

$$\begin{aligned} \mathrm{p}(w|w) &= 1 - \lambda e^{-\,\mathrm{freq}(w)^{\alpha}} \\ \mathrm{p}(w|w') &= \frac{\lambda e^{-\,\mathrm{freq}(w')^{\alpha}} e^{-\,\mathrm{freq}(w)^{\beta}}}{\sum_{\{w^*|w^* \neq w' \wedge w^* \in \mathrm{walk}(w')\}} e^{-\,\mathrm{freq}(w^*)^{\beta}}} \end{aligned}$$

When $\alpha > 0$ and $\beta > 0$, this parametrization will correctly impose a preference for using rare unobserved words as replacements and a penalty for replacing common observed words.

Training

For this parametrization, we have moved away from any reasonable notion of a coin flip. Instead, we have what amounts to a many-sided die, one per unobserved word, whose throwing distribution depends on the frequency of the unobserved word and that of all the words that it may walk to in our dictionary graph. Thus, we must in the E-step track an expected count statistic for each side of every attested die in order to update α, β, and λ in the M-step. We represent these statistics as before in a vector $\boldsymbol{\gamma}$ where γ_{i0} is the *no fix* count for word i, and γ_{ij} is the count of *replacements* from word i to word j.[2]

As in Section 4.2.3, we maximize the log probability of our sufficient statistics with respect to our parameters, but in contrast to the other random walk models, we find it necessary in this case to impose constraints on the optimization process. The actual constraint that we wish to enforce is $\forall w', 0 \leq \lambda e^{-\,\mathrm{freq}(w')^{\alpha}} \leq 1$, but we find it sufficient to use a set of linear inequality constraints instead. Specifically, we impose the box constraints $0 \leq \lambda \leq 1$, $-1 \leq \alpha \leq 1$, and $-1 \leq \beta \leq 1$ using the L-BFGS-B method (Byrd et al., 1995) implemented as part of `optim`. We present the function being maximized and its gradients with respect to each

[2]Naturally, our actual implementation deals with the sparseness of the $\boldsymbol{\gamma}$ vector accordingly.

parameter for reference.

$$
\begin{aligned}
\mathrm{p}(\boldsymbol{\gamma}|\alpha,\beta,\lambda) = & \left(\prod_i \left(1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}\right)^{\gamma_{i0}}\right) \\
& \left(\prod_{i,j}\left(\frac{\lambda e^{-\operatorname{freq}(w_i)^\alpha} e^{-\operatorname{freq}(w_j)^\beta}}{\sum_k e^{-\operatorname{freq}(w_k)^\beta}}\right)^{\gamma_{ij}}\right) \\
\log\left(\mathrm{p}(\boldsymbol{\gamma}|\alpha,\beta,\lambda)\right) = & \left(\sum_i \gamma_{i0}\log\left(1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}\right)\right) + \left(\log(\lambda)\sum_{i,j}\gamma_{ij}\right) - \\
& \left(\sum_i\left(\sum_j \gamma_{ij}\right)\operatorname{freq}(w_i)^\alpha\right) - \left(\sum_{i,j}\gamma_{ij}\operatorname{freq}(w_j)^\beta\right) - \\
& \left(\sum_i\left(\sum_j \gamma_{ij}\right)\log\left(\sum_k e^{-\operatorname{freq}(w_k)^\beta}\right)\right) \\
\frac{\partial}{\partial\alpha} = & \left(\sum_i \frac{\gamma_{i0}\lambda e^{-\operatorname{freq}(w_i)^\alpha}\operatorname{freq}(w_i)^\alpha\log\left(\operatorname{freq}(w_i)\right)}{1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}}\right) - \\
& \left(\sum_i\left(\sum_j \gamma_{ij}\right)\operatorname{freq}(w_i)^\alpha\log\left(\operatorname{freq}(w_i)\right)\right) \\
\frac{\partial}{\partial\beta} = & -\left(\sum_{i,j}\gamma_{ij}\operatorname{freq}(w_j)^\beta\log\left(\operatorname{freq}(w_j)\right)\right) + \\
& \left(\sum_i\left(\sum_j \gamma_{ij}\right)\frac{\sum_k e^{-\operatorname{freq}(w_k)^\beta}\operatorname{freq}(w_k)^\beta\log\left(\operatorname{freq}(w_k)\right)}{\sum_k e^{-\operatorname{freq}(w_k)^\beta}}\right) \\
\frac{\partial}{\partial\lambda} = & -\left(\sum_i \frac{\gamma_{i0}e^{-\operatorname{freq}(w_i)^\alpha}}{1-\lambda e^{-\operatorname{freq}(w_i)^\alpha}}\right) + \left(\frac{1}{\lambda}\sum_{i,j}\gamma_{ij}\right)
\end{aligned}
$$

where all w_k in the sums are in the set $\{w_k | w_k \neq w_i \wedge w_k \in \operatorname{walk}(w_i)\}$ for the appropriate w_i.

4.2.5 Constraining Random Walk Degree

We have thus far proceeded by describing the construction of an ideal noise model that completely implements the dictionary graph described previously. However, due to the size of the dictionary graph, such a model would be computationally prohibitive. After all preprocessing, the range of random walk candidates is 1683 with a mean of 47 and standard deviation of 82.2. To put these numbers in perspective, let us imagine that we are processing a three word sentence. Each word will generate an average of 47 candidate arcs plus an arc for itself, bringing our symbol count up to 144 from 3. From there, we must generate a language model. Under the bigram regime, this would meaning having an initial state plus one state per symbol and 144 arcs leaving each, bringing our language model edge count to over 20,000.[3] Recalling from Section 2.3 that the time complexity of the composition operation is $O(v_1 v_2 d_2(\log d_1 + m_1))$ when the first transducer is arc-sorted, the addition of each unique candidate word imposes a quadratic increase in computation time since v_1 and d_2 both increase linearly. Building a language model using the full dictionary graph over a mini-batch of only 100 sentences generates transducers in the gigabyte range and quickly becomes prohibitive for training over a reasonably-sized sample of example sentences. This issue is further exacerbated when composing the random walk model with other noise models that themselves generate many candidates for each observed word, e.g. the spelling correction model of Park and Levy (2011).

