Abstract of "Computationally Connecting Language Transfer and Second Language Education" by Ben Swanson, Ph.D., Brown University, May 2014.

We explore the role of software assistance in native language targeted second language education. Beginning with the classification task of determining an author's native language from second language text, we demonstrate the use of Tree Substitution Grammar fragments as an effectively discriminative class of features. We contrast the use of several syntactic analyses of second language text as well as different methods for feature selection through grammar induction. We move on to investigate the data driven formulation of hypotheses for syntactic language transfer, which refers to the preferential use of second language syntax that mirrors an author's native language. Our methodology produces a ranked and filtered list of hypotheses that provides compelling evidence for several examples of language transfer, and is easily augmented as more data becomes available. We conclude with a novel system for language generation in educational applications that incorporates the inherent vocabulary based constraints of the domain. While these constraints are easily handled with rejection sampling, this becomes inefficient as the amount of training data increases. To combat this, we show sampling algorithms for context free languages that avoid rejection, sampling directly from the acceptable outputs with tight approximation. Our work facilitates the construction of systems to both enhance the quality of second language education through awareness of students' native languages and reduce the effort required to create language education exercises.

Computationally Connecting Language Transfer and Second Language Education

by

Ben Swanson

B.A., Computer Science, Pomona College, 2006

B.A., Physics, Pomona College, 2006

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Computer Science at Brown University

Providence, Rhode Island

May 2014

This dissertation by Ben Swanson is accepted in its present form
by the Computer Science as satisfying the dissertation
requirement for the degree of Doctor of Philosophy.

Date _____ _____

Eugene Charniak, Co-Chair

Recommended to the Graduate Council

Date _____ _____

Erik Sudderth, Co-Chair

Date _____ _____

Uriel Cohen-Priva, Reader

Approved by the Graduate Council

Date _____ _____

Professor Peter M. Weber
Dean of the Graduate School

Acknowledgements

This work was made possible by the generous time of my advisors and collaborators over the years. First and foremost thank you to Eugene for his wisdom, experience, and patience. With his office door always open, Eugene has been a source of guidance on everything from encyclopedic knowledge of the field's past, present, and future, to experimental design and the dirty implementation details of taking something from a whiteboard to code. He is also an inspirational figure when considering how a balance between a solid intellect and a humorous attitude produces a researcher who is a true pleasure to work with.

My other readers, Erik Sudderth and Uriel Cohen-Priva, have also been invaluable in the development of this work. I would never have begun to understand the math under the hood without Erik's explanations and deep expertise in non-parametric Bayesian methods. Uriel's dual knowledge of pure linguistics and mathematical methods were essential in guiding and formulating the arguments for language transfer, the true goal of this entire line of research.

As a member of the CS Department at Brown I had the opportunity to work with several brilliant people who contributed either directly to the work or indirectly through a community of eager learners. The members of BLLIP were a constant source of advice; Rebecca Mason, Chris Tanner, and DK Choe. I am especially grateful for the opportunity to have worked with the incomperable Micha Elser whose breadth and depth of knowledge I have missed since his graduation. Also many thanks

to the Sudd's Buds; Daeil Kim, Soumya Ghosh, Jason Pacheco, and Michael Hughes for your friendship and level 9000 machine learning skills.

Thank you to my father Paul, my brother Carl, my godparents Auntie Meesh and Uncle Tom, and my Uncle Pete for their lifelong support and love. And while she never was able to see me graduate from any of the schools I have attended, it was all done in a way for my mother Patty, whose memory is my deepest motivation.

Finally, the defining aspect of my time spent with this work was that I shared it with Elif Yamangil, who is a co-author on nearly every publication that makes it up. She was the best friend I have ever known, and a brilliant person. Wherever her life leads, may the rain run off her shoulders.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As the world becomes more and more technologically fluent, there is a growing opportunity for occupation targeted software. Such software typically automates tasks that are repetitive or unnecessarily time consuming, such as simple error-free tabulation and calculation with spreadsheet software. In other cases, the use of computation opens up completely new aspects of a profession that would be unfeasible otherwise, with examples like data-journalism or high frequency stock trading.

The goal of this thesis is to develop the interaction between software and the profession of second language education. While second language education is already no stranger to software assistance, we hypothesize its role in two deeply connected and under-explored areas. The first is native language targeted instruction, taking into account the norms of a student's native language when providing second language instruction. The second is automatic generation of language education exercises or exams while obeying the constraints of the students' vocabulary.

To see our vision by example, consider a semi-fictional character named Ross, a 22 year old young man living in the city of Boston. Ross has recently earned his undergraduate degree in Philosophy and is saving money to take a long international trip, for which purpose he gets a job as a teacher at one of the many English as a

Second Language (ESL) schools in Boston. As a native English speaker with a degree from Dartmouth this is not a difficult job to get, but Ross's true interests lie in athletic running and his plans for law school in the future. As such, he rarely musters the motivation or ability to prepare classroom activities or assignments beyond those contained in the text book. He is assigned two daily classes, an afternoon class of students from Korea and an evening class for adults from Brazil. Using a system with the components that we develop here, Ross could be equipped with extra in-class exercises and homework, with minimal effort on his part outside the classroom. Moreover, the exercises for each class could be made distinct, using sentences that highlight grammatical structure tailored directly to the students' different native language backgrounds. The students get a better education, the school is able to hire unexperienced teachers, and Ross is able to devote more time to his interests without sacrificing the quality of his work.

Native language targeted instruction has been extensively shown to be beneficial to student performance in language education. An example of recent work is Laufer and Girsai (2008), who conducted a comparative study of three modes of instruction for English as a second language students whose native language is Hebrew. The first instruction technique group engaged in reading comprehension exercises. The second group conducted classroom activities directly focused on specific vocabulary and phrases in the text, but with only English examples. The third group was given instruction that explicitly discussed differences between phrases found in the English text (e.g. "hit the headlines") with their Hebrew equivalents ("break into headlines"). Finally the three groups were evaluated on their ability to translate relevant phrases to and from English. While it is not surprising that the reading comprehension technique was least effective, the significant difference in exam scores between the second and third groups is an excellent example of the real potential of native language targeted

instruction.

Software is already a common source of language education exercises, both as a supplement to traditional course material and as fully automatic courses, such as Rosetta Stone and DuoLingo. Existing examples are often some computerized version of flash cards, simple questions with a single right answer. Most systems take advantage of the benefits of automatic student profiles, recycling questions that the student has shown difficulty with and removing questions the student regularly answers correctly. This also allows aggregation of profiles across a classroom, letting the teacher compose lesson plans in reaction to the needs of the class. There is also an active academic community devoted to the construction and evaluation of this sort of software, with several yearly conferences and journals that use the name Computationally Assisted Language Learning, or CALL. The results of CALL research leave little reason for language education not to be based in a software platform, besides lack of computational resources or existence of the proper software.

We facilitate these educational techniques in the methods and software presented here, with the automation of two key components as our main contributions. The first component is the data driven identification of linguistic structures with connection to particular L1 backgrounds, providing content for instruction. While there are many possibilities here, we focus on language transfer, in which a student chooses some mode of second language expression that is familiar to them in their native tongue (Lado (1957)). This has been studied extensively by the linguistics community, with primary attention given to the use of direct translations of words or collocations. We investigate syntactic language transfer behavior, using rich representations that not only captures language transfer in an interpretable form but can also be used in familiar generative language models.

The second component of the education process that we seek to automate is

the construction of classroom exercises, which at best requires specialized training in second language education and at least a significant time commitment on the part of the instructor. We investigate systems that learn generative models of text from large amounts of data and can effectively generalize to novel correct sentences. While generative language models are nothing new, our education setting of language education provides an inherent vocabulary constraint that has not been addressed in previous work. We show how to efficiently generate text under the constraint that all words are in a given vocabulary, and also allow specification of a target word or structure to be featured in the sentence.

This thesis is presented in three main sections. The first is concerned with the task of Native Language Identification (NLI), a well studied classification problem in which an author's native language is predicted from second language text. While not essential to our primary goal, this puts our work in the context of a the current research community as NLI is frequently motivated by its ability to detect language transfer phenomena. Also, unlike the pure proposal of language transfer hypotheses which has no labeled data for evaluation, NLI is a straightforward classification task, allowing comparison between different feature sets and representations of the data. The real purpose of this digression in the larger context of this work is to provide evidence for our extensive use of Tree Substitution Grammar (TSG) fragments as a representation of the language transfer signal in later sections, and to provide their necessary background information.

In the second section we address the data driven construction of language transfer hypotheses, both from monoligual second language data and the combination of both second and native language data. We argue that while they are deeply connected, the use of classification algorithms and their common notions of relevancy are in fact inappropriate for our purposes. In our monolingual experiments, we propose several

intuitive measures of what is actually desired, and find automatic metrics that better mirror these properties. Using these efficient metrics, we compose a fully universal representation of text across three native languages (German, French, and Spanish) and the common second language of English, and produce a ranked list of language transfer hypotheses. The top ranked items, discussed at length in this document, are compelling cases of language transfer, and our methods are easily extended to larger amounts of data that are rapidly becoming available.

The third section is concerned with the generation of educational exercises. While Natural Language Generation is most commonly associated with communicative goals, or "what to say", we focus on the constraints of word inclusion that are implicit in language education. The first constraint is vocabulary based, ensuring that the output contains only words from a fixed user specified list. We then extend this constraint with a generation system that ensures a particular word appears in the output sentence. Our intended application would be to aid a teacher in composition of materials such as "translate this sentence" test questions. Our system can provide a teacher in such a situation with several candidate sentences that feature any given vocabulary word from the current section of their syllabus, containing only words that the teacher can be sure the students have covered and can be expected to translate. Our basic approach is that of context free language generation, which can handle both constraints through rejection sampling. For a highly accurate system with wide coverage, we propose that a large amount of context is necessary, which in turn requires large amounts of data to avoid sparsity in estimates of probabilistic models. With large enough training data sets, rejection sampling becomes unacceptably slow, and our primary contributions in this section are alternative sampling algorithms that closely approximate the behavior of the theoretically sound but inefficient rejection sampler solutions.

We take the reproducibility of our work as a high priority, releasing the relevant code for each component of this thesis. For our NLI experiments our code can be used for the classification of arbitrary labeled text data using Tree Substitution Grammar rules as features[1]. This code handles parsing of the data, TSG induction, and various forms of evaluation including cross validation and an API for prediction in downstream applications. For data driven language transfer hypothesis formulation, our code performs the full pipeline of operations, producing a LaTeXdocument containing the full ranked list along with the necessary empirical information to make a full assessment of each item[2]. The TSG induction component can be downloaded separately, and has several capabilities not featured in this work, such as semi-supervised learning[3]. We also release our ranked list of language transfer hypotheses based on the data available at this time[4]. Our generation system takes raw text as input and produces serialized models with an API for generation under user provided vocabulary constraints.[5] This software is written in Scala, making its API directly available in Java, with an additional pure Javascript frontend for the generation package to facilitate its use in web based educational applications.

[1] www.cs.brown.edu/people/chonger/fragalyzer.html
[2] www.cs.brown.edu/people/chonger/LTAnalyzer.html
[3] www.cs.brown.edu/people/chonger/enbuske.html
[4] bllip.cs.brown.edu/download/interlanguage_corpus.pdf
[5] www.cs.brown.edu/people/chonger/babbler.html

Chapter 2

Native Language Identification

The corpus linguistics approach to formulating language transfer hypotheses is often (Jarvis et al. (2012), Tetreault et al. (2013)) approached via the related task of Native Language Identification (NLI), the straightforward multi-label classification task of predicting an author's native language (L1) from second language (L2) text. In this chapter we discuss NLI independent of our larger goal of L1 targeted language education, as we will argue in later sections that in fact such NLI based techniques are not ideal. Our motivation for this initial digression is primarily to demonstrate Tree Substitution Grammar (TSG) fragments as a class of features that effectively captures the signal that disambiguates L2 text by L1, and to contrast the options available when using TSG fragments as a feature representation of text.

In our experiments we first contrast two popular TSG induction methods, the first using a Bayesian nonparametric model (Cohn and Blunsom (2010a)), and the second inspired by tree kernel methods (Sangati and Zuidema (2011)). Next, we fix our induction algorithm and investigate the use of various syntactic analyses of the L2 text including dependency syntax, unsupervised latent symbol classes (Petrov et al. (2006)), and heuristic annotations.

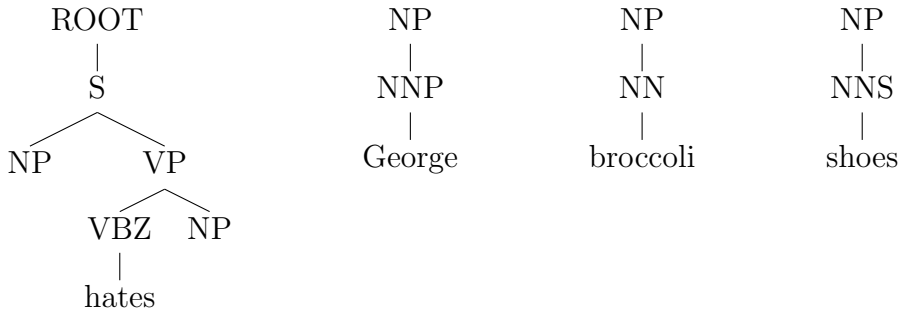


Figure 2.1: Fragments from a Tree Substitution Grammar capable of deriving the sentences "George hates broccoli" and "George hates shoes".

2.1 Tree Substitution Grammars

Tree Substitution Grammars are similar to Context Free Grammars, differing in that they allow rewrite rules of arbitrary parse tree structure with any number of nonterminal or terminal leaves. We adopt the term *fragment*, as opposed to *elementary tree* as is often used in related work, to refer to TSG rules as they are easily visualized as fragments of a complete parse tree (see Figure 2.1). We will also often refer to the root node of a fragment, which is its topmost node but is not necessary labeled with the symbol ROOT.

TSGs follow the traditional grammatical formalism of tree generation, beginning with a ROOT nonterminal leaf node and recursively substituting leaf nonterminals with fragments whose root symbol matches the leaf's symbol. A tree's *derivation* is the ordered list of the fragments used to construct it. As shown in Figure 2.2, a derivation can itself be visualized as a tree, with fragments as nodes.

One inherent difficulty in the use of TSGs is in controlling the number of fragments in grammars automatically induced from data. Given a training corpus of constituent parse trees such as the Penn Treebank, TSG induction must allow the potential inclusion of any fragment of any tree in the corpus, which with a reasonably large amount of data quickly becomes difficult to do efficiently.

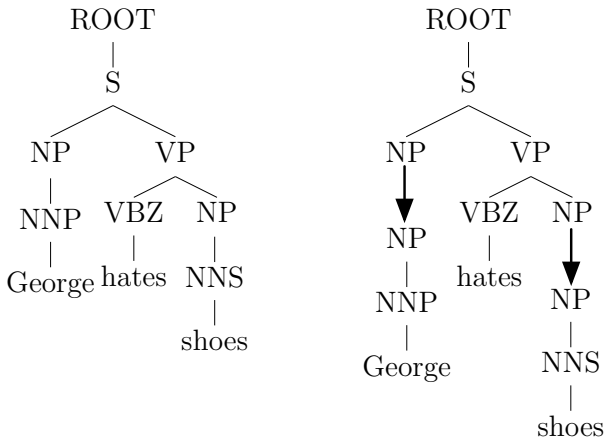


Figure 2.2: A parse tree (left) and its derivation (right) using three fragments from the TSG in Figure 2.1. The act of rewriting a nonterminal, also known as substitution, is indicated by arrows in the derivation.

When automatically induced TSGs were first proposed by Bod (1992), this problem of grammar induction was tackled with random selection of fragments or weak constraints, leading to massive grammars. A more principled technique is to use a Dirichlet process prior to model the large multinomials requireds, explicity representing only the most important fragments and backing off to a vague prior over the remainder of fragments, as presented by Cohn and Blunsom (2010b) and Post and Gildea (2009). They provide a local Gibbs sampling algorithm, and Cohn and Blunsom (2010a) later developed a block sampling algorithm with better convergence behavior. While the grammars produced by this Bayesian method fall slightly short of state of the art parsing results, they have achieved state of the art results for unsupervised grammar induction (Blunsom and Cohn (2010)) and have been extended to synchronous grammars for use in sentence compression (Yamangil and Shieber (2010)). A few years after the initial presentation of this method for TSG induction, state of the art parsing was achieved in combination with latent symbol classes in Shindo et al. (2012).

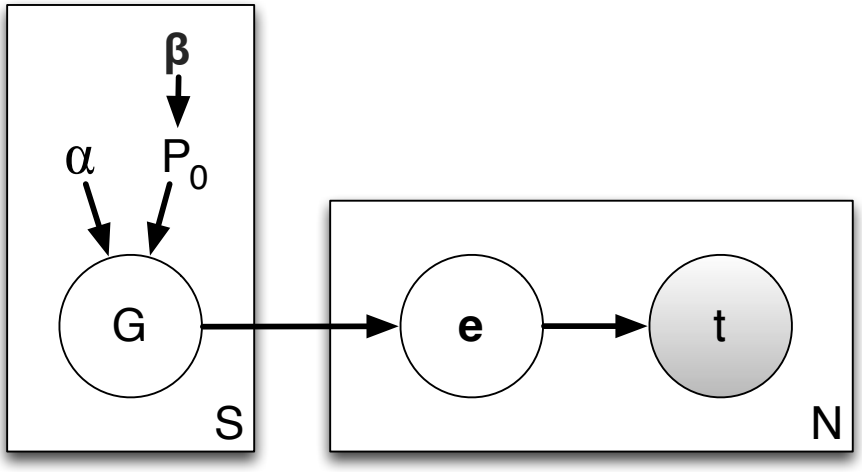


Figure 2.3: The graphical model for a Bayesian Nonparametric Tree Substitution Grammar

In another thread of research on TSG induction, Sangati and Zuidema (2011) presented an elegantly simple heuristic inspired by tree kernels that they call Double-DOP. They showed that manageable grammar sizes can be obtained from a corpus the size of the Penn Treebank by recording all fragments that occur at least twice in the data set. Using an additional heuristic to provide a distribution over fragments, DoubleDOP achieved the state of the art for TSG parsing at the time, competing closely with the absolute best results set by refinement based parsers.