Thus, even with a conservative estimate of candidate overlap, modern machinery simply cannot handle the large amount of data that would need to be generated and processed for even a small batch of sentences. In order to combat this issue, we employ a simple heuristic that chooses those words with the highest unigram frequencies from the set of possible random walk candidates for each word.[4] In West et al. (2011), we reported on models using a cutoff of 5 and 10 candidates per word, but our experiments demonstrated little difference in final results. As such, we will use a cutoff of 10 candidate words for all experiments reported in this thesis.

[3]It may initially seem naive to generate the language model transducer with so many extraneous arcs. However, as the model framework is designed to handle arbitrary noise models in cascades, keeping track of the context states and arcs that are actually needed can become quite complex. On the other hand, doing so successfully has the potential to significantly speed up our model and is a viable subject for future study.

[4]In fact, an omitted portion of our dictionary preprocessing algorithm presorts the candidates for each word by unigram frequency

Chapter 5

Results and Analysis

In total, we ran eight experiments, one trained with batch EM and one with online EM for each noise model presented in Chapter 4. In each, we trained over a set of 10,000 sentences randomly selected from our training set and then decoded the 1,007 sentences from our evaluation set for judgment and analysis. We found that the `rw_unif`, `rw_freq_unobs`, and `rw_freq_both` models all converged to within a delta of 5% after roughly 10 iterations, although some continued oscillating well afterward. The `inf_pp` model took somewhat longer, and so we present the results of those two experiments after 50 iterations.

We will proceed with our results presentation in two stages. In the first, we will present the evolution of parameter values over time for our series of otherwise identical batch and online EM experiments. In the proceeding sections, we will present the `BLEU` and `METEOR` scores for each experiment, the results of our AMT judgment task on decoded sentences, and an accompanying analysis of individual examples and trends. Our results were somewhat disappointing across the board, and so as a sanity check and to test our common sense hypotheses about parameter values versus the learned values, we additionally ran a decoding test and judgment task on versions of each of the four models using initial (untrained) parameter values.

5.1 Batch v. Online EM

Liang and Klein (2009) report that online EM not only converges faster for many natural language processing tasks, but also has the advantage of finding better parameters upon convergence. In our experiments, we achieve similar, though not identical results. Beginning with the `inf_pp` experiments plotted in Figure 5.1, we see that online EM converges to a much smaller parameter value for λ, the probability of transducing from the present participle to infinitive verb form. In fact, both models find very small values for λ, which we interpret as the unfortunate consequence of a more general issue with Park and Levy (2011)'s model, namely that all else being equal, shorter sentences are always preferred.[1] The λ learned under both regimes is small enough that no present participle to infinitive corrections are made during decoding for the online model, and only a single such correction is made under the batch model. As such, the difference between online and batch EM in this case is not actually significant. The parameters learned while training `rw_unif` exhibit a similar lowering effect under online EM (see Figure 5.2), but we do find that the model converges faster in this case when compared to batch EM.

The remaining experiments reveal more interesting results. Looking now to Figures 5.3 and 5.4, we immediately notice that the learning process under online EM is not nearly as smooth as with its batch counterparts. This actually leads to *slower* convergence for the online versions of these models, but with the added assurance that training has explored a larger section of the parameter space before converging. The plot of α in Figure 5.4 epitomizes this behavior. However, the results are oddly inconsistent. Noting that a value of approximately $\alpha = 0.1$ (or -0.1) maximizes the variance in $e^{-\operatorname{freq}(w')^{\alpha}}$ along the word frequency axis, we see that online EM learns a weaker dependence on unobserved word frequency than batch EM in `rw_freq_unobs`, but in `rw_freq_both`, the α learned under online EM connotes a *stronger* dependence than that of batch EM. This is especially surprising in that both online and batch EM eventually converge to $\beta \approx 1$, a practical non-

[1]Since path probability is computed via the $\otimes$ operation under the probability semiring, the addition of any arc with a weight of less than one invariably decreases the probability of a path.

dependence on observed word frequency, and as such one would suppose that the other parameters should be very much the same as in `rw_freq_unobs`. We shall explore the results of using these parameters in the decoding process in the proceeding sections.

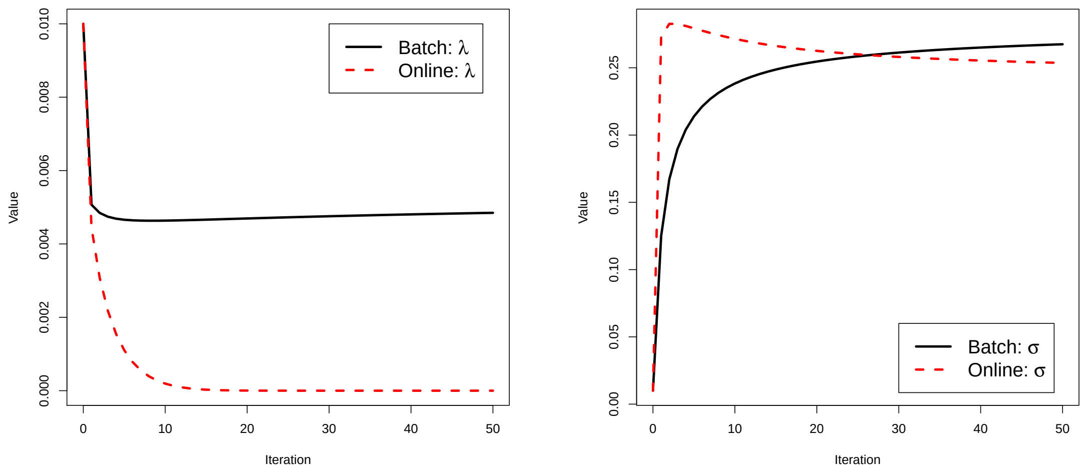

Figure 5.1: Plots of parameter values for the `inf_pp` model over time for 50 training iterations. λ is the present participle $\rightarrow$ infinitive replacement parameter, and σ represents the probability of transducing in the opposite direction.