2.1.1 Bayesian TSG Induction

Nonparametric Bayesian models can represent distributions of unbounded size using a dynamic parameter set that grows with the training data. Bayesian TSG induction represents the probability of the fragments that might rewrite a node as an infinite multinomial with a Dirichlet Process prior. Learning is performed with MCMC, sampling the TSG derivations of a provided Penn Treebank style corpus.

The simplest graphical model for this induction algorithm is shown in Figure 2.3,

which has the generative story

$$G_s \sim DP(\alpha_s, P_0^s) \tag{2.1}$$

$$\mathbf{e}_i \sim \mathbf{G} \tag{2.2}$$

$$\mathbf{e}_i \rightarrow t_i \tag{2.3}$$

with deterministic generation of a tree t_i from its derivation $\mathbf{e_i}$. The generative process of the grammatical formalism is referenced in Equation 2.2, referring to the production of a single derivation $\mathbf{e_i}$ and requiring a variable number of draws from the ensemble of G_s's, denoted with $\mathbf{G}$.

To make the generative process of a derivation $\mathbf{e}_i$ a bit more explicit, assume we are using the nonterminal symbol set of the Penn Treebank. Under this formalism, we begin with a node labeled ROOT and sample the first TSG fragment from G_{ROOT}, a distribution over fragments rooted at the ROOT symbol. The sampled fragment determines which samples we take next; if we generate a simple CFG rule such as $ROOT \rightarrow NP\ VP$ then we take our next sample from G_{NP}, substituting it for the new NP nonterminal leaf. If on the other hand we sample a full parse tree fragment with no nonterminal leaves then the sampling of $\mathbf{e}_i$ is complete and no further samples are required.

In these equations P_0^s is a vague base distrubution over TSG fragments, most clearly defined by its own generative process. P_0^s generates a TSG fragment in nearly the same manner as a CFG, using s as its root nonterminal symbol and recursively expanding nonterminal leaves. It differs from a CFG in that when it is about to expand a nonterminal with symbol x, with probability β_x the symbol is not expanded and left as a substitution site in the resulting TSG fragment. Using the CFG rules and parameters from the maximum likelihood estimate on the training data, P_0^s can

generate any TSG fragment rooted at s that might be used in a derivation for any tree in the data set.

Learning a Bayesian TSG from data, a process known as grammar induction, is done with sampling as variational inference approaches have yet to be developed. While initial work such as Cohn and Blunsom (2010b) used a Gibbs sampler, the Metropolis-Hastings (MH) blocked sampler of Cohn and Blunsom (2010a) is superior and we use it exclusively in this work. This algorithm integrates out the Dirichlet Processes $\mathbf{G}$ and iteratively resamples each $\mathbf{e}_i$, with a proposal distribution that is so close to the true distribution that the acceptance rate is nearly 100%. .

First, we describe the calculation of $p(\mathbf{e}_i|\mathbf{e}^{-i}, \boldsymbol{\alpha}, \mathbf{P}_0, t_i)$, the true sampling distribution where $\boldsymbol{\alpha}$ and $\mathbf{P}_0$ refer to the collection over all s of α_s and P_0^s, and $\mathbf{e}^{-i}$ is the set of sampled derivations for all other trees in the dataset. In order to use the MH algorithm, we must be able to evaluate this probability up to a normalization constant for any derivation $\mathbf{e}_i$, although we do not need to be able to sample from it. Note that due to the deterministic link between $\mathbf{e}_i$ and t_i,

$$p(\mathbf{e}_i|\mathbf{e}^{-i}, \boldsymbol{\alpha}, \mathbf{P}_0, t_i) \propto p(\mathbf{e}_i|\mathbf{e}^{-i}, \boldsymbol{\alpha}, \mathbf{P}_0) \tag{2.4}$$

if $\mathbf{e}_i$ derives t_i and zero otherwise, with the missing normalization constant equal to the sum of the probabilities of all possible derivations of t_i. Assuming we will only propose derivations that are actually possible, this means only a derivation's generative probability is necessary in MH sampling.

In general terms, a derivation is a sequence of draws from a Dirichlet Process (DP). The probability of a sequence of n draws from a DP with concentration parameter α and base distribution H is well defined, with the probability of the ith draw being X_i given as

$$P(X_i|X_{(0,\ldots,i-1)},\alpha,H) = \frac{\#(X_i \in X_{(0,\ldots,i-1)}) + \alpha H(X_i)}{|X_{(0,\ldots,i-1)}| + \alpha} \tag{2.5}$$

where $\#(X_i \in X_{(0,\ldots,i-1)})$ is the number of times X_i appears in the histogram of previous draws, $X_{(0,\ldots,i-1)}$. Due to the exchangability of draws from a Dirichlet Process, the joint probability of any set of items can be computed iteratively in any order, updating $X_{(0,\ldots,i-1)}$ with each draw. For TSGs, the probability $p(\mathbf{e}_i|\mathbf{e}^{-i},\boldsymbol{\alpha},\mathbf{P}_0)$ can therefore be calculated by initializing the histograms $X_{(0,\ldots,i-1)}$ for each G_s with the appropriate elements of $\mathbf{e^{-i}}$, and multiplying together the probabilities of drawing each fragment needed for $\mathbf{e_i}$, updating the DPs' histograms with each draw.

We now describe the proposal distribution q of Cohn and Blunsom (2010a), for which we must not only be able to evaluate $q(\mathbf{e}_i|\mathbf{e}^{-i},\boldsymbol{\alpha},\mathbf{P}_0,t_i)$, but also sample from it. We will refer to the histogram created when initializing the DP G_s from $\mathbf{e^{-i}}$ as its *cache*, and denote it with the variable C_s. To sample a fragment e_{ij} to expand a symbol s where C_s has n unique members we sample a multinomial with $n+1$ options. The first n outcomes correspond to the choice of one of the existing cache members e, and have probability

$$\text{CACHE}_s(e) = \frac{\#(e \in C_s)}{|C_s| + \alpha_s} \tag{2.6}$$

The final option corresponds to the decision to sample a TSG fragment from the base distribution P_0^s, and is chosen with the remaining probability

$$\text{BASE}_s = \frac{\alpha_s}{|C_s| + \alpha_s} \tag{2.7}$$

If we choose this outcome, then a TSG fragment is sampled using the generative model of P_0^s described above. This means that the actual probability of a single fragment e given a node to expand with symbol s is

$$q_s(e) = \text{CACHE}_s(e) + \text{BASE}_s P_0^s(e) = \frac{\#(e \in C_s) + \alpha_s P_0^s(e)}{|C_s| + \alpha_s} \tag{2.8}$$

A full derivation's generative probability is simply

$$q(\mathbf{e}_i|\mathbf{e}^{-i}, \boldsymbol{\alpha}, \mathbf{P}_0) = \prod_{j=0}^{J} q_s(e_{ij}) \tag{2.9}$$

where the derivation $\mathbf{e}_i$ contains J fragments and e_{ij} is the jth fragment of derivation $\mathbf{e}_i$. As in the case of the true distribution, as long as $\mathbf{e}_i$ derives t_i this generative probability is directly proportional to the probability $q(\mathbf{e}_i|\mathbf{e}^{-i}, \boldsymbol{\alpha}, \mathbf{P}_0, t_i)$ and is sufficient to evaluate it in MH sampling.

This distribution q over fragments is nearly identical to the incremental distribution provided by the Dirichlet Processes in the true model (Equation 2.5), with the crucial difference that q is not a stochastic process and so the distribution remains constant as samples are drawn. Intuitively, it is like freezing the state of the Dirichlet Processes after drawing $\mathbf{e}^{-\mathbf{i}}$ and using its posterior for all draws in as if they were the very next draw after observing $\mathbf{e}^{-\mathbf{i}}$. In the true model, if any of the fragments in $\mathbf{e}_i$ share the same root nonterminal s, then their joint probability will be slightly different than that provided by q as $\#(e \in C_s)$ and $|C_s|$ will be off by some small integer. However, with large training data this small integer difference will have negligible effect, resulting in a very tight approximation and a high acceptance rate when used in MH.

The final step is to show how to sample a derivation $\mathbf{e}_i$ using the proposal distribution q. The derivation $\mathbf{e}_i$ is sampled recursively, beginning at the root node of t_i. Cohn and Blunsom (2010a) contains a thorough explanation of one implementation that uses a variant of the Goodman transform (Goodman (1999)). This method encodes the decisions made by q in a CFG, allowing straightforward application of the Inside

Outside algorithm (equivalent to the sum product algorithm) and top down sampling of a CFG derivation that is isomorphic to a derivation of TSG fragments. For the sake of variety, we provide a different but theoretically equivalent implementation but do not claim one method's superiority over the other. The only true difference is that their use of the Goodman transform compresses TSG fragments that share common subtrees bottom up, while ours recognizes TSG fragments in a top down manner, compressing common paths from the root of a fragment downward.

In order to sample a derivation from q, we must compute inside probabilities that can be locally normalized across a finite set of options. The term inside probability here refers to the probability of generating the entire observed subtree from a given node, given that the algorithm is in some state at the time. The first inside probability that we need for each node n is the probability of its subtree given that it is used as a substitution site, which we will call $I(n)$. The second inside probability that we require is the probability of generating a subtree from n given that the immediate children of n are generated by a draw from the CFG in P_0, which we will call $I^*(n)$.

As with traditional inside probabilities, these quantities can be computed bottom up, which is to say that to calculate $I(n)$ and $I^*(n)$ for a node n, we assume the cached calculation of these quantities for all descendants of n. Let $CFG(r)$ refer to the CFG probability for some rule r used in all of the P_0^s, and let r_n be the CFG rule that expands n in the CFG derivation of t_i. First, we calculate $I^*(n)$ with

$$I^*(n) = CFG(r_n) \prod_{c \in child(n)} \Big(I(c)\beta_c + (1 - \beta_c) I^*(c) \Big) \tag{2.10}$$

where $child(n)$ gives the immediate child nodes of n. This records the probability of choosing the next CFG rule and then for each child node either continuing to use P_0 to derive the subtree with probability $(1 - \beta_c)$ or treating the child as a substitution site with probability β_c.

In order to calculate $I(n)$, we need to consider each outcome of the multinomial with probabilities defined in Equations 2.6 and 2.7. First, for any of the cached outcomes associated with Equation 2.6, we need to know if the cached fragment e is consistent with the tree structure that begins at n. Another way to put this is that we need to know if a fragment e overlays the tree when its root is placed at n.

To perform this subroutine, we use an efficient data structure that returns O_n, the subset of C_s that overlays the tree from n. This data structure can be described as a prefix tree over tree structures, extending the classic prefix tree over strings. The intution behind its use lies in eliminating repeated work when checking for overlays against a cache of fragments, assuming that overlays are checked in an incremental top down manner. As we check each CFG rule that composes a TSG fragment, we evaluate any rooted fragment subtree only once and recursively return the sucessful overlays. For each fragment that does overlay, the data structure also efficiently returns $leaves(n, e)$, the nodes in t_i where the nonterminal leaves of e overlay. The contribution to $I(n)$ of choosing one of the possible members of O_n is

$$I_{\text{CACHE}}(n) = \sum_{e \in O_n} \text{CACHE}_s(e) \prod_{c \in leaves(n,e)} I(c) \tag{2.11}$$

The other option, corresponding to the outcome with probability defined in Equation 2.7, occurs when we decide to generate the immediate children of n with P_0^s. This gives the full formula as

$$I(n) = I_{\text{CACHE}}(n) + \text{BASE}_s I^*(n) \tag{2.12}$$

With these inside probabilities calculated for all nodes in the tree t_i, we can easily sample a derivation tree e_i topdown. This is done with an algorithm that has two modes, the first of which occurs at nodes that are substitution sites in the derivation.

This set includes the root node of t_i, and so this is the initial mode of the sampling algorithm. In this mode, there is one sampling option corresponding to the choice of a fragment e from O_n, with probability proportional to

$$\textsc{cache}_s(e) \prod_{c \in leaves(n,e)} I(c) \tag{2.13}$$

and one option corresponding to the use of P_0^s with probability proportional to $\textsc{base}_s I^*(n)$. This finite set of values can be locally normalized and a decision sampled.

If a member of O_n is chosen, then we remain in this first mode and recurse to the members of $leaves(n, e)$. If the $\textsc{base}_s I^*(n)$ option is chosen, however, we enter the second mode. In this mode, we are building a potentially novel fragment by tacking CFG rules on incrementally. We maintain this growing fragment as we recursively add the current node's CFG expansion, and then for each child with symbol x we sample a Bernoulli with parameter β_x. If we get one, we stop growing our new fragment which makes the child node a substitution site and so we return to the first mode to choose its expansion. If we get zero, then we simply recurse to this child in mode two.

2.1.2 DoubleDOP Induction

DoubleDOP (Sangati and Zuidema (2011)) uses a simple but effective heuristic inspired by tree kernels, which are commonly used to measure similarity between two parse trees by counting the number of fragments that they share. DoubleDOP uses the same underlying technique, but records a subset of the shared fragments instead of counting them, yielding a set of fragments where each member is guaranteed to appear at least twice in the training set. This guarantee that a fragment appears in at least two trees gives this induction method its name, along with DOP which is borrowed from the term "Data Oriented Parsing" used to describe TSGs in early

work such as Bod (1992).

The fragment extraction algorithm for two trees with n and m nonterminal nodes respectively can be implemented with a $n \times m$ ternary valued chart. Initally, each cell is empty, and in the first pass a cell ij is marked if node i in the first tree has the same symbol as node j in the second. Now considering only these marked cells, a cell ij is doubly marked if the ordered list of child symbols of node i in the first tree and node j in the second also matches. Finally, each doubly marked cell is visited and used as the root of an extracted fragment.

Note that any shared fragment consisting of more than one CFG rule implies multiple other smaller shared fragments. In order to rein in this redundancy, DoubleDOP records only maximal shared fragments, those that are not contained within some larger shared fragment. This can be calculated efficiently by visiting the doubly marked cells in order of increasing row and column index. Each time a doubly marked node is visited, DoubleDOP traverses the maximal shared fragment beneath it by checking the cells corresponding to the zipped list of child indices. If one of these cells is also doubly marked, it recurses but also removes the markings from that cell so that it will not be used as the root of another shared fragment in subsequent iterations.

The main disadvantage of this method is that the complexity scales quadratically with the training set size, as all pairs of sentences must be considered. It is fully parallelizable, however, which mediates this disadvantage to some extent.

2.2 Syntactic Representations

In syntactic parsing research, the most basic approaches induce a CFG from the trees in the training data as they are provided, using the dataset's nonterminal symbols directly. Most advanced approaches refine this symbol set to more accurately

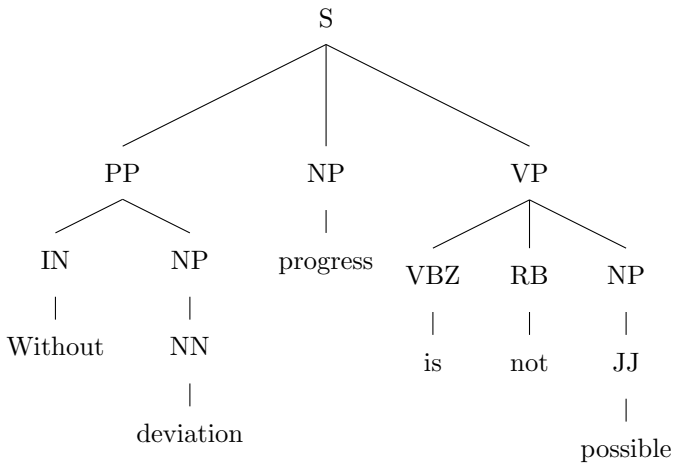
model the true distribution while satisfying the context free assumption of the model. Additionally, other forms of syntactic analysis such as the dependency format highlight different patterns in language when viewed through the lens of the structural proximity of their contents.

Many such representations exist, and most techniques that prove successful at the task of parsing have publicly available implementations, making them feasible options for incorporation into NLI systems. We investigate five variations on syntactic representation, all easily produced with freely available Java software; two with the Berkeley Parser, two with the Stanford Parser, and one with a combination of both software packages. In the context of NLI and language transfer the important thing to consider when contrasting these representations is the subtle differences between the information that is easily captured by TSG fragments in one form or another.

2.2.1 Berkeley Constituent Parses

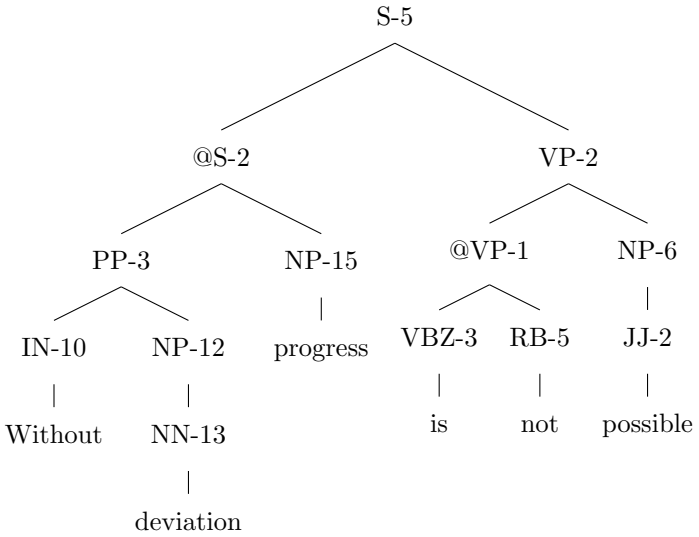
Our first two representations use the output of the Berkeley Parser (Petrov et al. (2006)), one of highest performing systems on the benchmark Penn Treebank parsing task. The basic motivating principle involved is that the traditional nonterminal symbols used in Penn Treebank parsing are too coarse to satisfy the context free assumption of a CFG. To combat this, hierarchical latent annotations are induced that split a symbol into several subtypes, and a larger CFG is estimated on this set of split nonterminals. A sentence is parsed using this large CFG and each resulting symbol is mapped back to its original unsplit supertype to produce the final parse.

To illustrate this more clearly, consider the following parsed wisdom from musician Frank Zappa.



(2.14)

The raw output of the Berkeley Parser might look something more like the following, using additional binarization nodes labeled with the @ symbol allow the use of $\mathcal{O}(n^3)$ parsing algorithms as is standard for syntactic parsing.