5.2 Infinitive $\leftrightarrow$ Present Participle Model

Table 5.1: Final parameter values after 50 iterations for `inf_pp` where σ is the probability of correcting infinitive $\rightarrow$ present participle and λ is the opposite transduction probability.

	Initial Values	Batch EM	Online EM
σ	0.01	0.267492	0.253619
λ	0.01	0.004848	1.31538e-12

The initial and final parameter values for the `inf_pp` model are presented in Table 5.1, the `METEOR` and `BLEU` scores in Table 5.2, and the AMT judgment task

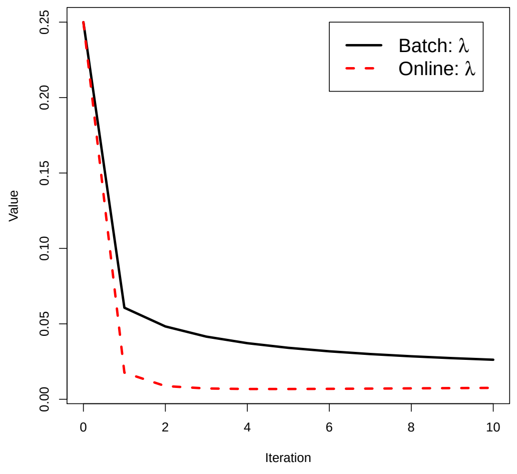

Figure 5.2: Plot of parameter values for the `rw_unif` model over time for 10 training iterations. λ is the uniform probability of word replacement in this model.

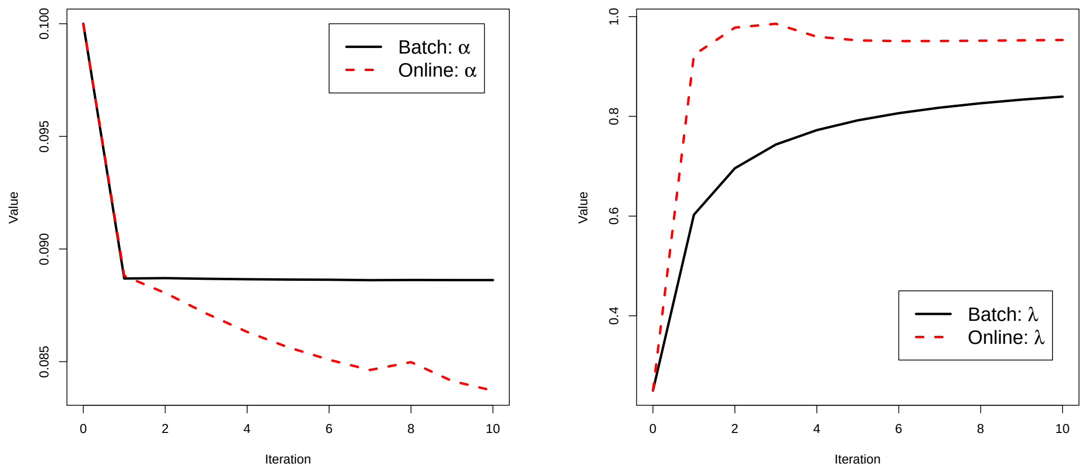

Figure 5.3: Plots of parameter values for the `rw_freq_unobs` model over time for 10 training iterations. λ is the base replacement probability in this model, and α represents the replacement variability related to unobserved word frequency.

results in Figure 5.5. The primary behavior that we observe is a strong preference for present participles in place of *to* infinitive forms, to the point where not a single correction is made in the opposite direction under the online model parameters.

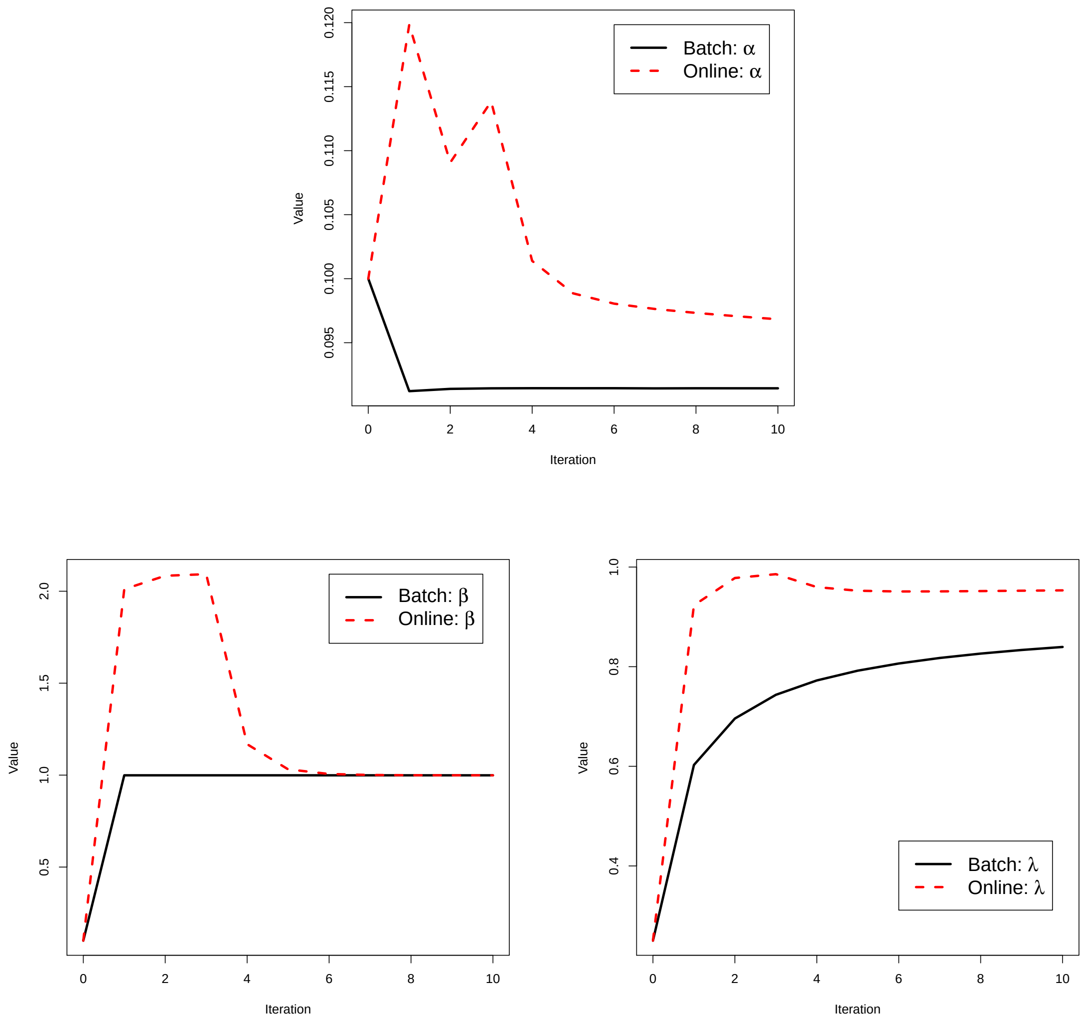

Figure 5.4: Plots of parameter values for the `rw_freq_both` model over time for 10 training iterations. λ is the base replacement probability in this model, α represents the replacement variability related to unobserved word frequency, and β represents the variability related to observed word frequency.

As mentioned in Section 5.1, this is a direct result of a more general systemic issue, namely that the framework of Park and Levy (2011) prefers shorter sentences. The problem persists even when decoding using initial parameter values, which we set to 0.01 for both σ and λ.

This is not to say that the model corrections are uniformly bad, of course. In

Table 5.2: METEOR and BLEU scores and number of sentences corrected (out of 1007) for online and batch variations of the inf_pp model. Better/Worse indicates the number of corrected sentences scored higher/lower than the associated ESL sentence. Note that sentences having identical scores are not listed in these counts.

	METEOR (Better/Worse)	BLEU (Better/Worse)	Total Num Corrected
ESL baseline	0.821	0.715634	–
inf_pp init vals	0.820577 (0/5)	0.715142 (0/5)	5
inf_pp batch EM	0.812668 (9/78)	0.705197 (8/78)	87
inf_pp online EM	0.812813 (9/76)	0.705465 (8/76)	85

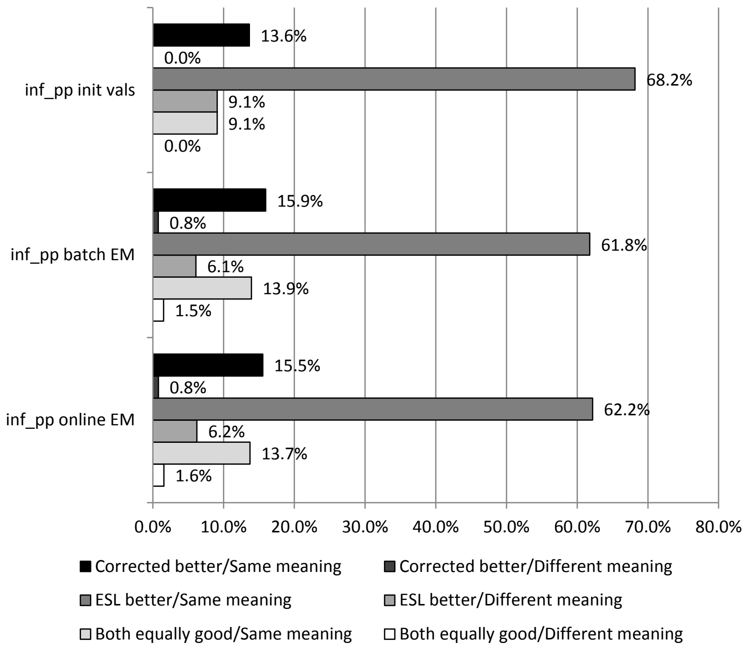

Figure 5.5: Human judgments of inf_pp-corrected sentences gathered using AMT. The items listed in the legend are answers to the questions *Between the [original (ESL) and corrected] sentences, which is more correct? / Is the meaning of the corrected sentence significantly different from that of the ESL sentence?* See Section 3.5 for methodological details.

fact, the METEOR and BLEU scores are only loosely correlated with actual correction quality, a trend that we will observe throughout this chapter. A large number of the changes are relatively harmless, e.g. ***to take*** *one of my cousin jennifer as an* *example , ...* → ***taking*** *one of my cousin jennifer as an example , ...*, where there exists an equally valid interpretation under which each form is correct. Others are actually rather good, e.g. ***to be*** *a normal entertainer is not so hard .* → ***being*** *a normal entertainer is not so hard .* In still other cases, the correction simply fails to unravel an already muddled sentence, e.g. *this is because not everyone has ability for leadership and they must* ***to have*** *responsibility about negative accident .* → *this is because not everyone has ability for leadership and they must* ***having*** *responsibility about negative accident .* But then there are also a good number of corrections that simply should not have been corrected, e.g. *first , its aspect of violation may influence people* ***to do*** *violent act .* → *first , its aspect of violation may influence people* ***doing*** *violent act .* The language model may prefer *doing* in the immediate context, but making such a replacement changes the sentence meaning and in some cases even renders a correct sentence ungrammatical. The judgment task results confirm that this poor correction behavior is the norm, and although training appears to have a minor positive effect on overall quality, upon closer inspection, the initial value corrections are just a subset of the trained corrections, only one of which is really very good.

Interestingly, an earlier, under-trained version of the model with $\lambda \approx 0.020$ and $\sigma \approx 0.038$ produced almost uniformly good or harmless corrections over our development set, e.g. *to illustration , when we learn* ***playing*** *tennis , ...* → *to illustration , when we learn* ***to play*** *tennis , ...* and *another my opinion is that becoming easier* ***making*** *food stimulus my desire for making more specific food .* → *another my opinion is that becoming easier* ***to make*** *food stimulus my desire for making more specific food .* We are not well-placed to say why such a disparity between development and evaluation set results should occur, but the fact remains that *to infinitive* ↔ *present participle* mistakes do occur in both directions in our dataset and that our `inf_pp` model is capable of intelligently correcting them in at least some cases. There is thus a strong motivation in future work to develop a method of normalizing for sentence length analogous to the word insertion penalty common in speech processing (Takeda et al., 1998) such that trivial MLE solutions such as our extremely low λ values are not learned during the optimization process.