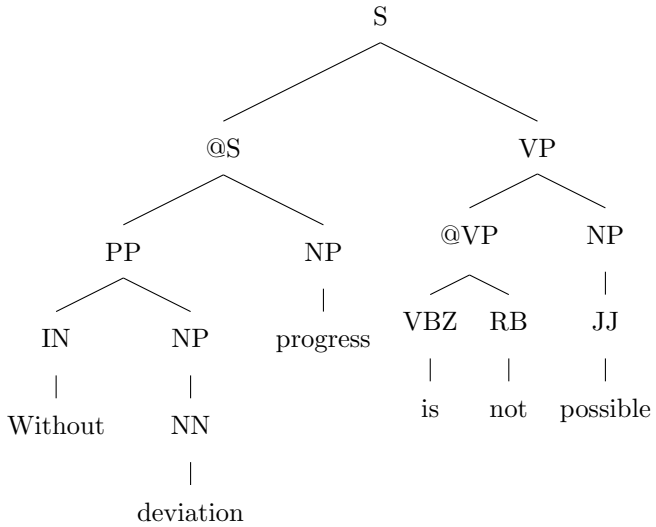
(2.15)

This parsed sentence shows how each nonterminal is annotated with a split category, and illustrates the potential advantages that this method affords. For example, consider the @VP node in the second tree, whose subtree is generated with a CFG by first choosing to produce a VBZ and RB, and then by lexicalizing each independently. These two lexicalizations are not in fact independent, as can be seen by the

combination of "is" with the RB "may", which is impossible although each are independently quite likely. Splitting the symbols as shown on the right allows us to create a special RB node that is most likely to produce "not" and VBZ node likely to produce "is". Their likely co-occurrence can then be modeled as shown by a rule with both specialized tags as children.

It is worth noting that this particular ability of split symbol grammars to coordinate lexical items is easily captured with the TSG rules that we induce on these parses, regardless of the presence of split symbols. The more orthogonal quality of these split grammars is their ability to categorize symbols that appear in similar syntactic situations. Consider that some adjectives are more likely to appear in "X is Y" sentences in the "Y" position, while some are more likely to be used directly to the left of nouns. A split symbol grammar handily captures this trait with a split POS tag, while a TSG cannot associate patterns containing different lexical items on its own.

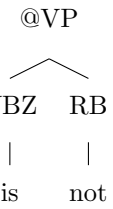
We use this raw output as our first syntactic representation, and for our second we simply remove the split symbol categories, as shown here



(2.16)

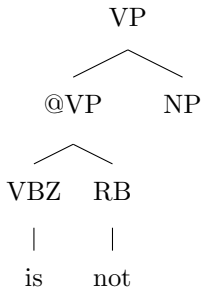
Rather then collapsing the binarization nodes as is done in parsing evaluation, we

retain them in both of these representations. The use of binarization allows us to capture patterns such as verb phrases that begin with "is not" independent of the following child constituents, using the following fragment.



(2.17)

The capabilities of TSG rules makes the use of binarization even more apt, as we can easily choose to recover the unbinarized pattern with a slightly larger fragment.



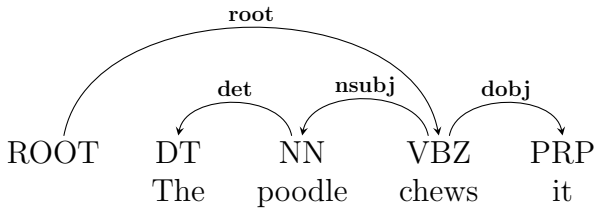
(2.18)

This choice will be made in TSG induction based on the frequency with which the combination occurs, which intuitively aligns with our goal of choosing representative features.

2.2.2 Stanford Dependency Parses

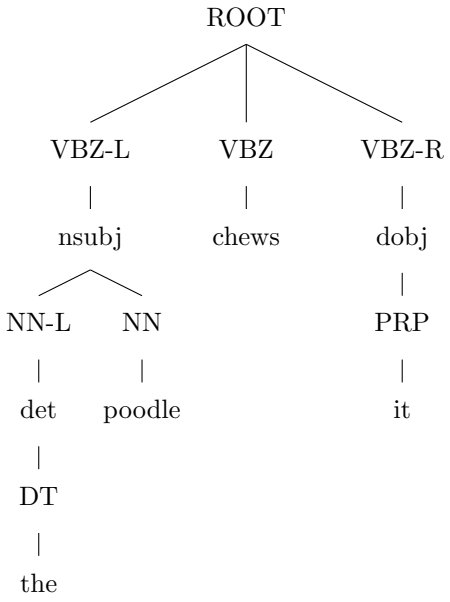
The third and fourth syntactic models we employ are derived from dependency parses produced by the Stanford parser (Marneffe et al. (2006)). In its standard form, a dependency parse is a directed tree in which each word except the special ROOT node has exactly one incoming edge and zero to many outgoing edges, where edges represent syntactic dependence. Arcs are labeled with the type of syntactic

dependence that they indicate. Following convention, we represent each word in combination with its part of speech tag as shown in



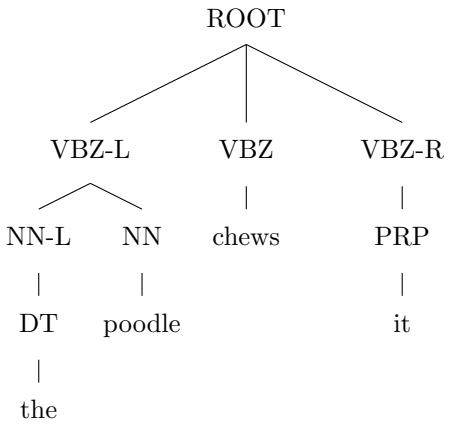
(2.19)

In order to apply the techniques of TSG induction to dependency parsed data, we implement a conversion from dependency tree to constituent form. The mechanics of this conversion are simple and illustrated by the following conversion of the dependency tree from (2.19) into (2.20), and are similar to transforms used in previous work in unsupervised dependency parsing (e.g. Carroll and Charniak (1992)).



(2.20)

Note that it is always the case that the arc labels from the dependency parses are produced by unary rules. This allows the simple removal of the nodes corresponding to arc labels, yielding our fourth syntactic model, shown here



(2.21)

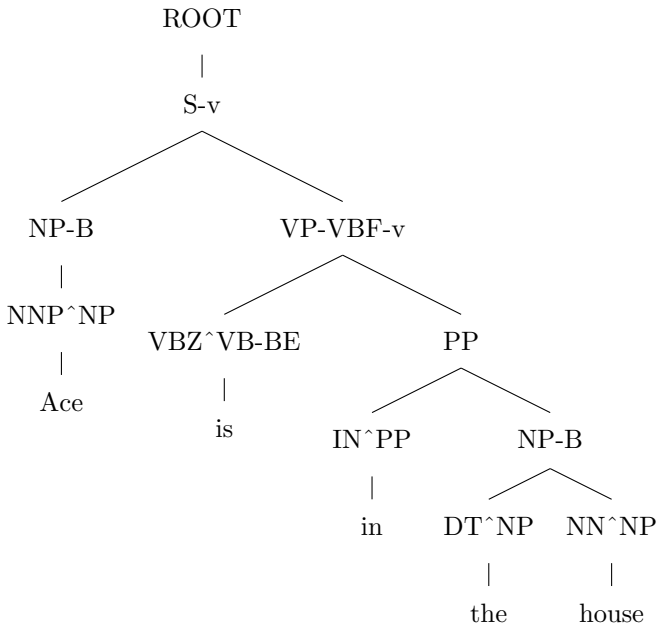
Those familiar with the Stanford Parser may be concerned that the dependency parses used here are made by a deterministic transform of a constituent parse of Penn Treebank style, and then simply transformed back into constituent form. This is especially concerning when considering the second form in which arc labels have been removed; this form can be constructed directly from the Berkeley Parse form used above, and contains no additional information. Our motivation in the investigation of dependency parses is not that they offer new information, but that they are organized differently than constituent parses. When inducing a TSG, our ability to find a useful connections is impeded by physical distance between structures. In particular, in a dependency parse, the head of the subject and the verb are always contained in some TSG fragment made up of small number of CFG rules, five or four depending on the presence of arc labels. In constituent parses, the presence of modifying phrases can arbitrarily increase this distance.

2.2.3 Stanford Heuristic Annotations

Our final variation uses the annotations internal to the Stanford Penn Treebank parser, as presented in Klein and Manning (2003). These annotations are motivated in the same way as Berkeley Parser split states, but are deterministically applied to

parse trees using linguistic motivations. Besides handling explicit tracking of binarization and parent annotation, several additional annotations are applied, such as the splitting of certain POS tags into useful categories and annotation of some nodes with their number of children or siblings.

For ease of implementation, we do not use the Stanford Parser itself to produce our trees, instead employing parses produced by the Berkeley Parser. The Stanford Parser annotations are then applied to these trees after binarization symbols were first collapsed. The following tree is an example of the actual annotations applied by this process, and includes a fair subset of the many annotation types that are used. The original symbol in each case is the leftmost string of capital letters in the resulting symbol strings shown.



(2.22)

2.3 Experiments

2.3.1 NLI Background

Work in automatic native language detection was for a time limited to International Corpus of Learner English, or ICLE (Granger et al. (2002)). The NLI task was introduced by Koppel et al. (2005), which constructed a classification system with a heterogeneous feature set consisting of function words, POS bi-grams, character n-grams, and clustered spelling errors, followed by Tsur and Rappoport (2007) which performed a continued investigation of character n-gram features. Wong and Dras (2009) considered hand picked syntactic features such as subject-verb and noun-number coordinations, and continued to investigate Probabilistic Context Free Grammar (PCFG) features from automatic parses (Wong and Dras (2011)). Another interesting contribution is Wong et al. (2011), which investigates the use of the Latent Dirichlet Allocation topic posterior as a latent feature set of reduced dimensionality.

As interest in NLI grew, the quality of the ICLE as a corpus was repeatedly called into question. The primary concern was that the ICLE was not collected with NLI in mind and so nothing was done to control correlation between topic and native language label (Brooke and Hirst (2012)). In response to this, a corpus designed specifically for NLI was recently created, and debuted in a shared task (Blanchard et al. (2013)). This data set consists of TOEFL essays, and spans 11 native languages. In addition to efforts in the construction of this data set to remove unwanted correlations with L1 label, the shared task tackled the lack of standardized notions in NLI of sample size and use of content words, specifically designating a train, development, and test set and explicitly allowing the use of all lexical items in the text.

2.3.2 ICLE

For our first set of NLI experiments, we use the International Corpus of Learner English (Version 2), which consists of raw unsegmented English text tagged with L1 labels. We consider two experimental setups that randomly subsample this corpus, with all empirical results averaged over 5 subsamplings of the full data set.

Corpus Subsampling

The first setup that we use follows Wong and Dras (2011) in analyzing Chinese, Russian, Bulgarian, Japanese, French, Czech, and Spanish L1 essays. As in their work we randomly sample 70 training and 25 test documents for each language, referring to this setup as 7x95.

The ICLE was expanded in 2009 to include a total of 16 languages, containing texts from the languages just mentioned as well as Dutch, Polish, Turkish, Norwegian, Finnish, Italian, Swedish, Tswana, and German. To take advantage of the resources available we include all of these languages but Dutch, which we exclude due to a disproportionately small amount of data with this label. We also extend the number of documents per language from 95 to 200 and use a 20/80 test train split. With these 15 languages we now have 600 test documents and a training set of approximately 80000 trees, compared to 175 test documents and approximately 14000 training trees in the 7x95 setup. We refer to this experimental setup as 15x200.

Data Preparation

Unfortunately, there is little consensus in the NLI community as to the information that should be available to a classification system. To see why this is important, consider that when asked to compose a essay a Turkish person will be much more likely to mention the city Istanbul. Indeed, if a simple logistic regressor is trained

on unigram features, the majority of highly discriminative features are precicely this sort of information. While these tendencies in topic and opinion that result from geography and culture certainly provide evidence of native language, they are unrelated to the language transfer connection that we seek to illuminate.

We choose to limit the available evidence using the common distinction between stop words and content words, replacing all content words with a single unknown word symbol. It is worth noting, however, that even this is not sufficient to completely remove all unwanted evidence, as is discussed in Brooke and Hirst (2011). An example of this in the ICLE data set is the prevalence of the CFG rule NP $\rightarrow$ NNP NNP in L1 Chinese essays. While devoid of lexical items, this construction corresponds almost exclusively to the mention of Hong Kong, one of the few major cities consisting of two words that happens to appear in the data.

Our data preproccesing pipeline is as follows: First we perform sentence segmentation with OpenNLP, and then parse each sentence with the 6 split Berkeley Parser (Petrov et al. (2006)). We then replace all terminal symbols that do not occur in a list of approximately 600 function words with the word token UNK, using the function word list distributed with the ROUGE summarization evaluation package.

We split the resulting parse trees into test and training sets and perform TSG induction on the training data in three different ways. First, to provide a baseline, we simply read off the CFG rules from the data set. Note that a CFG is a TSG with all fragments having depth one, and so this can be seen as another TSG system with a trivial form of grammar induction. Second, in the method we call BTSG, we use the Bayesian induction model with alpha tuned to 100 and run for 1000 iterations of blocked sampling. We take as our resulting finite grammar the fragments that appear in the sampled derivations. Third, we run the parameterless DoubleDOP (2DOP) induction method, which returns a comparatively large number of fragments.

Using the full 2DOP feature set for the 7x95 setup involves over 400k features, which heavily taxes the resources of an average modern computer. On the 15x200 setup, 3.2 million features are extracted, which for a 15 class problem gives over 48 million parameters and a weight vector of doubles of approximately 500MB in size. Even with sparse feature vectors our logistic regression model requires over 10GB of RAM to fit all 2400 training examples into memory and run L-BFGS optimization.

To construct a more practically useful 2DOP feature set we balance the feature set sizes between 2DOP and BTSG by passing back over the training data and counting the actual number of times each fragment recovered by 2DOP appears. We then limit the list to the n most common fragments, where n is the average number of fragments recovered by the BTSG method, around 7k and 10k for the 7x95 and 15x200 setups respectively. We refer to results with this trimmed feature set with the label 2DOP, using 2DOP(F) to refer to DoubleDOP with the full set of features.

Given each TSG, we create a binary feature function for each fragment e in the grammar such that the feature f_e is active for a document d if there exists a derivation of some tree $t \in d$ that uses e. We investigated the use of count based feature functions but found that they degraded performance, consistent with Wong et al. (2011). Classification and training was performed with the Mallet package for logistic regression using the default initialized MaxEntTrainer, which uses a Gaussian regularizer. We evaluated a Laplacian (LASSO) regularized version as well, but found that it lowered performance across the board.

Predictive Power

Classification accuracies for features using the three methods of TSG induction are shown in Table 2.1. In both experimental setups BTSG gives the highest classification accuracy and outperforms the CFG baseline. 2DOP does not perform as well,

especially on the larger experimental setup.

Model	**Acc (7x95)**	**Acc (15x200)**
CFG	72.6	69.6
2DOP	73.5	70.3
2DOP(F)	76.8	69.7
BTSG	78.4	75.7

Table 2.1: Classification accuracy (%).

For 2DOP we limit the 2DOP(F) fragments by frequency, but there may exist superior methods. Indeed, Wong and Dras (2011) claims that Information Gain is a better criteria. We experimented with Information Gain but found no significant difference in performance. We note that when all rules are used, the averaged accuracy still lags behind BTSG for both data sets.

With the BTSG model it is possible to compute the probability of a test document given each of the class labels and use the maximum conditional probability as an estimator. As expected, this method does not perform nearly as well as the use of a discriminative model, achieving 67.6 percent average accuracy on the 15x200 setup.

The confusion matrix for the 15x200 setup is shown in Figure 2.4. The most common misclassifications (shown in bold) are between the Scandinavian languages Finnish, Swedish, and Norwegian, with the Slavic languages Bulgarian, Russian, Polish, and Czech also frequently confused. The Scandinavian confusions may seem troubling, as Finnish is not in the same language family as the other Germanic languages in this group. Note that in Finland, however, Swedish is not only a second official national language, but is also compulsory for grades 7 through 9. As the data was drawn from college students, we can be nearly sure that all of the Finnish students also speak Swedish. With this in mind, the confusions not only make sense but also raise an interesting question as to the effects of L2 on L3 acquisition, as explored

	PO	**CN**	**RU**	**TR**	**NO**	**FI**	**BG**	**IT**	**JP**	**SW**	**FR**	**CZ**	**TS**	**GE**	**SP**
PO	148	1	9	3	.	4	7	3	1	8	6	6	2	2	.
CN	3	177	2	1	.	.	4	1	6	.	2	.	.	3	1
RU	3	.	132	1	4	5	9	3	5	2	6	**15**	3	9	3
TR	4	1	2	177	1	.	2	1	2	.	2	2	1	1	4
NO	1	.	2	.	152	9	4	1	.	**18**	3	2	2	1	5
FI	10	.	4	3	**19**	90	10	.	3	**34**	5	9	2	11	.
BG	**17**	.	**15**	1	5	5	126	1	.	9	5	8	2	5	1
IT	4	.	3	.	.	1	2	181	1	.	2	1	.	1	4
JP	3	3	1	1	2	2	.	1	182	1	.	.	2	1	1
SW	10	.	2	1	**22**	**25**	12	1	.	106	5	4	3	7	2
FR	4	2	3	3	2	3	8	7	.	3	157	.	.	5	3
CZ	8	.	12	2	4	10	3	3	.	4	.	145	.	5	4
TS	1	.	1	1	2	2	2	.	.	2	.	1	187	.	1
GE	6	.	8	.	4	**16**	9	3	1	6	10	8	4	124	1
SP	3	.	7	1	3	1	8	3	1	4	5	1	4	9	150

Figure 2.4: The confusion matrix for native language detection using the BTSG model, summed over five separate data set samplings. An entry in row i and column j denotes the number of documents of type i classified as type j. The native languages in order are Polish, Chinese, Russian, Turkish, Norwegian, Finnish, Bulgarian, Italian, Japanese, Swedish, French, Czech, Tswana, German, and Spanish. The top nine confusion pairs are shown in bold.

in Murphy (2005).

Feature Redundancy

We hypothesize that the inferior performance of 2DOP is to some extent due to feature redundancy, which is known to adversely effect classification performance in predictive models (Tuv et al. (2009)). Feature redundancy is a serious concern with TSG features, as it is possible to represent the same general language pattern with several slightly different fragments. Consider, for example, a noun phrase consisting of two unknown nouns. This might lead to any subset of the following fragments being included in our feature set

Figure 2.5: Equivalent but distinct fragments arising from the same pattern

While the Bayesian model is encouraged by the rich-get-richer dynamic of the sampling process to choose a small subset of these fragments to represent in derivations, 2DOP has no such motivation and can easily end up including all four.

To investigate this we approximate feature redundancy with the co-occurrence of two fragments in the set of derivations of a single sentence, as this is a necessary but not sufficient criteria for redundant representation. Considering a pair fragments e_A and e_B we can calculate the point-wise mutual information between events signifying their occurrence.

$$pmi(A;B) = \log\left(\frac{P(A|B)}{P(A)}\right) \tag{2.23}$$

where P(A) is the probability of seeing the fragment e_A in at least one derivation of a sentence. The numerator $P(A|B)$ is the probability that e_A appears in a sentence in which e_B appears. For BTSG on 7x95, the average value of this metric over all pairs (e_A, e_B) is $-.14$, while for 2DOP it is $-.01$. If e_A and e_B always co-occur, this metric approaches $-\log(P(A))$ which is greater than zero, while if they are completely exclusive the value approaches negative infinity. This shows that, as expected, BTSG yields a set of more exclusive features than 2DOP.

2.3.3 TOEFL

We contrast the syntactic formalisms on the NLI shared task experimental setup for the NAACL 2013 BEA workshop. We prepared the data in the five forms described above and induced TSGs on each version of the parsed training set with the blocked sampling algorithm of Cohn and Blunsom (2010a). The resulting rules were used as binary feature functions over documents indicating the presence of the rule in some derivation of sentence in that document. We used the Mallet implementation of a log-linear (MaxEnt) classifier with a zero mean Gaussian prior with variance .1 on

	BP	BPS	DP	DPA	KM	AVG
Acc	74.5	69.3	72.4	73.5	73.5	77.3

Figure 2.6: The resulting classification accuracies on the development set for the various syntactic forms that we considered. The forms used are plain Berkeley Parses (BP), Berkeley Parses with split symbols (BPS), dependency parses (DP), dependency parses without arc labels (DPA), and the heuristic annotations from Klein and Manning (2003) (KM). When the predictive distributions of the five models are averaged (AVG), a higher accuracy is achieved.

the classifier's weights. Our results on the development set are shown in Figure 2.6.

While a range of performance is achieved, when we construct a classifier that simply averages the predictive distributions of all five methods we get better accuracy than any model on its own. We observed further evidence of the orthogonality of these methods by looking at pairs of formalisms and observing how many development set items were predicted correctly by one formalism and incorrectly by another. This was routinely around 10 percent of the development set in each direction for a given pair, implying that gains of up to at least 20 percent classification accuracy are possible with an expert system that approaches oracle selection of which formalism to use.

As our submission to the shared task, we used the Berkeley Parser output in isolation, the average of the five classifiers, and the weighted average of the classifiers using the optimal weights on the development set (Figure 2.7). The former two models use the development set as additional training data, which is one possible explanation of the slightly higher performance of the equally weighted average model. Another explanation of note is that while the weight optimization was carried out with EM over the likelihood of the development set labels, this did not in correlate positively with classification accuracy; even as we optimized on the development set the accuracy in absolute classification of these items decreased slightly.

The confusion matrix for the evenly averaged model, our best performing system, is shown in Figure 2.8. The most frequently confused L1 pairs were Hindi and Telegu,

	BP	AVG	AVG-EM
Acc	74.7	77.5	77.0

Figure 2.7: The classification accuracies obtained on the test data using the Berkeley parser output alone (BP), the arithmetic mean of all five predictive distributions (AVG) and the weighted mean using the optimal weights from the development set as determined with EM (AVG-EM)

	ARA	CHI	FRE	GER	HIN	ITA	JPN	KOR	SPA	TEL	TUR	P	R	F
ARA	76	2	4	1	2	2	2	1	4	3	3	76.8	76.0	76.4
CHI	2	86	0	1	1	0	4	4	1	0	1	81.1	86.0	83.5
FRE	2	1	77	3	2	6	2	1	5	1	0	82.8	77.0	79.8
GER	0	1	1	91	1	1	0	0	2	0	3	86.7	91.0	88.8
HIN	2	2	1	2	71	0	0	0	0	20	2	73.2	71.0	72.1
ITA	2	0	2	1	1	84	0	1	7	0	2	79.2	84.0	81.6
JPN	3	4	0	1	0	0	83	7	1	0	1	74.1	83.0	78.3
KOR	1	6	1	1	1	0	20	65	2	1	2	69.1	65.0	67.0
SPA	4	2	4	3	2	12	0	3	66	0	4	71.7	66.0	68.8
TEL	1	2	0	0	16	0	0	0	0	81	0	76.4	81.0	78.6
TUR	6	0	3	1	0	1	1	12	4	0	72	80.0	72.0	75.8

Figure 2.8: Confusion Matrix and per class results on the final test set evaluation using the evenly averaged model.

Japanese and Korean, and Spanish and Italian. The similarity between Hindi and Telegu is particularly troubling, as they come from two completely different language families and their most obvious similarity is that they are both spoken primarily in India. This suggests that even though the TOEFL corpus has been balanced by topic that there is a strong geographical signal that is correlated with but not caused by native language.

Chapter 3

Data Driven Language Transfer Hypotheses

3.1 Introduction

Language transfer refers to the preferential use of second language grammar or words due to similarity to the native language of the speaker. It is one of several facets of Interlanguage, the degraded form of the true second language language spoken by a second language learner. A commonly cited example of language transfer is the omission of articles "a" and "the" in English by native speakers of languages without articles, such as Chinese. Another example is the dropping of certain pronouns in English by native speakers of pro-drop languages like Spanish, as in the sentence "I like it because is red". These are examples of negative language transfer, as they result in errors in the second language. Positive language transfer occurs when the preferential pattern is valid in the second language; a simple example of this is the preferential use of cognates, (e.g. "different" in English and "diférrent" in French).

The quality of a language transfer hypothesis is a tricky notion to construct and

quantify. First, an argument must show that the usage statistics of some feature of language fits the statistical profile of language transfer. One natural profile that fits the definition of language transfer is a feature that is overused in both L1 and L2 text by speakers of a certain native language in comparison to data for speakers of other native languages. The nature of this claim requires that the feature be detectable in all languages involved, including the second language. Second, whatever feature is used must translate into some human understandable concept, such as dropping of determiners or pronouns, in order to be understood by educators and implemented in a classroom setting.

Early language transfer research began to appear in publications dating back to the 1950s (Lado (1957)), with extensive follow-up in the linguistics community. In this work the formulation of language transfer hypothesis presumably occurs first in the mind of the researcher, who then seeks out methods to detect it in text. With the recent advent of computational corpus linguistic methods, language transfer detection has been approached through classification based strategies as in Jarvis and Crossley (2012) and most research in NLI. Although these classification experiments typically operate in second language data only, their primary advantage is that by constructing a feature based classifier and examining the highly discriminative features, a short list of potential transfer hypotheses is obtained. A linguist would still need to subsequently investigate each hypothesis, constructing parallels to native language data to build an argument. Classification based experiments are also an attractive mode of research, as systems can be compared empirically. However, the use of classification for evaluation implies the assumption that a set of features and classification model that can better guess the L1 label of L2 text is a better source of potential hypotheses. We propose that this is a poor implication, as it has been shown in previous research such as Brooke and Hirst (2011) that there can be a large amount of L1-discriminative

information that is derived from topic, geography, or culture.

We depart from this previous work that is heavily invested in NLI, guided by the observation that the actual ability to automatically determine L1 from text is of limited utility in the language education domain where the native language of a student is either known or easily solicited. More precisely, the new direction we take is in the pursuit of features that are individually discriminative between native languages, rather than as an ensemble. In addition, we consider metrics to eliminate data set dependence and redundancy, with the goal of a concise ranked list of features that are expressive enough to capture language transfer phenomenon. This property is described nicely by the desire for features that capture rather than cover linguistic phenomena (Johnson (2011)); while features such as character n-grams, POS tag sequences, and CFG rules may provide a usable L1 signal, each feature is likely covering only a component of a pattern instead of capturing it in full. TSG fragments, on the other hand, offer remarkable flexibility in the patterns that they can represent, potentially utilizing any contiguous parse tree structure.

In our first set of experiments, we consider monolingual (English) L2 data, as is used in NLI experiments. In the second phase we present a more complete method that not only considers L2 data but the actual native languages under investigation. This requires features whose usage frequency can be determined for each L1 background in both L1 and L2 text (e.g. in both German and English written by Germans). We end this section with discussion of several actual language transfer hypotheses that are output by our system. The patterns discussed are the top ranked elements of the list, and not only do their usage statistics provide a convincing argument that they are capturing language transfer, but the expressive power of TSG fragments allows clear translation into human notions of grammatical patterns.

3.2 Muti-grammar TSG induction

Based on our experiments in the previous section we use the Bayesian method for TSG induction, and we additionally extend the single grammar model described above to a supervised mixture of latent grammars. In our experiments we show how this more sophisticated model can be used to incorporate linguistic knowledge and extract discriminative features more effectively. The intuition behind this extension is that while the rich get richer dynamic of the DP leads to the use of a compact set of TSG rules that describes the data, the same property makes rare rules less likely to be explicitly represented in the induction cache. We observed in preliminary experiments that most TSG rules that seemed connected to language transfer are rather rare, and even in a data set of several thousand sentences a specific sentence construction will be sparsely observed.

We define a general model for TSG induction in labeled documents that combines a Hierarchical Dirichlet Process (Teh et al. (2006)) with supervised labels in a manner similar to upstream supervised LDA (Mimno and McCallum (2012)). Each data type η is given a fixed Dirichlet prior ν_η, and a hidden multinomial θ_η over grammars is drawn from this prior. The traditional grammatical model of nonterminal expansion is augmented such that to rewrite a symbol we first draw a grammar index from the sentence's θ_η and then choose a TSG fragment from that grammar.

We can express the generative model formally by defining the probability of a

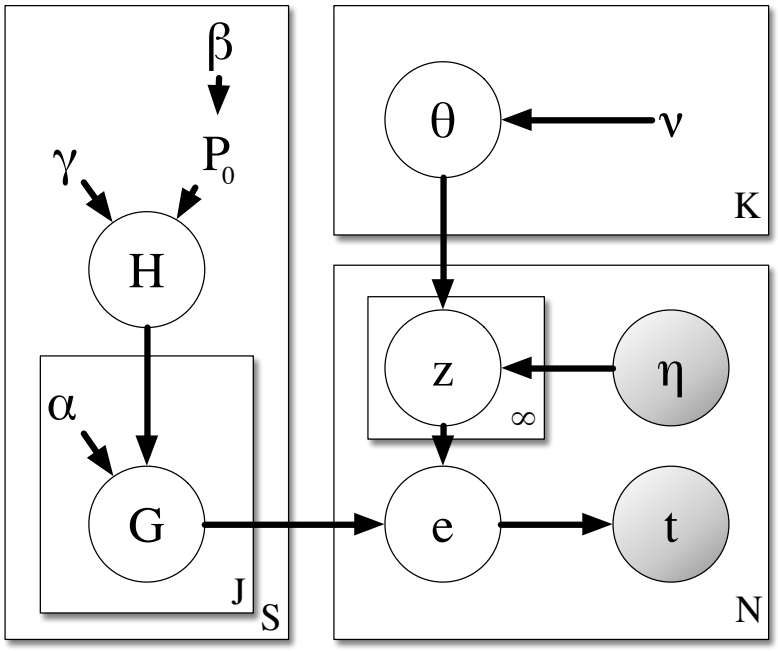


Figure 3.1: A Multigrammar TSG induction system

fragment e_{in} expanding a symbol s in a sentence with label η as

$$\theta_\eta \sim Dir(\nu_\eta)$$
$$H_s \sim DP(\gamma_s, P_0^s)$$
$$G_{js} \sim DP(\alpha_{js}, H_s)$$
$$z_{in} \sim Mult(\theta_\eta)$$
$$e_{in} \sim G_{z_{in}s}$$

The **e** in Figure 3.1 refers to the ordered set of e_{in} that forms the derivation of t_n. In order to facilitate its representation as a graphical model, this generative process samples an infinite stream of iid z_{in} variables that are consumed as the fragments e_{in} are generated for a particular tree; the unused z_{in} are simply discarded. This is equivalent to the generative process that is more natural from the point of view of

implementation, where a fresh z_{in} is sampled with each fragment's generation.

This is closely related to the application of the Hierarchical Pitman Yor Process used in Blunsom and Cohn (2010) and Shindo et al. (2012), which interpolates between multiple coarse and fine representations of TSG fragments. While the sampling algorithm is similar, our model differs in that it models several different distributions with the same support that share a common prior. To use an analogy to more common systems, their design is like smoothing a bigram model with a unigram model, while ours is like a set of bigram models that are smoothed with a universal bigram model.

To use Metropolis Hastings we require a true distribution p and a proposal distribution q for a multigrammar derivation of a tree t_i, which is a list of tuples (e, z) consisting of a TSG fragment and the index of the grammar from which it was drawn. As in the single grammar case, the deterministic production of a tree from its derivation only introduces a normalization constant on top of the generative probabiltiy of a derivation, as long as that derivation derives t_i. This means that for evaluation of p and q we can simply use their generative probability of the derivation.

One component of the true distribution is the probability of the series of fragments drawn from the various HDPs in the model, which like the DPs in the single grammar model update their model parameters after each draw. These parameter updates become more complicated with the HDP, as does removing a tree's contribution to the model parameters before resampling its derivation. The complications arise from the fact that when a draw is made from leaf DP G_{js}, with some probability a new fragment is chosen by sampling the base DP H_s. We continue our use of the notation $\#(A \in B)$ for the count of item A in histogram B. Using C_{js} to denote the cache of fragments drawn from G_{js} and C_{Hs} as the set of fragments that have been drawn from the base DP H_s, the probability that a draw from leaf DP G_{js} triggers a draw

from H_s is

$$p(\text{Use } H_s|e) \propto \frac{\alpha_{js}\frac{\#(e\in C_{Hs})+\gamma_s P_0^s(e)}{|C_{Hs}|+\gamma_s}}{|C_{js}|+\alpha_{js}} \tag{3.1}$$

$$p(\text{Do not use } H_s|e) \propto \frac{\#(e \in C_{js})}{|C_{js}|+\alpha_{js}} \tag{3.2}$$

To keep C_{Hs} up to date, we must sample this implied Bernoulli distribution on each draw from G_{js} and increment the cache C_{Hs} if the base DP was used.

Further complications arise when considering the fact that before sampling a new derivation for tree t_i we must remove the fragments from its old derivation tree from the HDP caches, using exchangability to treat all other draws as occurring before those we are about to sample. In the simple DP model this is as simple as decrementing counts, but now the entire sampling history must be considered, including the actual "table assignments" of the Chinese Restaurant Franchise (Teh et al. (2006)) that are integrated out in the treatment of the basic DP. The basic approach requires recording a table index for each fragment, but a more elegant way to handle the book-keeping is presented in Blunsom et al. (2009). In their method, explicit seating arrangements are not recorded, a histogram of table sizes for each fragment in each leaf G_{js} is kept. The probability of a new draw joining a table is proportional to its value in the histogram, which can be sampled and the table histogram updated appropriately. They also sample a table assignment when a fragment is removed, and if the chosen table has only one customer then it is removed from the histogram and the base cache C_{Hs} is also decremented.

First, we discuss the true distribution to be used in our Metropolis Hastings sampler. In addition to the caches C_{js} and C_{Hs} we also need the histogram of grammar indices from the other trees with label η, which we call with $\mathbf{z}_\eta^-$. To define the

probability of expanding a node labeled s with a fragment e using a specific grammar indexed j we require the distribution

$$p(e, z = j|\mathbf{z}_\eta^-, C_{js}, C_{Hs}, \alpha_{js}, \gamma_s, P_0^s, \nu_\eta) \tag{3.3}$$

As a derivation tree now consists of (z, e) pairs, we can evaluate the true probability of an entire derivation by iteratively calculating the probability of a derivation element with Equation 3.3, updating $\mathbf{z}_\eta^-$, C_{js}, and C_{Hs} with each draw and taking the product as the derivation's probability. Based on the dependency structure in the graphical model, Equation 3.3 factorizes into

$$p(e|z = j, C_{js}, C_{Hs}, \alpha_{js}, \gamma_s, P_0^s)p(z = j|\mathbf{z}_\eta^-, \nu_\eta) \tag{3.4}$$

The form of $p(z|\mathbf{z}_\eta^-, \nu_\eta)$ is familiar from the sampling equations for LDA.

$$p(z = j|\mathbf{z}_\eta^-, \nu_\eta) = \frac{\#(j \in \mathbf{z}_\eta^-) + \nu_{\eta j}}{|\mathbf{z}_\eta^-| + \sum \nu_\eta} \tag{3.5}$$

where $\#(j \in \mathbf{z}_\eta^-)$ is the count of grammar index j in $\mathbf{z}_\eta^-$, and $\nu_{\eta j}$ is the jth element of the Dirichlet parameter vector ν_η. The other factorized component that conditions on $z = j$ is

$$p(e|z = j, C_{js}, C_{Hs}, \alpha_{js}, \gamma_s, P_0^s) = \frac{\#(e \in C_{js}) + \alpha_{js} p_2(e|C_{Hs}, \gamma_s)}{|C_{js}| + \alpha_{js}} \tag{3.6}$$

$$p_2(e|C_{Hs}, \gamma) = \frac{\#(e \in C_{Hs}) + \gamma_s P_0^s(e)}{|C_{Hs}| + \gamma_s} \tag{3.7}$$

As in Cohn and Blunsom (2010a) we use a proposal distribution q that effectively takes a snapshot of the predictive distribution of the stochastic process. Specifically, we use the state of the DP caches (C_{js} and C_{Hs}) and histogram of z_η^- after removing

the derivation of the tree being resampled, which we denote as $C_j^* s$,$C_H^* s$, and z_η^*. Like the true distribution p, the proposal q treats the probability of a derivation as the product of the probability of its elements, with the probability of one element $(e, z = j)$ defined by

$$q(e, z = j) = q(e|z = j, C_{js}^*, C_{Hs}^*, \alpha_{js}, \gamma_s, P_0^s) q(z = j|z_\eta^*, \nu_\eta) \tag{3.8}$$

These factored components that mirror the true distribution, replacing the dynamic histograms that update with each draw in the true HDP with static histograms that remain fixed.

$$q(z = j|z_\eta^*, \nu_\eta) = \frac{\#(j \in z_\eta^*) + \nu_{\eta j}}{|z_\eta^*| + \sum \nu_\eta} \tag{3.9}$$

$$q(e|z = j, C_{js}^*, C_{Hs}^*, \alpha_{js}, \gamma_s, P_0^s) = \frac{\#(e \in C_{js}^*) + \alpha_{js} q_2(e|C_{Hs}^*, \gamma_s)}{|C_{js}^*| + \alpha_{js}} \tag{3.10}$$

$$q_2(e|C_{Hs}^*, \gamma) = \frac{\#(e \in C_{Hs}^*) + \gamma_s P_0^s(e)}{|C_{Hs}^*| + \gamma_s} \tag{3.11}$$

While this provides the ability to evaluate $q(e, z = j)$, in order to define a sampling algorithm we rephrase q with the following generative story. First, we sample a grammar index j with Equation 3.9 and then we decide if we will draw a cached rule or generate from P_0^s. Reorganizing the equations above gives the probability of choosing a certain cached rule e as

$$q_C(e|z = j) = \frac{\#(e \in C_{js}^*) + \alpha_{js} \frac{\#(e \in C_{Hs}^*)}{|C_{Hs}| + \gamma_s}}{|C_{js}^*| + \alpha_{js}} \tag{3.12}$$

while the probability of generating from P_0^s is

$$q(\text{use } P_0^s|z = j) = \frac{\alpha_{js} \frac{\gamma_s}{|C_{Hs}| + \gamma_s}}{|C_{js}^*| + \alpha_{js}} \tag{3.13}$$

To sample a derivation under q we compute inside probabilities and sample a derivation top down, as in the single grammar case. However, as we now are able to choose a grammar at each generation, for an J grammar model we must record $J+1$ inside probabilities at each node. First there are J options representing the case where the node is used as a substitution site and grammar j is used to expand the node.

$$I_j(n) = q(z=j)\left(\sum_{e\in O_n}\left(q_C(e|z=j)\prod_{c\in leaves(n,e)}\sum_k I_k(c)\right) + q(\text{use } P_0^s)I^*(n)\right) \tag{3.14}$$

The inside probability representing the probability of a subtree given that a node n's children are generated from P_0 must also include these inside probabilities,

$$I^*(n) = CFG(r_n)\prod_{c\in child(n)}(1-\beta_c)I^*(c) + \beta_c\sum_k I_k(c) \tag{3.15}$$

By careful choice of the number of grammars K, the Dirichlet priors ν, and the backoff concentration parameter γ, a variety of interesting models can easily be defined, as demonstrated in our experiments.