5.3 Random Walk Models

Table 5.3: Final parameter values after 10 iterations for all random walk models. See Section 4.2 for parametrization details.

	Initial Values	Batch EM	Online EM
rw_unif λ	0.25	0.026210	0.0075548
rw_freq_unobs α	0.1	0.088620	0.0837149
rw_freq_unobs λ	0.25	0.0354126	0.0133420
rw_freq_both α	0.1	0.0914392	0.0968309
rw_freq_both β	0.1	0.999	0.999043
rw_freq_both λ	0.25	0.839603	0.953206

Table 5.4: METEOR and BLEU scores and number of sentences corrected (out of 1007) for online and batch variations of all random walk models. Better/Worse indicates the number of corrected sentences scored higher/lower than the associated ESL sentence. Note that sentences having identical scores are not listed in these counts.

	METEOR (Better/Worse)	BLEU (Better/Worse)	Total Num Corrected
ESL baseline	0.821	0.715634	–
rw_unif init vals	0.81494 (1/73)	0.706141 (1/76)	98
rw_unif batch EM	0.819977 (1/12)	0.713769 (1/13)	25
rw_unif online EM	0.820728 (0/3)	0.715414 (0/3)	9
rw_freq_unobs init vals	0.816419 (1/55)	0.708953 (1/57)	79
rw_freq_unobs batch EM	0.820127 (1/11)	0.714081 (1/12)	24
rw_freq_unobs online EM	0.82054 (0/5)	0.714737 (0/6)	11
rw_freq_both init vals	0.816935 (1/49)	0.709692 (1/51)	71
rw_freq_both batch EM	0.812502 (1/104)	0.702765 (1/109)	132
rw_freq_both online EM	0.812502 (1/104)	0.702765 (1/109)	132

The initial and final parameter values for all random walk model experiments are presented in Table 5.3, the METEOR and BLEU scores in Table 5.4, and the AMT judgment task results in Figure 5.6. We will begin by analyzing the trained model results in aggregate and then proceed to a discussion of model disparities, including the effect of using initial parameter values for decoding.

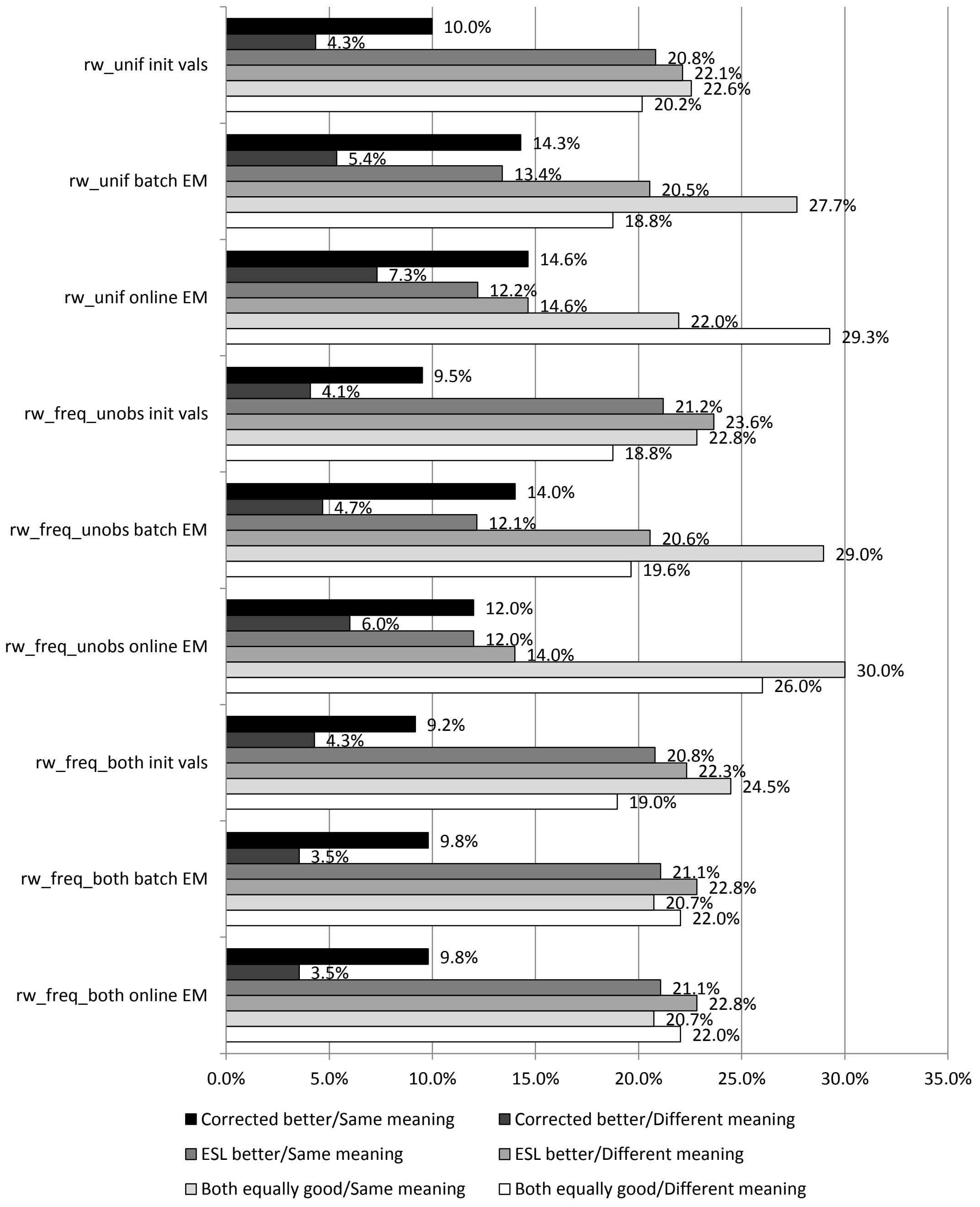

Figure 5.6: Human judgments of random walk model-corrected sentences gathered using AMT. The items listed in the legend are answers to the questions *Between the [original (ESL) and corrected] sentences, which is more correct? / Is the meaning of the corrected sentence significantly different from that of the ESL sentence?* See Section 3.5 for methodological details.