3.3 Monolingual Experiments

3.3.1 Corpus Description

We perform analysis of English text from Chinese, German, Spanish, and Japanese L1 backgrounds drawn from four corpora. The first three consist of responses to essay prompts in educational settings, while the fourth is submitted by users in an internet forum.

The first corpus is the International Corpus of Learner English, or ICLE (Granger et al. (2002)), a mainstay in NLI that has been shown to exhibit a large topic bias due to correlations between L1 and the essay prompts used (Brooke and Hirst (2011)). The second is the International Corpus of Crosslinguistic Interlanguage (ICCI) (Tono et al. (2012)), which is annotated with sentence boundaries and had not yet been used in NLI at the time of this work. The third is the public sample of the Cambridge International Corpus (FCE), and consists of short prompted responses. One quirk of the FCE data is that several responses are written in the form of letters, leading to skewed distributions of the specialized syntax involved with use of the second person. The fourth is the Lang8 data set introduced by Brooke and Hirst (2011). This data set is free of format, with no prompts or constraints on writing aids. The samples are often very short and are qualitatively the most noisy of the four data sets.

We treat each sentence as an individual datum. As document length can vary dramatically, especially across corpora, this gives increased regularity to the number of features per data item. More importantly, this creates a rough correspondence between feature co-occurrence and the expression of the same underlying linguistic phenomenon, which is desirable for automatic redundancy metrics.

We automatically detect sentence boundaries when they are not provided, and parse all corpora with the 6-split Berkeley Parser. We then replace all word tokens that do not occur in a list of common words with an unknown word symbol, UNK.

While these are standard data preprocessing steps, from our experience with this problem we propose additional practical considerations. First, we filter the parsed corpora, retaining only sentences that are parsed to a Clause Level[1] tag. This is primarily due to the fact that automatic sentence boundary detectors must be used on the ICLE, Lang8, and FCE data sets, and false positives lead to sentence fragments that are parsed as NP, VP, FRAG, etc. The wild internet text found in the Lang8

[1] S, SINV, SQ, SBAR, or SBARQ

data set also yields many non-Clause Level parses from non-English text or emotive punctuation. Sentence detection false negatives, on the other hand, lead to run-on sentences, and so we additionally remove sentences with more than 40 words.

We also impose a simple preprocessing step for better treatment of proper nouns. Due to the geographic distribution of languages, the proper nouns used in a writer's text naturally present a strong L1 signal. The obvious remedy is to replace all proper nouns with UNK, but this is unfortunately insufficient as the structure of the proper noun itself can be a covert signal of these geographical trends. To fix this, we also remove all proper noun left sisters of proper nouns. We choose to retain the rightmost sister node in order to preserve the plurality of the noun phrase, as the rightmost noun is most likely the lexical head.

From these parsed, UNKed, and filtered corpora we draw 2500 sentences from each L1 background at random, for a total of 10000 sentences per corpus. The exception is the FCE corpus, from which we draw 1500 sentences per L1 due to its small size.

3.3.2 Feature Selection

Feature selection itself is a well studied problem, and the most thorough systems address both relevancy and redundancy. While some work tackles these problems by optimizing a metric over both simultaneously (Peng et al. (2005)), we decouple the notions of relevancy and redundancy to allow ad-hoc metrics for either, similar to the method of Yu and Liu (2004). Additionally, we address automatic methods for removing patterns whose discriminative signal does not generalize across data sets.

Dataset Independence

The first step in our L1 signal extraction pipeline controls for patterns that occur too frequently in certain combinations of native language and data set. Such patterns

	L_1	L_2
D_1	1000	500
D_2	100	50

	L_1	L_2
D_1	1000	500
D_2	750	750

Figure 3.2: Two hypothetical feature profiles that illustrate the problems with filtering only on data set independence, which prefers the right profile over the left. Our method has the opposite preference.

arise primarily from the reuse of essay prompts in the creation of certain corpora, and we construct a hard filter to exclude features of this type.

A simple first choice of metric would be the variance of a fragment's frequency across data sets. However, this misses the subtle but important point that corpora have different qualities such as register and author proficiency and so some variance in frequency of all fragments is to be expected. Instead, we treat the set of sentences containing an arbitrary feature X as a set of observations of a pair of categorical random variables L and D, representing native language and data set respectively, and measure the dependence of L and D. Intuitively, this metric prefers features that exhibit the same relative usage patterns across languages regardless of data set, although the prevalance of the pattern across data sets may vary.

To see why this treatment is superior, consider the outcomes for the two hypothetical features shown in Figure 3.2. The left table has a high data set dependence but exhibits a clean twofold preference for L_1 in both data sets, making it a desirable feature to retain. Conversely, the right table shows a feature where the distribution is uniform over data sets, but has language preference in only one. This is a sign of either a large variance in usage or some data set specific tendency, and in either case we can not make confident claims as to this feature's association with any native language.

The L-D dependence can be measured with Pearson's χ^2 test, although the specifics of its use as a filter deserve some discussion. As we eliminate the features for which the

null hypothesis of independence is rejected, our noisy data will cause us to overzealously reject. In order to prevent the unneccesary removal of interesting patterns, we use a very small p value as a cutoff point for rejection. In all of our experiments the χ^2 value corresponding to $p < .001$ is in the twenties; we use $\chi^2 > 100$ as our criteria for rejection.

Another possible source of error is the sparsity of some features in our data. To avoid making predictions of rules for which we have not observed a sufficient number of examples, we automatically exclude any rule with a count less than five for any L-D combination. This also satisfies the common requirements for validity of the χ^2 test that require a minimum number of 5 expected counts for every outcome.

Relevancy

Whatever list is returned by a system such as ours should clearly be sorted, with "better" hypotheses at the top. What we call relevancy is a quantification of our confidence that its usage profile provides evidence for language transfer. We define a simple evaluation metric, per-feature loss, and use it to contrast three relevancy ranking metrics: Information Gain (IG), Symmetric Uncertainty (SU), and χ^2 statistic. The first two metrics are commonly considered superior in classification problems where the goal is accuracy using an ensemble of features, but we show that when per-feature loss is the goal and there exist very rare but desirable features, the χ^2 statistic is a better choice.

The formulas for these ranking metrics are

$$IG(L, X_i) = H(L) - H(L|X_i)$$

$$SU(L, X_i) = 2\frac{IG(L, X_i)}{H(L) + H(X_i)}$$

$$\chi^2(X_i) = \sum_m \frac{(n_{im} - \frac{N_i}{M})^2}{\frac{N_i}{M}}$$

We define L as the Multinomial distributed L1 label taking values in $\{1, ..., M\}$ and X_i as a Bernoulli distributed indicator of the presence or absence of the ith feature, which we represent with the events X_i^+ and X_i^- respectively. We use the Maximum Likelihood estimates of these distributions from the training data to compute the necessary entropies for IG and SU. For the χ^2 metric we use n_{im}, the count of sentences with L1 label m that contain feature X_i, and their sum over classes N_i.

While SU is often preferred over IG in feature selection for several reasons, their main difference in the context of selection of binary features is the addition of $H(X_i)$ in the denominator, leading to higher values for rare features under SU. This helps to counteract a subtle preference for common features that these metrics can exhibit in data such as ours, as shown in Figure 3.3. The source of this preference is the overwhelming contribution of $p(X_i^-)H(L|X_i^-)$ in $IG(L, X_i)$ for rare features, which will be essentially the maximum value of $\log(M)$. In most classification problems a frequent feature bias is a desirable trait, as a rare feature is naturally less likely to appear and contribute to decision making.

We note that binary features in sentences are sparsely observed, as the opportunity for use of the majority of patterns will not exist in any given sentence. This leads to a large number of rare features that are nevertheless indicative of their author's L1. The χ^2 statistic we employ is better suited to retain such features as it only deals

	IG	SU	χ^2
r	.84	.72	.15

Figure 3.3: Sample Pearson correlation coefficients between different ranking functions and feature frequency over a large set of TSG features.

with counts of sentences containing X_i.

The ranking behavior of these metrics is highlighted in Figure 3.4. We expect that features with profiles like X_a and X_b will be more useful than those like X_d, and only χ^2 ranks these features accordingly. Another view of the difference between the metrics is taken in Figure 3.5. As shown in the left plot, IG and SU are nearly identical for the most highly ranked features and significantly different from χ^2.

	L_1	L_2	L_3	L_4	IG	SU	χ^2
X_a	20	5	5	5	**.0008**	**.0012**	**19.29**
X_b	40	20	20	20	.0005	.0008	**12.0**
X_c	2000	500	500	500	**.0178**	**.0217**	**385.7**
X_d	1700	1800	1700	1800	**.0010**	**.0010**	5.71

Figure 3.4: Four hypothetical features in a 4 label classification problem, with the number of training items from each class using the feature listed in the first four columns. The top three features under each ranking are shown in bold.

Redundancy

The second component of thorough feature selection is the removal of redundant features. From an experimental point of view, it is inaccurate to compare feature selection systems under evaluation of the top n features or the number of features with ranking statistic at or beyond some threshold if redundancy has not been taken into account. Furthermore, as our stated goal is a list of discriminative patterns, multiple representations of the same pattern clearly degrade the quality of our output. This is

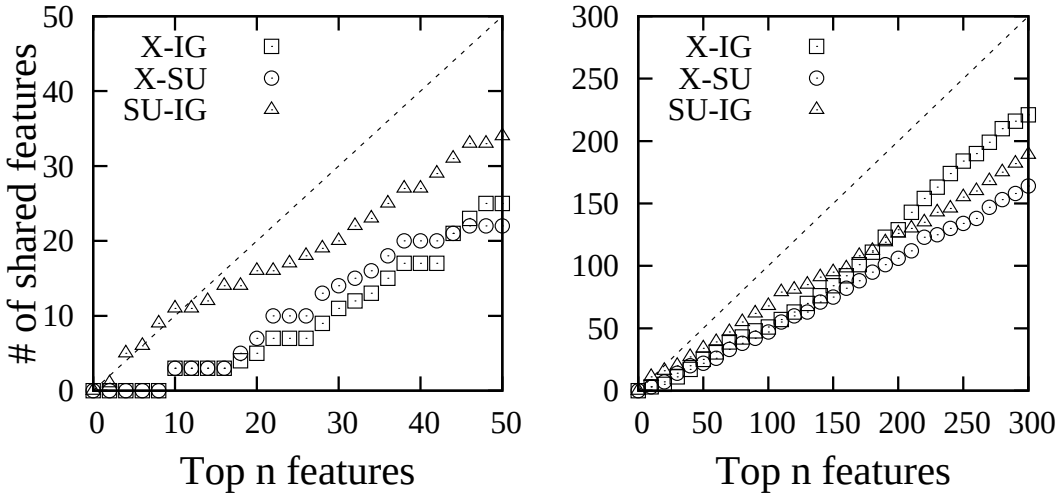


Figure 3.5: For all pairs of relevancy metrics, we show the number of features that appear in the top n of both. The result for low n is highlighted in the left plot, showing a high similarity between SU and IG.

especially necessary when using TSG fragments as features, as it is possible to define many slightly different rules that essentially represent the same linguistic act.

Redundancy detection must be able to both determine that a set of features are redundant and also select the feature to retain from such a set. We use a greedy method that allows us to investigate different relevancy metrics for selection of the representative feature for a redundant set (Yu and Liu (2004)). The algorithm begins with a list S containing the full list of features, sorted by an arbitrary metric of relevancy. While S is not empty, the most relevant feature X^* in S is selected for retention, and all features X_i are removed from S if $R(X^*, X_i) > \rho$ for some redundancy metric R and some threshold ρ.

We consider two probabilistic metrics for redundancy detection, the first being SU, as defined in the previous section. We contrast this metric with Normalized Pointwise Mutual Information (NPMI) which uses only the events $A = X_a^+$ and $B = X_b^+$ and has a range of [-1,1].

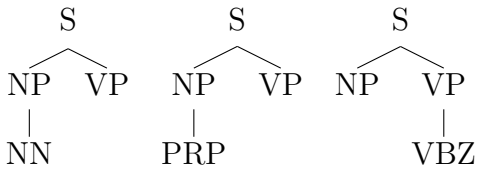


Figure 3.6: Three similar fragments that highlight the behavior of the structural redundancy metric; the first two fragments are not considered redundant, while the third is made redundant by either of the others.

$$\text{NPMI}(X_a, X_b) = \frac{\log(P(A|B)) - \log(P(A))}{-\log(P(A, B))}$$

Another option that we explore is the structural redundancy between TSG rules themselves. We define a 0-1 redundancy metric such that $R(X_a, X_b)$ is one if there exists a fragment that contains both X_a and X_b with a total number of CFG rules less than the sum of the number of CFG rules in X_a and X_b. The latter constraint ensures that X_a and X_b overlap in the containing fragment. Note that this is not the same as a nonempty set intersection of CFG rules, as can be seen in Figure 3.6.

3.3.3 Results

Relevancy Metrics

The traditional evaluation criterion for a feature selection system such as ours is classification accuracy or expected risk. However, as our desired output is not a set of features that capture a decision boundary as an ensemble, a per feature risk evaluation better quantifies the performance of a system for our purposes. We plot average risk against number of predicted features to view the rate of quality degradation under a relevancy metric to give a picture of a each metric's utility.

The per feature risk for a feature X is an evaluation of the ML estimate of $P_X(L) = P(L|X^+)$ from the training data on T_X, the test sentences that contain the feature X. The decision to evaluate only sentences in which the feature occurs removes an

implicit bias towards more common features.

We calculate the expected risk $\mathcal{R}(X)$ using a 0-1 loss function, averaging over T_X.

$$\mathcal{R}(X) = \frac{1}{|T_X|} \sum_{t \in T_X} P_X(L \neq L_t^*) \tag{3.16}$$

where L_t^* is the gold standard L1 label of test item t. This metric has two important properties. First, given any true distribution over class labels in T_X, the best possible $P_X(L)$ is the one that matches these proportions exactly, ensuring that preferred features make generalizable predictions. Second, it assigns less risk to rules with lower entropy, as long as their predictions remain generalizable. This corresponds to features that find larger differences in usage frequency across L1 labels.

The alternative metric of per feature classification accuracy creates a one to one mapping between features and native languages. This unnecessarily penalizes features that are associated with multiple native languages, as well as features that are selectively dispreferred by certain L1 speakers. Also, we wish to correctly quantify the distribution of a feature over all native languages, which goes beyond correct prediction of the most probable.

Using cross validation with each corpus as a fold, we plot the average $\mathcal{R}(X)$ for the best n features against n for each relevancy metric in Figure 3.7. This clearly shows that for highly ranked features χ^2 is able to best single out the type of features we desire. Another point to be taken from the plot is that it is that the top ten features under SU are remarkably inferior. Inspection of these rules reveals that they are precisely the type of overly frequent but only slightly discriminative features that we predicted would corrupt feature selection using IG based measures.

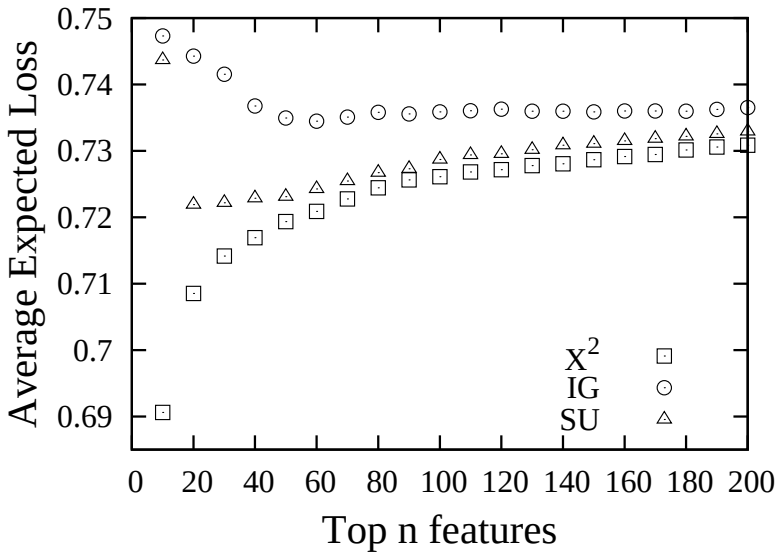


Figure 3.7: Per-feature Average Expected Loss plotted against top N features using χ^2, IG, and SU as a relevancy metric

Redundancy Metrics

We evaluate the redundancy metrics by using the top n features retained by redundancy filtering for ensemble classification. Under this evaluation, if redundancy is not being effectively eliminated performance should increase more slowly with n as the set of test items that can be correctly classified remains relatively constant. Additionally, if the metric is overzealous in its elimination of redundancy, useful patterns will be eliminated leading to diminished increase in performance. Figure 3.8 shows the tradeoff between Expected Loss on the test set and the number of features used with SU, NPMI, and the overlap based structural redundancy metric described above. We performed a coarse grid search to find the optimal values of ρ for SU and NPMI.