At first glance, the experimental results are less than satisfactory. However, as with the `inf_pp` model results, `METEOR` and `BLEU` do not tell the whole story. At a high level, these metrics work by computing the level of agreement, e.g. unigram and bigram precision, between the sentence being evaluated and a pool of "correct" sentences (Lavie and Agarwal, 2007; Papineni et al., 2002). When the correct sentences agree strongly with each other, the evaluated sentence is heavily penalized for any departures from the correct sentence pool. This sort of penalization can occur even when the model-corrected sentence is a perfectly valid correction that just had the misfortune of choosing a different replacement word than the majority of the human workers. For example, one ESL sentence in our evaluation set reads, *progress of medical science helps* ***human*** *live longer.* All of our (trained) models correct this to *progress of medical science helps* ***people*** *live longer*, but none of the workers correct to "people," instead opting for "humans." This issue is exacerbated by the fact that AMT workers were instructed to change each ESL sentence as little as possible, which helps their consistency but hurts these particular models' evaluation scores.[2]

With the exception of some mostly harmless but ultimately useless exchanges, e.g. changing "reduce mistakes" to "reduce errors," the models actually do fairly well when they correct ungrammatical words and phrases. As we alluded to in Chapter 1, all of our models correct the sentence *to begin with, i'd rather not room with someone who is a* ***entire*** *stranger to me* from our development set to *to begin with, i'd rather not room with someone who is a* ***complete*** *stranger to me.* But only 2 out of 5 human workers make this correction, 2 retain "entire," and 1 removes it altogether. As another example, all model variations correct *however, depending* ***merely*** *on luck is very dangerous* from our evaluation set to *however, depending* ***solely*** *on luck is very dangerous.* However, only 1 worker corrects "merely" to "solely," with the others either preferring to retain "merely" or leaving it out entirely.

None of this is to say that the models suffer only from an unfortunate

[2]To be perfectly clear, we refer here to the AMT workers tasked with creating our pool of correct sentences, not those who participated in the judgment task.

difference in correction bias relative to the workers, or even that the models make good corrections a majority of the time. In fact, they make a range of false-positive corrections as well.[3] These seem to fall into three major categories: slight preferences for similar words that don't fit in the overall context of the sentence or change its meaning in an undesired way, e.g. changing "roommate" to "lodger" in *you and your **roommate** must devide [sic] the housework*, strong preferences for very common words in the local context that render the corrected sentence ungrammatical, e.g. changing "compose" to "take" in *first, during childhood years, we **compose** our personality*, and misinterpretations of ambiguous parts of speech that cause nouns to be replaced with verbs, etc., e.g. changing "circumstance" to "go" in *... that help you look abound your **circumstance** and find out ...*.

Many of these issues can be blamed at least partially on the myopia of the language model, which, for example, vastly prefers "go and find" to "circumstance and find." However, they can also be attributed to the motivational intuition for `rw_freq_unobs`, which states that we should avoid replacing observed words with common alternatives. While Table 5.3 does demonstrate that `rw_freq_unobs` and `rw_freq_both` learn to prefer rarer alternative candidates (see Figure 5.7 to get a sense of exactly how much), overly common words are still preferred to a fault in many corrections. This can be traced to the truncation policy detailed in section 4.2.5, which selects only the highest frequency words from an over-sized set of random walk candidates. While it is unclear how to intelligently select a good candidate set of manageable size, a policy that butts heads with our intuition about which words we should be correcting is clearly not the right one.

5.3.1 Uniform Replacement Model

Despite its simplicity, `rw_unif` performs relatively well. The initial value appears to be much too high to be at all discriminating in making corrections,

[3]Although Type I errors are of course undesirable, Gamon et al. (2009) suggest that learners are able to effectively distinguish between good and bad corrections when presented with possible error locations and scored alternatives. Such an interactive system is beyond the scope of this thesis but nonetheless feasible without significant model modification.

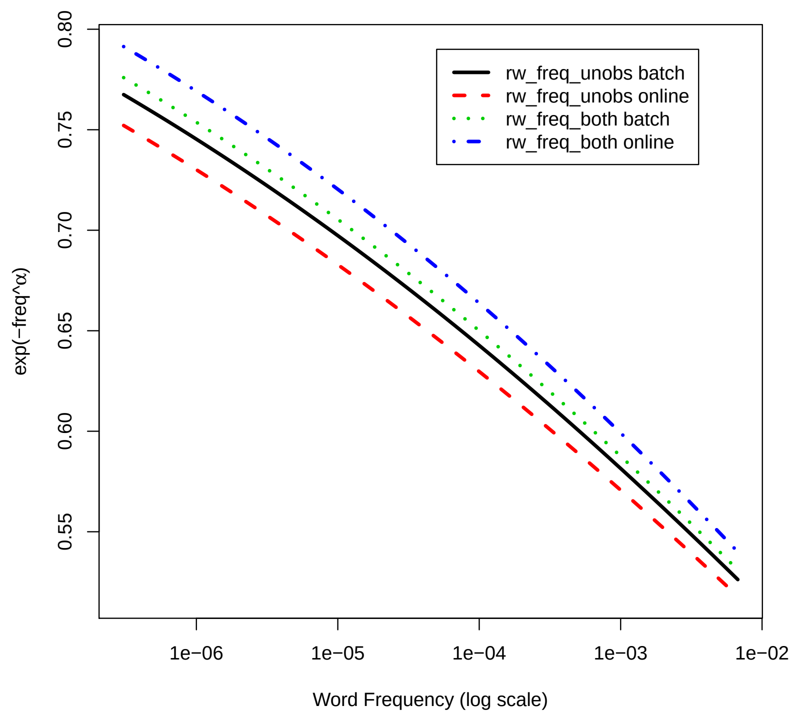

Figure 5.7: Plot of $e^{\text{freq}(w')^{\alpha}}$ for a representative range of word frequencies using the values of α learned for `rw_freq_unobs` and `rw_freq_both` under batch and online EM training regimes. Note that the range of transduction variability due to unobserved word frequency is roughly 0.25.

but the trained values are low enough that they achieve some measure of success according to our AMT judgment task. The worker judgments expose one particularly interesting finding: When the corrected sentence is judged to be at least as grammatical as the ESL sentence, it also tends to preserve the ESL sentence's meaning. However, when the ESL sentence is judged more correct, the meaning preservation trend is reversed. This observation leads us to believe that incorporating some measure of semantic distance into our random walk function f might prove effective. We note that this trend breaks down on the ESL-more-correct side when we decode with initial values, but this is to be expected.

5.3.2 One-Way Word Frequency Sensitivity Model

As in the AMT results for `rw_unif`, we observe a strong correlation between sentence quality judgment and meaning preservation. Additionally, the value of α learned in `rw_freq_unobs` is nearly the strongest dependence on unobserved word frequency possible under our model. Despite this, however, the `BLEU`, `METEOR`, and AMT results are not much different from the base model, a flaw which we trace as before to the truncation policy detailed in Section 4.2.5. There might be some further headway to be made with a function capable of describing a stronger or otherwise differently shaped frequency dependence, e.g. the logit function.

5.3.3 Two-Way Word Frequency Sensitivity Model

Despite our high hopes, the `rw_freq_both` model fails rather catastrophically relative to the other models. This is especially obvious when one considers the data from Figure 5.6, in particular the way in which the sentence quality judgment by meaning preservation correlation breaks down. The α value learned is very similar to that of `rw_freq_unobs`, but the model does not learn any significant dependence on observed word frequency with β, essentially reducing it to its one-way word frequency counterpart. Rather than learning a similar λ value to `rw_freq_unobs` as one might expect, however, `rw_freq_both` learns, depending on $\text{freq}(w')$, to reserve nearly half of its transduction probability mass for corrections. The results when using this extremely high value of λ are commensurately unstable.

Our best explanation for this instability lies in our use of approximate linear box constraints to bound what is actually a more complex, non-linearly constrained problem. Although L-BFGS-B does consistently converge within our constraints, unconstrained quasi-gradient and global methods find very different values at each step. Thus, while we are not convinced that our model is incorrect in its formulation, our M-step is almost certainly ill-posed. We will need in future work to either engage in a non-linear programming exercise or find a more satisfactory way to linearly constrain our problem.

Chapter 6

Conclusion and Future Work

We have presented two novel noisy channel models for correcting verb infinitive/past participle confusion and word choice production errors and a demonstration of the application of online expectation maximization to the model of Park and Levy (2011). Although our experimental results are mixed, we believe that our random walk model especially constitutes an interesting and potentially very fruitful approach to ESL grammar correction.

There are a number of opportunities for improvement available. Using a richer language model, such as a PCFG, would undoubtedly improve our results. Although our training process would need to be altered quite significantly to use a Type-2 grammar, PCFGs are closed under intersection with regular languages (Bar-Hillel, 1964), and as such might be useful for the decoding process if not more directly during training. As noted previously, a higher-order n-gram model could also come into the range of feasibility should we develop a way to implement our language model as a wFST in a less naive and more space-conscious fashion. Noting that ESL errors tend to occur in groups within sentences and are often interdependent, the addition of other noise models, such as those detailed in Park and Levy (2011), would further improve things by allowing the language model to consider a wider range of corrected contexts around each word, as would a method of normalizing transduction probability with respect to path length.

Our random walk model itself could also be improved by incorporating some notion of semantic difference between observed and unobserved words, or by learning separate parameters for different word classes. Moreover, a more disciplined application of convex optimization to our `rw_freq_both` model might yield more promising results. Somewhat counter-intuitively, a structured reduction of dictionary richness could also yield better results by limiting the breadth of random walk candidates. Finally, a more intelligent, non-frequency-based heuristic for truncating large sets of random walk candidates would likely foster improvement.

Appendix A

Stop Word Filter

In preprocessing our English-Korean dictionary, we removed a list of stop words gathered from `http://www.textfixer.com/resources/common-english-words.txt`. This list is fully reproduced in Table A.1.

Table A.1: A list of stopwords used in filtering random walk candidates.

a	able	about	across	after	all	almost	also
am	among	an	and	any	are	as	at
be	because	been	but	by	can	cannot	could
dear	did	do	does	either	else	ever	every
for	from	get	got	had	has	have	he
her	hers	him	his	how	however	i	if
in	into	is	it	its	just	least	let
like	likely	may	me	might	most	must	my
neither	no	nor	not	of	off	often	on
only	or	other	our	own	rather	said	say
says	she	should	since	so	some	than	that
the	their	them	then	there	these	they	this
tis	to	too	twas	us	wants	was	we
were	what	when	where	which	while	who	whom
why	will	with	would	yet	you	your	

Appendix B

Amazon Mechanical Turk Judgment Task

The following is the full text of the Amazon Mechanical Turk judgment task outlined in Section 3.5. Note that there are ten unique sentence pairs presented per task.

Please compare the grammaticality/fluency of the following sentence pairs

Task Description

IMPORTANT: As this task involves judging English sentence fluency, we request that **only native English speakers** participate. **Please DO NOT attempt this task if you are not a native English speaker.**

The following are a list of closely-related sentence pairs. The first sentence in each pair, labeled "ESL Sentence," was written by a non-native English speaker, and the second, labeled "Corrected Sentence," is a corrected version of the ESL sentence. One or both may contain errors (spelling, word choice, etc.). Your task is to judge whether the corrected sentence improves the grammaticality and/or fluency of the ESL sentence without changing the ESL sentence's basic meaning.