Both the structural overlap hueristic and SU perform similarly, and outperform NPMI. Analysis reveals that NPMI seems to overstate the similarity of large fragments with their small subcomponents. We choose to proceed with SU, as it is not only

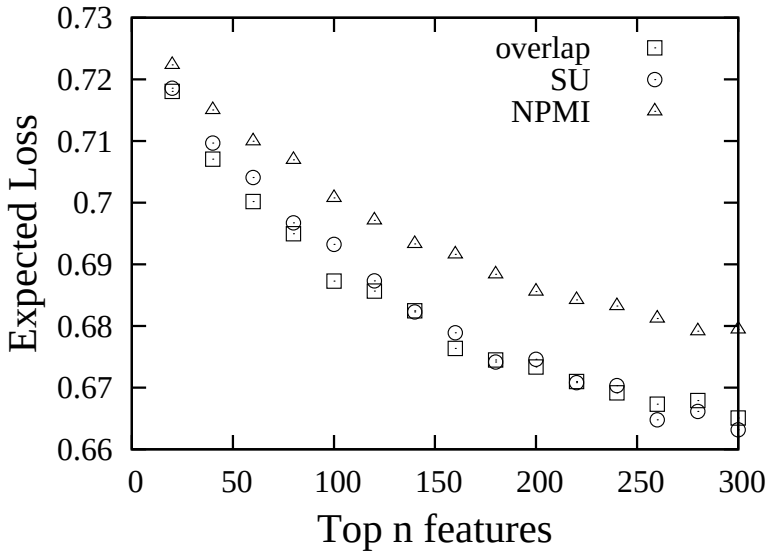


Figure 3.8: The effects of redundancy filtering on classification performance using different redundancy metrics. The cutoff values (ρ) used for SU and NPMI are .2 and .7 respectively.

faster in our implementation but also can generalize to feature types beyond TSG rules.

TSG Induction

We demonstrate the flexibility and effectiveness of our general model of mixtures of TSGs for labeled data by example. The tunable parameters are the number of grammars K, the Dirichlet priors ν_η over grammar distributions for each label η, and the concentration parameter γ of the smoothing DP.

For a first baseline we set the number of grammars $K = 1$, making the Dirichlet priors ν irrelevant. With a large $\gamma = 10^{20}$, we essentially recover the basic block sampling algorithm of Cohn and Blunsom (2010a). We refer to this model as M1. Our second baseline model, M2, sets K to the number of native language labels, and sets the ν variables such that each η is mapped to a single grammar by its L1 label,

	$p < .1$	$p < .05$	$p < .01$	$p < .001$
M1	56.5(3.1)	54.5(3.0)	49.8(2.7)	45.1(2.5)
M2	55.3(3.7)	53.7(3.6)	49.1(3.3)	44.7(3.0)
M3	**59.0**(**4.1**)	**57.2**(**4.1**)	**52.4**(**3.6**)	**48.4**(**3.3**)
M4	58.9(3.8)	57.0(3.7)	51.9(3.4)	47.2(3.1)

Figure 3.9: The percentage of rules from each model that reject L1 independence at varying levels of statistical significance. The first number is with respect to the number rules that pass the L1/corpus independence and redundancy tests, and the second is in proportion to the full list returned by grammar induction.

creating a naive Bayes model. For M2 and the subsequent models we use $\gamma = 1000$ to allow moderate smoothing.

We also construct a model (M3) in which we set $K = 9$ and ν_η is such that three grammars are likely for any single η; one shared by all η with the same L1 label, one shared by all η with the same corpus label, and one shared by all η. We compare this with another $K = 9$ model (M4) where the ν are set to be uniform across all 9 grammars.

We evaluate these systems on the percent of their resulting grammar that rejects the hypothesis of language independence using a χ^2 test. Slight adjustments were made to α for these models to bring their output grammar size into the range of approximately 12000 rules. We average our results for each model over single states drawn from five independent Markov chains.

Our results in Figure 3.9 show that using a mixture of grammars allows the induction algorithm to find more patterns that fit arbitrary criteria for language dependence. The intuition supporting this is that in simpler models a given grammar must represent a larger amount of data that is better represented with more vague, general purpose rules. Dividing the responsibility among several grammars lets rare patterns form clusters more easily. The incorporation of informed structure in M3

further improves the performance of this latent mixture technique.

3.4 Mutilingual Experiments

3.4.1 Corpus Description

These experiments are made possible primarily due to the Universal Dependency Treebank, or UTB (McDonald et al. (2013)), that is under ongoing development at Google. They define a common symbol and edge label set that can be used to represent text in a variety of languages using dependency parse trees. They also release conversion software from the constituent parse format and symbol set of the ubiquitous Penn Treebank, allowing us to convert the ETS Corpus of Non-Native English (Blanchard et al. (2013)), which we will call TOEFL as it is drawn from TOEFL® exam essays. Limited by the intersection of languages across these data sets, we take French, Spanish, and German as our set of L1s with English as the L2. The UTB provides native language data, containing around 15000 sentences of human annotated text for each L1 and the TOEFL data consists of over 10K sentences per L1 label . Finally, we use the Penn Treebank as our source of native English data, for a total of seven data types; four in English, and one in each L1.

3.4.2 Methodology

Our methodology can be broken down into four steps that are representative of systems that propose language transfer hypotheses in general. The first is the definition of a class of features $\mathcal{F}$ such that a single feature $F \in \mathcal{F}$ is capable of capturing language transfer phenomenon. The second is a universal representation of both L1 and L2 data that allows us to count the occurrences of any F in an arbitrary sentence. Third, as any sufficiently expressive $\mathcal{F}$ is likely to be very large, a method

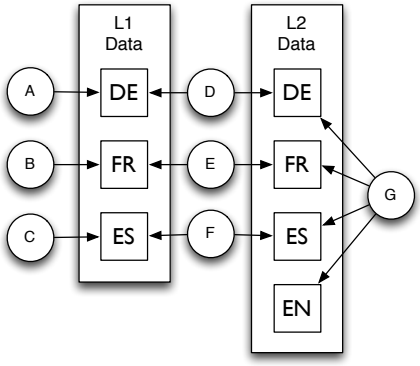
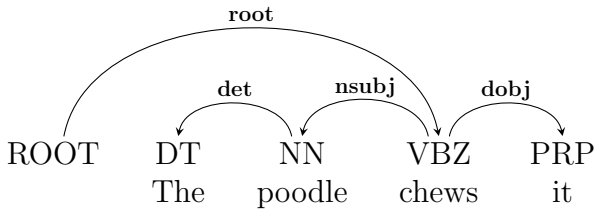


Figure 3.10: The multi-grammar induction setup used in our experiments. Squares indicate data types, and circles indicate grammars. Data type labels indicate the native language of the speaker, and all L2 data is in English.

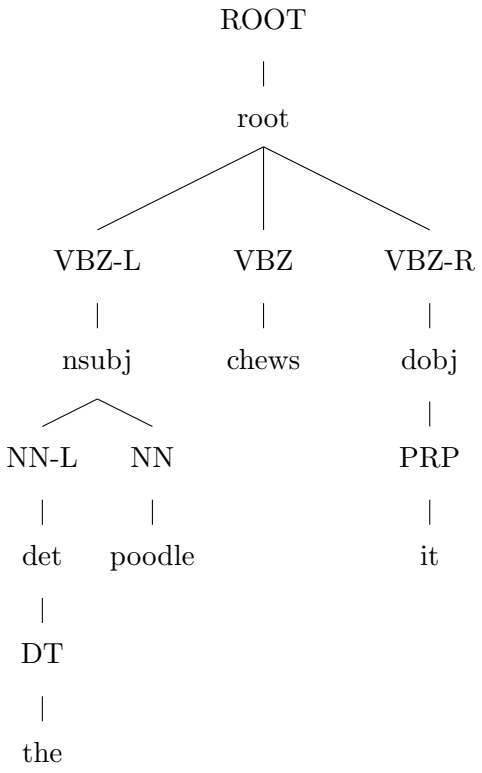
is required to propose an initial candidate list $C \subset \mathcal{F}$. Finally, we refine C into a ranked list H of language transfer hypotheses, where H has also been filtered to remove redundancy.

In this work we continue to let $\mathcal{F}$ be the set of Tree Substitution Grammar (TSG) fragments in our data, which allows any connected syntactic structure to be used as a feature. As such, our universal representation of L1/L2 data must be a constituent tree structure of the general form used in syntactic parsing experiments on the Penn Treebank. The UTB gets us most of the way to our goal, defining a dependency grammar with a universal set of part of speech (POS) tags and dependency arc labels.

Two barriers remain to the use of standard TSG induction algorithms. The first is to define a mapping from the dependency tree format to constituency format. We use the following dependency tree to illustrate our transformation.



Under our transformation, the above dependency parse becomes



We also require a multilingual lexicon in the form of a function $M_L(w)$ for each language L that maps words to clusters representing their meaning. In order to avoid cultural cues and reduce noise in our mapping, we restrict ourselves to clusters that correspond to a list of L2 stopwords. Any L2 words that do not appear on this list are mapped to the unknown "UNK" symbol, as are all foreign words that are not good translations of any L2 stopword. Multiple words from a single language can map to the same cluster, and it is worth noting that this is true for L2 stopwords as well.

To determine the mapping functions M_L we train IBM translation models in both directions between the L2 and each L1. We create a graph in which nodes are words, either the L2 stopwords or any L1 word with some translation probability to or from one of the L2 stopwords. The edges in this graph exist only between L2 and L1 words, and are directed with weight equal to the IBM model's translation probability of the

EN	my	will	little	very	after following
FR	mon mes	seront,seta feront,fera aura	peu	très	suite,suivantes suivants,après
DE	meine,meinem mein,meiner meinen	werden,wird willen,wille	wenig	sehr	nach,folgende folgenden
ES	mi,mis	tendrá,será	poco,poca	muy	tras,después siguientes

Figure 3.11: Sample clusters from our automatic mapping M_L, in the words in a column are mapped to a universal stopword index. The automatic method is effective, but still contains both false negatives, such as lack of "meines" for German in the first column, and false positives such as "aura" in French, which is the third person future tense of "avoir".

edge's target given its source. We construct M_L by removing edges with weight below some threshold and calculating the connected components of the resulting graph. We then discard any cluster that does not contain at least one word from each L1 and at least one L2 stopword. Examples of the resulting clusters can be seen in Figure 3.11.

To propose a candidate list C, we use the TSG induction technique described above that simultaneously induces multiple TSGs from data that has been partitioned into labeled types. For an experimental setup that considers n different L1s, we use $2n+1$ data types; Figure 3.10 shows the exact layout used in our experiments. Besides the necessary n data types for each L1 in its actual native language form and n in L2 form, we also include L2 data from L2 native speakers. We also define $2n+1$ grammars. We begin with n grammars that can each be used exclusively by one native language data type, representing behavior that is unique to each native language (grammars A-C in Figure 3.10) . This is done for the L2 as well (grammar G). Finally, we create an interlanguage grammar for each of our L1 types that can be used in derivation of both L1 and L2 data produced by speakers of that L1 (grammars D-F).

The final step is to filter and rank the TSG fragments produced in C. For redundancy filtering we use Symmetric Uncertainty, described in our monolingual experiments, and for ranking we use a variant of the expected per feature loss, defined in Equation 3.16. While in the monolingual experiments the predictive distribution $P_F(L)$ is determined by the observed counts of F in L2 training data, we take our estimates directly from the L1 data of the languages under study.

The final result is a ranked and filtered list of hypotheses H. The elements of H can be subjected to further investigation by experts and the accompanying histogram of counts contains the relevant empirical evidence. As more data is added, the uncertainty in the relative proportions of these histograms and their corresponding $\mathcal{R}$ is decreased. One additional benefit of our method is that TSG induction is a random process, and repeated runs of the sampling algorithm can produce different features. Since redundancy is filtered automatically, these different feature lists can be combined and processed to potentially find additional features given more computing time.

3.4.3 Results

When calculating metrics such as redundancy and $\mathcal{R}(F)$ we use all available data. For TSG sampling, we balance our data sets to 15000 sentences from each data type and sample using the Enbuske sampler that was released with Swanson and Charniak (2013). To construct word clusters, we use Giza++ (Och and Ney (2003)) and train on the Europarl data set (Koehn (2005)), using .25 as a threshold for construction on connected components.

Given our setup of three native languages, a feature with $\mathcal{R}(F) < .66$ is a candidate for language transfer. However, several members of our filtered list have $\mathcal{R}(F) > .66$, which is to say that their L2 usage does not mirror L1 usage. This is to be expected in

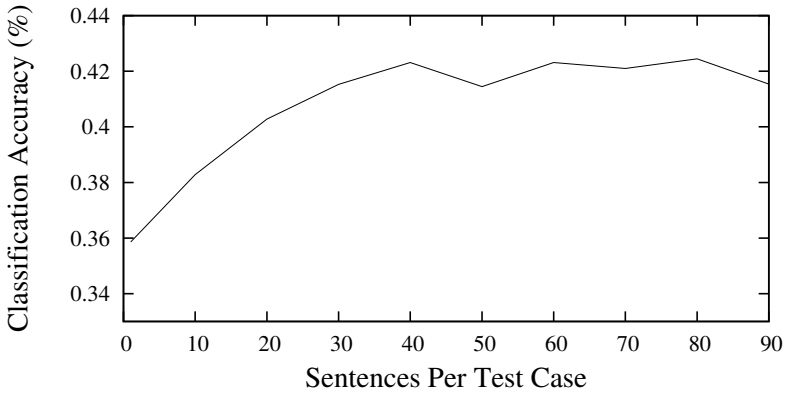


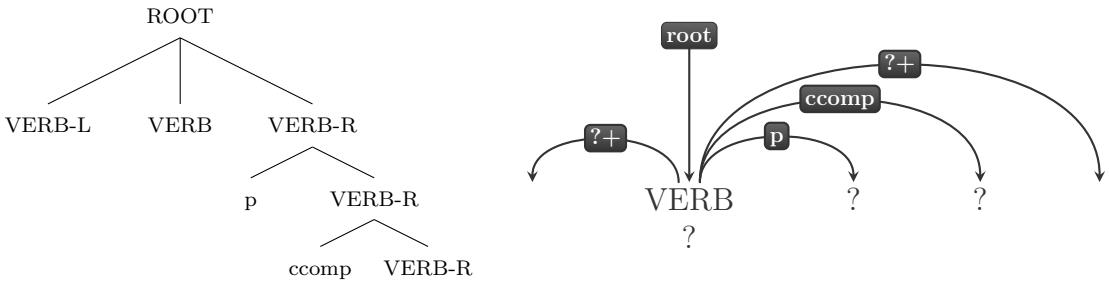
Figure 3.12: Creating test cases that consist of several sentences mediates feature sparsity, providing clear evidence for the discriminative power of the chosen feature set.

some cases due to noise, but it raises the concern that our features with $\mathcal{R}(F) < .66$ are also the result of noise in the data. To address this, we apply our features to the task of cross language NLI using only L1 data for training. If the variation of $\mathcal{R}(F)$ around chance is simply due to noise then we would expect near chance (33%) classification accuracy. The leftmost point in Figure 3.12 shows the initial result, using boolean features in a log-linear classification model, where a test case involves guessing an L1 label for each individual sentence in the L2 corpus. While the accuracy does exceed chance, the margin is not very large.

One possible explanation for this small margin is that the language transfer signal is sparse, as it is likely that language transfer can only be used to correctly label a subset of L2 data. We test this by combining randomly sampled L2 sentences with the same L1 label, as shown along the horizontal axis of Figure 3.12. As the number of sentences used to create each test case is increased, we see an increase in accuracy that supports the argument for sparsity; if the features were simply weak predictors, this curve would be flat. The resulting margin is much larger, providing evidence that a significant portion of our features with $\mathcal{R}(F) < .66$ are not selected due to random noise in $\mathcal{R}$ and are indeed connected to language transfer.

The number and strength of these hypotheses is easily augmented with more data, as is the number of languages under consideration and we encourage the reader to peruse the full list of results[2]. We will now discuss the top eight members of our list, demonstrating the final and only un-automated step in the formulation of a language transfer hypothesis using our method. Each discussion will appear on its own page, and includes the actual TSG fragment feature, and we have also provided its equivalent representation as a pattern matcher on dependency trees as they are normally represented.

[2] bllip.cs.brown.edu/download/interlanguage_corpus.pdf

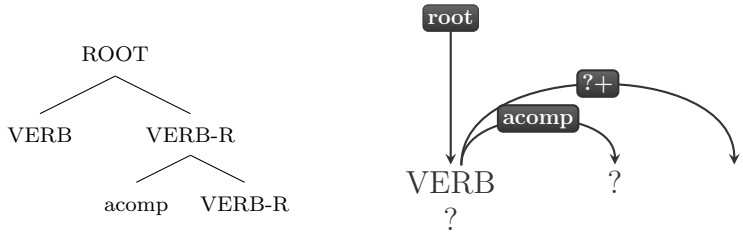


Counts per 1000 sentences

	ES	FR	**DE**	EN
L1	0.38	1.04	**8.94**	
L2	1.81	1.22	**8.37**	1.68

Expected Loss on L2 data - 0.331

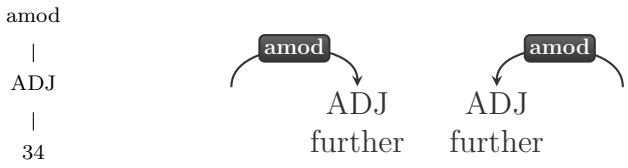
This pattern occurs when the main verb of a sentence is followed immediately with punctuation, usually a comma, and then a complementary clause. In English this leads to a sentence like “Now I know, that every step in life has its advantages.” or “I claim, that we do not taste a Big Mac in another way if we see an advertisement.”. This is a clear example of negative language transfer, as this phrasing is incorrect in English. The pattern is used most often in both German samples, and its native equivalent is correct in German, “Die Polizei glaubt, da etwa 30 Bomben in der Stadt sind.”



Counts per 1000 sentences				
	ES	FR	DE	EN
L1	**9.55**	.06	.19	
L2	**.72**	.06	.20	.34

Expected Loss on L2 data - 0.331

This pattern captures pronoun dropping at the beginning of a sentence, as in the actual L2 sentences “Is much more eficent to take the public bus.” and “Is possible that in 20 years, will not possible have a car.”. This pattern is associated with the Spanish speaking data samples, and appears in Spanish as “Es inofensivo para los humanos.” This is another example of negative language transfer. The L2 sentences provided also highlight the ability of the formalism to handle malformed learner input, remaining detectable despite the misspelling of ”efficient” in the first sentence and the additional pronoun drop in the second.

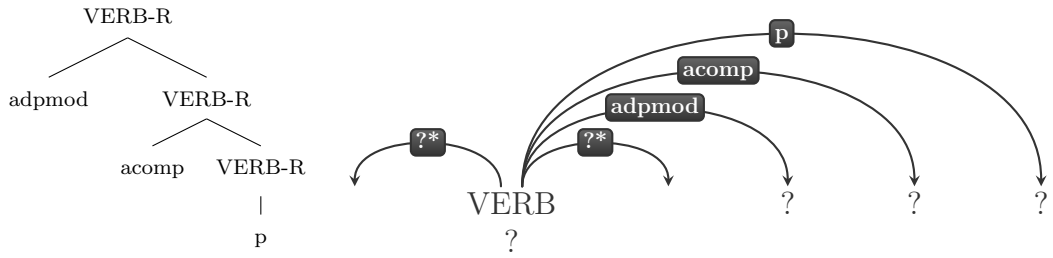


Counts per 1000 sentences

	ES	FR	**DE**	EN
L1	0.00	0.00	**7.24**	
L2	0.84	0.44	**2.24**	0.89

Expected Loss on L2 data - 0.332

A clear case of positive language transfer, this pattern simply detects the use of a certain multilingual stopword with index 34. Hypotheses containing such stopword classes are not as strong as they could be if our stopword lists were manually constructed, as we cannot be certain that the behavior is not an artifact of our clustering. The trend that German native speakers use the word "further" in English is clearly supported by the data, but we must perform some further straightforward analysis of our text samples to illuminate the cross-lingual connection.

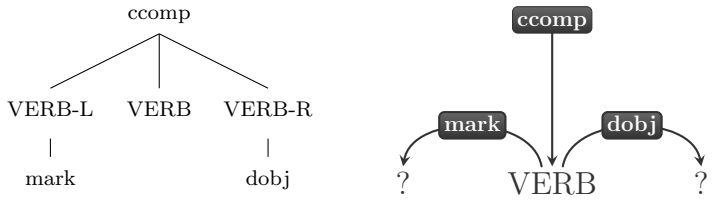


Counts per 1000 sentences

	ES	FR	**DE**	EN
L1	.50	.31	**10.8**	
L2	.13	.17	**.51**	.22

Expected Loss on L2 data - 0.374

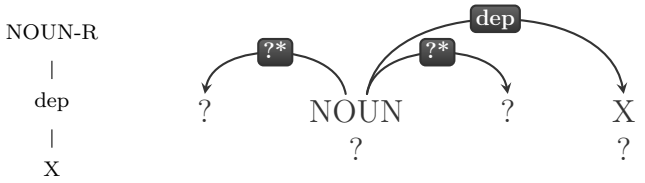
This pattern occurs at the end of sentences, as represented by the final punctuaion arc. The final two dependents of the main verb are an adpositional modifier followed by an adjectival compliment. In the sentence “From my point of view it is of course understandable.”, they are “of course”, and “understandable” respectively. In “Also the variety of ideas is in a group much bigger.”, they are “in a group” and “much bigger”. This is a technically correct but awkward in English, making it a case of positive language transfer that merits discussion in a classroom setting. The usage counts in L1 data suggest that this pattern is more natural in German, where the outcome is “Das Spiel war in Frankreich am weitesten verbreitet.”



Counts per 1000 sentences				
	ES	FR	DE	EN
L1	**2.39**	.06	0.00	
L2	**.91**	.33	0.10	.34

Expected Loss on L2 data - 0.394

This fragment represents a clausal complement beginning with a *mark* relationship, typically lexicalized by the word "that". This is followed by a single verb and a possibly complex direct object, resulting in a clauses such as "the facts *that determine it*" or "the causes *that argue this idea*". This is a distinction in the type of the clausal complement, differentiating from the common pattern used in "I know *that he ate it*" in which the clause has a subject as well. While this contrast is clear and points to a case of positive language transfer, in analysis of the learner sentences produced we find a large number of pattern matches to have arisen from misparsed learner text. To confidently claim that this particular linguistic pattern is associated with Spanish to English language transfer further data is required, although its prevalence in native Spanish is clearly shown in the first row in the table.

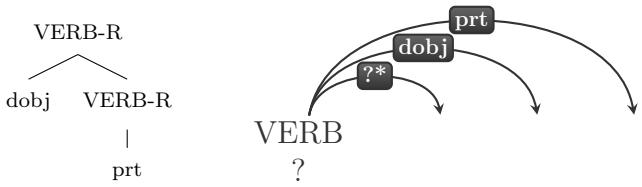


Counts per 1000 sentences

	ES	FR	DE	EN
L1	**11.05**	0.06	0.63	
L2	**3.38**	1.50	0.41	1.17

Expected Loss on L2 data - 0.407

This pattern is a false positive, and exemplifies not only the weakness of a data-driven system, but also the ease with which such patterns can be filtered away. The first red flag for this pattern is that besides the NOUN tag, it only uses the arc label "dep" and the pos tag "X", which are fallback labels in the UTB when other analyses do not fit. On observation of the english data, we see that it is trigged almost exclusively by things like "pollution, stress, etc.", where the dependency and X postag dominate the "etc". When triggered in the Spanish data, the dominated span is primarily English text "Britain's Got Talent" that does not fit the Spanish language model, or the exponential 2 in a phrase like "una superficie total de 92.98 km^2". It is pure coincidence that square meterage is often discussed in the native language data and "etc" is used more frequently in the Spanish L2 data, and that these are represented by the same TSG fragment.

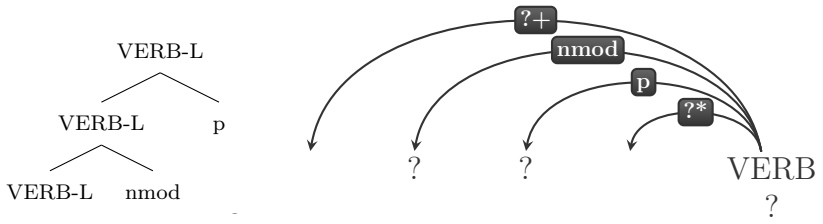


Counts per 1000 sentences

	ES	FR	**DE**	EN
L1	0.00	0.00	**4.72**	
L2	0.90	0.72	**2.09**	1.68

Expected Loss on L2 data - 0.407

Putting a verb particle after the direct object is perfectly fine in English, as in "try it out" or "cut their sparetime down" where the particles are "out" and "down". This is consistent with German seperable verbs in german, where a verb such as "abfahren" is used in a sentence as "Auf Gefahr fahre ich ab, aber angefahren fahre ich ins Krankenhaus ein." The use of seperable verbs makes this construction a natural result of direct translation from German to English by learners. This example of positive language transfer highlights the abilities of data driven systems, as they do not jump out as awkward or erroneous in L2 text and might be more easily missed by human linguists.



Counts per 1000 sentences

	ES	**FR**	DE	EN
L1	0.19	**1.84**	0.13	
L2	0.45	**1.22**	0.15	.62

Expected Loss on L2 data - 0.409

This pattern represents a certain way of starting sentences, preferred by French native speakers. The rightmost left dependent in the pattern is typically the verb's subject, and the punctuation arc usually a comma, and so the pattern is capturing the use of a nominal modifier before the subject of a sentence. In English this results in sentences such as "But today, many elders reache 100 years or over." or "Thus, in twenty year, we want to get things without any effort", where the nominal modifiers are "today" and "year". The corresponding pattern in French can be seen in the sentence "Ce matin, une bande d'anarcho-hitléro-trotskystes a voulu faire sauter". This is similar to the previous example in that it suggests positive language transfer that does not result in a awkward phrasing.

Chapter 4

Generating Language Education Exercises

4.1 Introduction

In our final section we investigate freeform data driven Natural Language Generation (NLG), with the goal of incorporating it in second language education classroom activities. While NLG is a natural choice for a domain that requires large amount of exemplar text, motivating its empirical study is a difficult task. The key issue is that although many language models used in statistical NLP are generative and can easily produce sample sentences by running their "generative mode", if all that is required is a plausible sentence one might as well pick a sentence at random from any existing corpus.

NLG becomes useful when constraints exist such that only certain sentences are valid. The majority of NLG applies a semantic constraint of "what to say", producing sentences with communicative goals. Examples include work such as Belz (2008), which produces weather reports from structured data, or Mitchell et al. (2013) which

generates descriptions of objects from images. Our work is more similar to NLG work that concentrates on structural constraints such as generative poetry (Greene et al. (2010))(Colton et al. (2012))(Jiang and Zhou (2008)) or song lyrics (Wu et al. (2013))(Ramakrishnan A et al. (2009)), where specified meter or rhyme schemes are enforced. In these papers soft semantic goals are sometimes also introduced that seek responses to previous lines of poetry or lyric.

Computational creativity is another subfield of NLG that often does not fix an a priori meaning in its output. Examples such as Özbal et al. (2013) and Valitutti et al. (2013) use template filling techniques guided by quantified notions of humor or how catchy a phrase is.

We study two constraints concerning the words that are allowed in a sentence. The first sets a fixed vocabulary such that only sentences where all words are in-vocab are allowed. The second demands not only that all words are in-vocab, but also requires the inclusion of a specific word somewhere in the sentence.

These constraints are natural in the construction of language education exercises, where students have small known vocabularies and exercises that reinforce the knowledge of arbitrary words are required. To provide an example, consider a Chinese teacher composing a quiz that asks students to translate sentences from English to Chinese. The teacher cannot ask students to translate words that have not been taught in class, and would like ensure that each vocabulary word from the current book chapter is included in at least one sentence. Using a system such as ours, she could easily generate a number of usable sentences that contain a given vocab word and select her favorite, repeating this process for each vocab word until the quiz is complete.

The construction of such a system presents two primary technical challenges. First, while highly parameterized models trained on large corpora are a good fit

for data driven NLG, sparsity is still an issue when constraints are introduced. Traditional smoothing techniques used for prediction based tasks are inappropriate, however, as they liberally assign probability to implausible text. We investigate smoothing techniques better suited for NLG that smooth more precisely, sharing probability only between words that have strong semantic connections.

The second challenge arises from the fact that both vocabulary and word inclusion constraints are easily handled with a rejection sampler that repeatedly generates sentences until one that obeys the constraints is produced. Unfortunately, for models with a sufficiently wide range of outputs the computation wasted by rejection quickly becomes prohibitive, especially when the word inclusion constraint is applied. We define models that sample directly from the possible outputs for each constraint without rejection or backtracking, and closely approximate the distribution of the true rejection samplers.

We contrast several generative systems through both human and automatic evaluation. Our best system effectively captures the compositional nature of our training data, producing error-free text with nearly 80 percent accuracy without wasting computation on backtracking or rejection. When the word inclusion constraint is introduced, we show clear empirical advantages over the simple solution of searching a large corpus for an appropriate sentence.

4.2 Freeform Generation

For clarity in our discussion, we phrase the sentence generation process in the following general terms based around two classes of atomic units : *contexts* and *outcomes*. In order to specify a generation system, we must define

1. the set $\mathcal{C}$ of contexts c

2. the set $\mathcal{O}$ of outcomes o
3. the "Imply" function $I(c, o) \rightarrow List[c \in \mathcal{C}]$
4. $\mathcal{M}$: derivation tree $\rightleftarrows$ sentence

where $I(c, o)$ defines the further contexts implied by the choice of outcome o for the context c. Beginning with a unique root context, a derivation tree is created by repeatedly choosing an outcome o for a leaf context c and expanding c to the new leaf contexts specified by $I(c, o)$. $\mathcal{M}$ converts between derivation tree and sentence text form.

This is simply a convenient rephrasing of the Context Free Grammar formalism, and as such the systems we describe all have some equivalent CFG interpretation. Indeed, to describe a traditional CFG, let $\mathcal{C}$ be the set of symbols, $\mathcal{O}$ be the rules of the CFG, and $I(c, o)$ return a list of the symbols on the right hand side of the rule o. To define an n-gram model, a context is a list of words, an outcome a single word, and $I(c, o)$ can be procedurally defined to drop the first element of c and append o.

To perform the sampling required for derivation tree construction we must define $P(o|c)$. Using $\mathcal{M}$, we begin by converting a large corpus of sentence segmented text into a training set of derivation trees. Maximum likelihood estimation of $P(o|c)$ is then as simple as normalizing the counts of the observed outcomes for each observed context. However, in order to obtain contexts for which the conditional independence assumption of $P(o|c)$ is appropriate, it is necessary to condition on a large amount of information. This leads to sparse estimates even on large amounts of training data, a problem that can be addressed by smoothing. We identify two complementary types of smoothing, and illustrate them with the following sentences.

The furry dog bit me.

The cute cat licked me.

An unsmoothed bigram model trained on this data can only generate the two sentences verbatim. If, however, we know that the tokens "dog" and "cat" are semantically similar, we can smooth by assuming the words that follow "cat" are also likely to follow "dog". This is easily handled with traditional smoothing techniques that interpolate between distributions estimated for both coarse, $P(w|w_{-1}$=[*animal*]), and fine, $P(w|w_{-1}$="dog"), contexts. We refer to this as *context smoothing*.

However, we would also like to capture the intuition that words which can be followed by "dog" can also be followed by "cat", which we will call *outcome smoothing*. We extend our terminology to describe a system that performs both types of smoothing with the following

- the set $\bar{\mathcal{C}}$ of smooth contexts $\bar{c}$
- the set $\bar{\mathcal{O}}$ of smooth outcomes $\bar{o}$
- a smoothing function $S_C : \mathcal{C} \rightarrow \bar{\mathcal{C}}$
- a smoothing function $S_O : \mathcal{O} \rightarrow \bar{\mathcal{O}}$

We describe the smoothed generative process with the flowchart shown in Figure 4.1. In order to choose an outcome for a given context, two decisions must be made. First, we must decide which context we will employ, the true context or the smooth context, marked by edges 1 or 2 respectively. Next, we choose to generate a true outcome or a smooth outcome, and if we select the latter we use edge 6 to choose a true outcome given the smooth outcome. The decision between edges 1 and 2 can be sampled from a Bernoulli random variable with parameter λ_c, with one variable estimated for each context c. The decision between edges 5 and 3 and the one between 4 and 7 can also be made with Bernoulli random variables, with parameter sets γ_c and $\gamma_{\bar{c}}$ respectively.

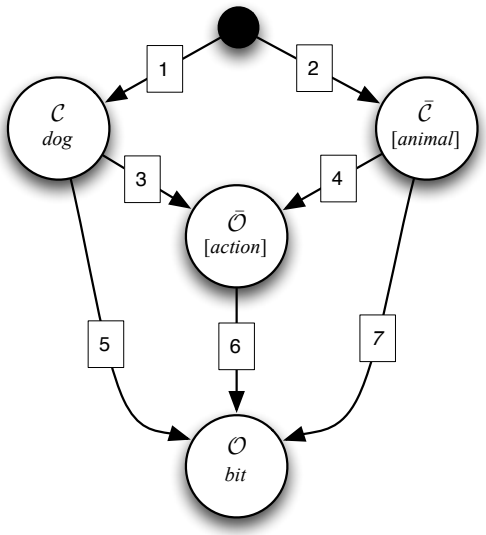


Figure 4.1: A flow chart depicting the decisions made when choosing an outcome for a context. The large circles show the set of items associated with each decision, and contain examples items for a bigram model where S_C and S_O map words (e.g. *dog*) to semantic classes (e.g. [*animal*]). The example items illustrate that there are four ways to decide that the "dog" should be followed by "bit". These four paths differ in the smoothness of their statements, ranging from "bit can follow dog" (path 1-5) to "dog is an animal, an animal can be followed by an action, and an action can be the word bit" (path 2-4-6).

This yields the full form of the unconstrained probabilistic generative model as follows

$$P(o|c) = \lambda_c P_1(o|c) + (1 - \lambda_c) P_2(o|S_c(c))$$

$$\begin{aligned} P_1(o|c) &= \gamma_c P_5(o|c) + \\ &\quad (1 - \gamma_c) P_7(o|\bar{o}) P_3(\bar{o}|c) \end{aligned} \tag{4.1}$$

$$\begin{aligned} P_2(o|\bar{c}) &= \gamma_{\bar{c}} P_6(o|c) + \\ &\quad (1 - \gamma_{\bar{c}}) P_7(o|\bar{o}) P_4(\bar{o}|\bar{c}) \end{aligned}$$

requiring estimation of the λ and γ variables as well as the five multinomial distributions P_{3-7}. This can be done with a straightforward application of EM.

4.3 Limiting Vocabulary

A primary concern in the generation of language education exercises is the working vocabulary of the students. If efficiency were not a concern, the natural solution to the vocabulary constraint would be rejection sampling: simply generate sentences until one happens to obey the constraint. In this section we show how to generate a sentence directly from this constrained set with a distribution closely approximating that of the rejection sampler.

4.3.1 Pruning

The first step is to prune the space of possible sentences to those that obey the vocabulary constraint. For the models we investigate there is a natural predicate $V(o)$ that is true if and only if an outcome introduces a word that is out of vocab, and so the vocabulary constraint is equivalent to the requirement that $V(o)$ is false for all possible outcomes o. Considering transitions along edges in Figure 4.1, the removal of all transitions along edges 5,6, and 7 that lead to outcomes where $V(o)$ is true satisfies this property.