More specifically, there are two parts to each question:

1. In the first part, you will be asked to judge whether the corrected sentence improves the grammaticality and/or fluency of the ESL sentence. If you believe that the corrected sentence is an improvement over the ESL sentence, you should select "Corrected sentence is more correct." If the ESL sentence is more grammatical/fluent, you should select "ESL sentence is more correct." If you believe that both sentences are equally correct, you should select "Both are equally correct," and if the sentences are identical, that is, if every word in the corrected sentence is the same as the ESL sentence, you should select "The sentences are identical."

2. In the second part, you will be asked to judge whether the basic meaning of the two sentences is different, regardless of correctness. In response to the question, "Is the meaning of the corrected sentence significantly different from that of the ESL sentence," if you believe that the meaning of the two sentences is significantly different, please select "Yes, the two sentences do not mean the same thing." If the meaning of the two sentences is roughly the same, please select "No, the two sentences have roughly the same meaning."

Please ignore incorrect capitalization and any spaces between words and punctuation.

Examples

1. - *ESL Sentence:* however , i am truly sure that there are a lot of advantages to building restaurant .
   - *Corrected Sentence:* however , i am very sure that there are a lot of advantages to building restaurant .
   - *Answer, Part 1:* Corrected sentence is more correct
   - *Answer, Part 2:* No, the two sentences have roughly the same meaning

2.
   - *ESL Sentence:* one house are supposed to park one car but some people have cars more than one .
   - *Corrected Sentence:* one house are supposed to take one car but some people have cars more than one .
   - *Answer, Part 1:* ESL sentence is more correct
   - *Answer, Part 2:* Yes, the two sentences do not mean the same thing

3.
   - *ESL Sentence:* for example , when i lived in dormitory with a student last semester , i had so much trouble with her from cleaning the room to locking the door .
   - *Corrected Sentence:* for example , when i lived in house with a student last quarter , i had so much trouble with her from cleaning the room to locking the door .
   - *Answer, Part 1:* Both are equally correct
   - *Answer, Part 2:* Yes, the two sentences do not mean the same thing

4.
   - *ESL Sentence:* it could interesting to spend time with a large number of friends .
   - *Corrected Sentence:* it could interesting to spend time with a large number of friends .
   - *Answer, Part 1:* The sentences are identical
   - *Answer, Part 2:* No, the two sentences have roughly the same meaning

Questions

1. PLEASE COMPARE THE FOLLOWING TWO SENTENCES:
   ESL Sentence: ⟨SENTENCE 1 ESL⟩
   Corrected Sentence: ⟨SENTENCE 1 CORRECTED⟩
   Part 1: Between the two sentences listed above, which is more correct?

   - ESL sentence is more correct

- Corrected sentence is more correct
- Both are equally correct
- The sentences are identical

Part 2: Is the meaning of the corrected sentence significantly different from that of the ESL sentence?

- Yes, the two sentences do not mean the same thing
- No, the two sentences have roughly the same meaning

2. ...

Bibliography

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). Openfst: a general and efficient weighted finite-state transducer library. In *Proceedings of the 12th international conference on Implementation and application of automata*, CIAA'07, pages 11–23, Berlin, Heidelberg. Springer-Verlag.

Baayen, H. R., Piepenbrock, R., and Gulikers, L. (1995). *The CELEX Lexical Database. Release 2 (CD-ROM).* Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania.

Bar-Hillel, Y. (1964). *Language and information.* Addison-Wesley, Reading, Mass.

Brockett, C., Dolan, W. B., and Gamon, M. (2006). Correcting esl errors using phrasal smt techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 249–256, Stroudsburg, PA, USA. Association for Computational Linguistics.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19:263–311.

Burnard, L. (1995). *Users Reference Guide British National Corpus Version 1.0.* Oxford University Computing Services, UK.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16:1190–1208.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38.

Désilets, A. and Hermet, M. (2009). Using automatic roundtrip translation to repair general errors in second language writing. pages 198–206. MT Summit XII.

Dreyer, M., Smith, J. R., and Eisner, J. (2008). Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1080–1089, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eisner, J. (2002). Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13:317–322.

Gamon, M., Leacock, C., Brockett, C., Dolan, W. B., Gao, J., Belenko, D., and Klementiev, A. (2009). Using statistical techniques and web search to correct esl errors. *CALICO Journal*, 26:491–511.

Goodman, J. (1999). Semiring parsing. *Comput. Linguist.*, 25:573–605.

Kok, S. and Brockett, C. (2010). Hitting the right paraphrases in good time. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 145–153, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lavie, A. and Agarwal, A. (2007). Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lee, J. and Seneff, S. (2006). Automatic grammar correction for second-language learners. ICSLP.

Liang, P. and Klein, D. (2009). Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mohri, M. (2001). Generic ϵ-removal algorithm for weighted automata. In Yu, S. and Paun, A., editors, *Implementation and Application of Automata*, volume 2088 of *Lecture Notes in Computer Science*, pages 230–242. Springer Berlin / Heidelberg.

Mohri, M. (2004). Weighted finite-state transducer algorithms: An overview. In Martín-Vide, C., Mitrana, V., and Paun, G., editors, *Formal Languages and Applications*, volume 148. Springer, Berlin.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Park, Y. A. and Levy, R. (2011). Automated whole sentence grammar correction using a noisy channel model. In *Proceedings of the 49th annual meeting on Asso-*

ciation for Computational Linguistics, ACL '11. Association for Computational Linguistics. In Press.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Sato, M.-A. and Ishii, S. (2000). On-line em algorithm for the normalized gaussian network. *Neural Comput.*, 12:407–432.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27:623–656.

Takeda, K., Ogawa, A., and Itakura, F. (1998). Estimating entropy of language from optimal word insertion penalty. In *Proceedings of Int. Conf. Spoken Language Processing*. Citeseer.

West, R., Park, Y. A., and Levy, R. (2011). Bilingual random walk models for automated grammar correction of esl author-produced text. In *Proceedings of the NAACL HLT 2011 Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, Stroudsburg, PA, USA. Association for Computational Linguistics. In Press.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*.