Our remaining concern is that the generation process reaches a failure case. Again considering transitions in Figure 4.1, failure occurs when we require $P(o|c)$ for some c and there is no transition to c on edge 1 or $S_C(c)$ along edge 2. We refer to such a context as *invalid*. Our goal, which we refer to as *consistency*, is that for all valid contexts c, all outcomes o that can be reached in Figure 4.1 satisfy the property that all members of $I(c, o)$ are valid contexts.

To see how we might end up in failure, consider a trigram model on POS/word pairs for which S_C is the identity function and S_O backs off to the POS tag. Given a context $c = (\binom{t_{-2}}{w_{-2}}, \binom{t_{-1}}{w_{-1}})$ if we generate along a path using edge 6 we will choose a smooth outcome t_0 that we have seen following c in the data and then independenently choose a w_0 that has been observed with tag t_0. This implies a following context $(\binom{t_{-1}}{w_{-1}}, \binom{t_0}{w_0})$. If we have estimated our model with observations from data, there is no guarantee that this context ever appeared, and if so there will be no available transition along edges 1 or 2.

Let the list $\bar{I}(c, o)$ be the result of the mapped application of S_C to each element of $I(c, o)$. In order to define an efficient algorithm, we require the following property **D** referring to the amount of information needed to determine $\bar{I}(c, o)$. Simply put, **D** states if the smoothed context and outcome are fixed, then the implied smooth contexts are determined.

D $\{S_C(c), S_O(o)\} \rightarrow \bar{I}(c, o)$

To highlight the statement **D** makes, consider the trigram POS/word model described above, but let S_C also map the POS/word pairs in the context to their POS tags alone. **D** holds here because given $S_C(c) = (t_{-2}, t_{-1})$ and $S_O(o) = t_0$ from the outcome, we are able to determine the implied smooth context (t_{-1}, t_0). If context smoothing instead produced $S_C(c) = (t_{-2})$, **D** would not hold.

If **D** holds then we can show consistency based on the transitions in Figure 4.1 alone as any complete path through Figure 4.1 defines both $\bar{c}$ and $\bar{o}$. By **D** we can determine $\bar{I}(c, o)$ for any path and verify that all its members have possible transitions along edge 2. If the verification passes for all paths then the model is consistent.

Algorithm 1 produces a consistent model by verifying each complete path in the manner just described. One important feature is that it preserves the invariant that if a context c can be reached on edge 1, then $S_C(c)$ can be reached on edge 2. This

Algorithm 1 Pruning Algorithm

```
Initialize with all observed transitions
for all out of vocab o do
  remove ? → o from edges 5,6, and 7
end for
repeat
  for all paths in flow chart do
    if ∃c̄ ∈ Ī(c, o) s.t. c̄ is invalid then
      remove transition from edge 5,7,3 or 4
    end if
  end for
  Run FIXUP
until edge 2 transitions did not change
```

means that if the verification fails then the complete path produces an invalid context, even though we have only checked the members of $\bar{I}(c, o)$ against path 2.

If a complete path produces an invalid context, some transition along that path must be removed. It is never optimal to remove transitions from edges 1 or 2 as this unnecessarily removes all downstream complete paths as well, and so for invalid complete paths along 1-5 and 2-7 Algorithm 1 removes the transitions along edges 5 and 7. The choice is not so simple for the complete paths 1-3-6 and 2-4-6, as there are two remaining choices. Fortunately, **D** implies that breaking the connection on edge 3 or 4 is optimal as regardless of which outcome is chosen on edge 6, $\bar{I}(c, o)$ will still produce the same invalid $\bar{c}$.

After removing transitions in this manner, some transitions on edges 1-4 may no longer have any outgoing transitions. The subroutine FIXUP removes such transitions, checking edges 3 and 4 before 1 and 2. If FIXUP does not modify edge 2 then the model is consistent and Algorithm 1 terminates.

4.3.2 Estimation

In order to replicate the behavior of the rejection sampler, which uses the original probability model $P(o|c)$ from Equation 1, we must set the probabilities $P_V(o|c)$ of the pruned model appropriately. We note that for moderately sized vocabularies it is feasible to recursively enumerate $\mathcal{C}_V$, the set of all reachable contexts in the pruned model. In further discussion we simplify the representation of the model to a standard PCFG with $\mathcal{C}_V$ as its symbol set and its PCFG rules indexed by outcomes. This also allows us to construct the *reachability graph* for $\mathcal{C}_V$, with an edge from c_i to c_j for each $c_j \in I(c_i, o)$. Such an edge is given weight $P(o|c)$, the probability under the unconstrained model, and zero weight edges are not included.

Our goal is to retain the form of the standard incremental recursive sampling algorithm for PCFGs. The correctness of this algorithm comes from the fact that the probability of a rule R expanding a symbol X is precisely the probability of all trees rooted at X whose first rule is R. This implies that the correct sampling distribution is simply the distribution over rules itself. When constraints that disallow certain trees are introduced, the probability of all trees whose first rule is R only includes the mass from valid trees, and the correct sampling distribution is the renormalization of these values.

Let the *goodness* of a context $G(c)$ be the probability that a full subtree generated from c using the unconstrained model obeys the vocabulary constraint. Knowledge of $G(c)$ for all $c \in \mathcal{C}_V$ allows the calculation of probabilities for the pruned model with

$$P_V(o|c) \propto P(o|c) \prod_{c' \in I(c,o)} G(c') \tag{4.2}$$

While $G(c)$ can be defined recursively as

$$G(c) = \sum_{o \in \mathcal{O}} P(o|c) \prod_{c' \in I(c,o)} G(c') \tag{4.3}$$

its calculation requires that the reachability graph be acyclic. We approximate an acyclic graph by listing all edges in order of decreasing weight and introducing edges as long as they do not create cycles. This can be done efficiently with a binary search over the edges by weight. Note that this approximate graph is used only in recursive estimation of $G(c)$, and the true graph can still be used in Equation 4.2.

4.4 Generating Up

In this section we show how to efficiently generate sentences that contain an arbitrary word w^* in addition to the vocabulary constraint. We assume the ability to easily find $\mathcal{C}_{w^*}$, a subset of $\mathcal{C}_V$ whose use guarantees that the resulting sentence contains w^*. Our goal is once again to efficiently emulate the rejection sampler, which generates a derivation tree T and accepts if and only if it contains at least one member of $\mathcal{C}_{w^*}$.

Let $\mathcal{T}_{w^*}$ be the set of derivation trees that would be accepted by the rejection sampler. We present a three stage generative model and its associated probability distribution $P_{w^*}(\tau)$ over items τ for which there is a functional mapping into $\mathcal{T}_{w^*}$.

In addition to the probabilities $P_V(o|c)$ from the previous section, we require an estimate of $\mathbb{E}(c)$, the expected number of times each context c appears in a single tree. This can be computed efficiently using the mean matrix, described in Miller and Osullivan (1992). This $|\mathcal{C}_V| \times |\mathcal{C}_V|$ matri x M has its entries defined as

$$M(i,j) = \sum_{o \in \mathcal{O}} P(o|c_i) \#(c_j, c_i, o) \tag{4.4}$$

where the operator # returns the number of times context c_j appears $I(c_i, o)$. Defining a $1 \times |\mathcal{C}_V|$ start state vector z_0 that is zero everywhere and 1 in the entry corresponding to the root context gives

$$\mathbb{E}(z) = \sum_{i=0}^{\infty} z_0 M^i$$

which can be iteratively computed with sparse matrix multiplication. Note that the ith term in the sum corresponds to expected counts at depth i in the derivation tree. With definitions of context and outcome for which very deep derivations are improbable, it is reasonable to approximate this sum by truncation.

Our generation model operates in three phases.

1. Chose a start context $c_0 \in \mathcal{C}_{w^*}$
2. Generate a spine S of contexts and outcomes connecting c_0 to the root context
3. Fill in the full derivation tree T below all remaining unexpanded contexts

In the first phase, c_0 is sampled from the multinomial

$$P_1(c_0) = \frac{\mathbb{E}(c_0)}{\sum_{c \in \mathcal{C}_{w^*}} \mathbb{E}(c)} \tag{4.5}$$

The second step produces a spine S, which is formally an ordered list of triples. Each element of S records a context c_i, an outcome o_i, and the index k in $I(c_i, o_i)$ of the child along which the spine progresses. The members of S are sampled independantly given the previously sampled context, starting from c_0 and terminating when the root context is reached. Intuitively this is equivalent to generating the path from the root to c_0 in a bottom up fashion.

We define the probability P_σ of a triple (c_i, o_i, k) given a previously sampled context c_j as

$$P_\sigma(\{c_i, o_i, k\}|c_j) \propto \begin{cases} \mathbb{E}(c_i)P_V(o_i|c_i) & I(c_i, o_i)[k] = c_j \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Let $S = (c_1, o_1, k_1)\ldots(c_n, o_n, k_n)$ be the results of this recursive sampling algorithm, where c_n is the root context, and c_1 is the parent context of c_0. The total probability of a spine S is then

$$P_2(S|c_0) = \prod_{i=1}^{|S|} \frac{\mathbb{E}(c_i)P_V(o_i|c_i)}{Z_{i-1}} \tag{4.7}$$

$$Z_{i-1} = \sum_{(c,o)\in\mathbb{I}_{c_{i-1}}} \mathbb{E}(c)P_V(o|c)\#(c_{i-1}, c, o) \tag{4.8}$$

where $\mathbb{I}_{c-1}$ is the set of all (c, o) for which $P_\sigma(c, o, k|c_{i-1})$ is non-zero for some k. A key observation is that $Z_{i-1} = \mathbb{E}(c_{i-1})$, which cancels nearly all of the expected counts from the full product. Along with the fact that the expected count of the root context is one, the formula simplifies to

$$P_2(S|c_0) = \frac{\prod_{i=1}^{|S|} P_V(o_i|c_i)}{\mathbb{E}(c_0)} \tag{4.9}$$

The third step generates a final tree T by filling in subtrees below unexpanded contexts on the spine S using the original generation algorithm, yielding results with probability

$$P_3(T|S) = \prod_{(c,o)\in T/S} P_V(o|c) \tag{4.10}$$

where the set T/S includes all contexts that are not ancestors of c_0, as their outcomes are already specified in S.

We validate this algorithm by considering its distrubution over complete derivation trees $T \in \mathcal{T}_{w^*}$. The algorithm generates $\tau = (T, S, c_0)$ and has a simple functional mapping into $\mathcal{T}_{w^*}$ by extracting the first member of τ.

Combining the probabilities of our three steps gives

$$P_{w^*}(\tau) = \frac{\mathbb{E}(c_0)}{\sum_{c \in \mathcal{C}_{w^*}} \mathbb{E}(c)} \frac{\prod_{i=1}^{|S|} P_V(o_i|c_i)}{\mathbb{E}(c_0)} \prod_{(c,o) \in T/S} P_V(o|c)$$

$$P_{w^*}(\tau) = \frac{P_V(T)}{\sum_{c \in \mathcal{C}_{w^*}} \mathbb{E}(c)} = \frac{1}{\rho} P_V(T) \tag{4.11}$$

where ρ is a constant and

$$P_V(T) = \prod_{(c,o) \in T} P_V(o|c)$$

is the probability of T under the original model. Note that several τ may map to the same T by using different spines, and so

$$P_{w^*}(T) = \frac{\eta(T)}{\rho} P_V(T) \tag{4.12}$$

where $\eta(T)$ is the number of possible spines, or equivalently the number of contexts $c \in \mathcal{C}_{w^*}$ in T.

Recall that our goal is to efficiently emulate the output of a rejection sampler. An ideal system P_{w^*} would produce the complete set of derivation trees accepted by the rejection sampler using P_V, with probabilities of each derivation tree T satisfying

$$P_{w^*}(T) \propto P_V(T) \tag{4.13}$$

Consider the implications of the following assumption

A each $T \in \mathcal{T}_{w^*}$ contains exactly one $c \in \mathcal{C}_{w^*}$

A ensures that $\eta(T) = 1$ for all T, unifying Equations 4.12 and 4.13. **A** does not generally hold in practice, but its clear exposition allows us to design models for which it holds most of the time, leading to a tight approximation.

The most important consideration of this type is to limit redundancy in $\mathcal{C}_{w^*}$. For illustration consider a dependency grammar model with parent annotation where a context is the current word and its parent word. When specifying $\mathcal{C}_{w^*}$ for a particular w^*, we might choose all contexts in which w^* appears as either the current or parent word, but a better choice that more closely satisfies **A** is to choose contexts where w^* appears as the current word only.

4.5 Experiments

We train our models on sentences drawn from the Simple English Wikipedia[1]. We obtained these sentences from a data dump which we liberally filtered to remove items such as lists and sentences longer than 15 words or shorter then 3 words. We parsed this data with the recently updated Stanford Parser Socher et al. (2013) to Penn Treebank constituent form, and removed any sentence that did not parse to a top level S containing at least one NP and one VP child. Even with such strong filters, we retained over 140K sentences for use as training data, and provide this exact set of parse trees for use in future work.[2]

Inspired by the application in language education, for our vocabulary list we use the English Vocabulary Profile (Capel (2012)), which predicts student vocabulary at different stages of learning English as a second language. We take the the most basic

[1] http://simple.wikipedia.org
[2] data url anon for review

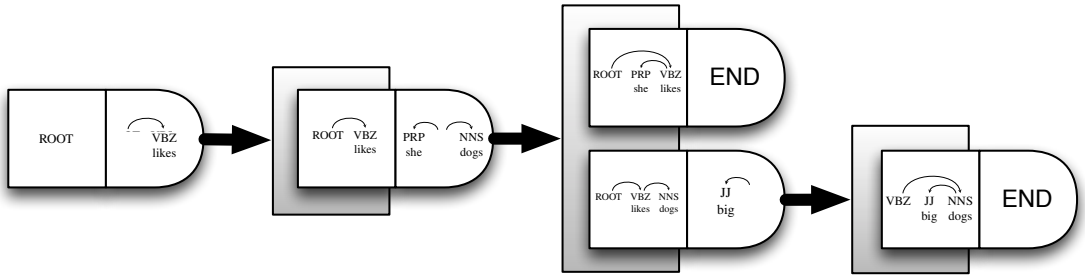


Figure 4.2: The generation system SpineDep draws on dependency tree syntax where we use the term *node* to refer to a POS/word pair. Contexts consist of a node, its parent node, and grandparent POS tag, as shown in squares. Outcomes, shown in squares with rounded right sides, are full lists of dependents or the END symbol. The shaded rectangles contain the results of $I(c, o)$ from the indicated (c, o) pair.

American English vocabulary (the A1 list), and retrieve all inflections for each word using SimpleNLG (Gatt and Reiter (2009)), yielding a vocabulary of 1226 simple words and punctuation.

To mitigate noise in the data, we discard any pair of context and outcome that appears only once in the training data, and estimate the parameters of the unconstrained model using EM.

4.5.1 Model Comparison

We experimented with many generation models before converging on SpineDep, described in Figure 4.5.1, which we use in these experiments. SpineDep uses dependency grammar elements, with parent and grandparent information in the contexts to capture such distinctions as that between main and clausal verbs. Its outcomes are full configurations of dependents, capturing coordinations such as subject-object pairings. This specificity greatly increases the size of the model and in turn reduces the speed of the true rejection sampler, which fails over 90% of the time to produce an in-vocab sentence.

	Corr(%)	% uniq
SpineDep unsmoothed	87.6	5.0
SpineDep WordNet	78.3	32.5
SpineDep word2vec 5000	72.6	52.9
SpineDep word2vec 500	65.3	60.2
KneserNey-5	64.0	25.8
DMV	33.7	71.2

Figure 4.3: System comparison based on human judged correctness and the percentage of unique sentences in a sample of 100K.

We found that large amounts of smoothing quickly diminishes the amount of error free output, and so we smooth very cautiously, mapping words in the contexts and outcomes to fine semantic classes. We compare the use of human annotated hypernyms from Wordnet (Miller (1995)) with automatic word clusters from word2vec (Mikolov et al. (2013)), based on vector space word embeddings, evaluating both 500 and 5000 clusters for the latter.

We compare these models against several baseline alternatives, shown in Figure 4.3. To determine correctness, used Amazon Mechanical Turk, asking the question: "Is this sentence plausible?". We further clarified this question in the instructions with alternative definitions of plausibility as well as both positive and negative examples. Every sentence was rated by five reviewers and its correctness was determined by majority vote, with a .496 Fleiss kappa agreement. To avoid spammers, we limited our hits to Turkers with an over 95% approval rating.

Traditional language modeling techniques such as such as the Dependency Model with Valence (Klein and Manning (2004)) and 5-gram Kneser Ney (Chen and Goodman (1996)) perform poorly, which is unsurprising as they are designed for tasks in recognition rather than generation. For n-gram models, accuracy can be greatly increased by decreasing the amount of smoothing, but it becomes difficult to find long n-grams that are completely in-vocab and results become redundant, parroting the

	Corr(%)	-LLR
True RS	79.3	–
Uniform	47.3	96.2
$G(c) = 1$	77.0	25.0
$G(c)$ estimated	**78.3**	**1.0**

Figure 4.4: A comparison of our system against both a weak and a strong baseline based on correctness and the negative log of the likelihood ratio measuring closeness to the true rejection sampler.

few completely in-vocab sentences from the training data. The DMV is more flexible, but makes assumptions of conditional independence that are far too strong. As a result it is unable to avoid red flags such as sentences not ending in punctuation or strange subject-object coordinations. Without smoothing, SpineDep suffers from a similar problem as unsmoothed n-gram models; high accuracy but quickly vanishing productivity.

All of the smoothed SpineDep systems show clear advantages over their competitors. The tradeoff between correctness and generative capacity is also clear, and our results suggest that the number of clusters created from the word2vec embeddings can be used to trace this curve. As for the ideal position in this tradeoff, we leave such decisions which are particular to specific application to future work, arbitrarily using SpineDep WordNet for our following experiments.

4.5.2 Fixed Vocabulary

To show the tightness of the approximation presented in Section 4.2, we evaluate three settings for the probabilities of the pruned model. The first is a weak baseline that sets all distributions to uniform. For the second, we simply renormalize the true model's probabilities, which is equivalent to setting $G(c) = 1$ for all c in Equation 4.2. Finally, we use our proposed method to estimate $G(c)$.

We show in Figure 4.4 that our estimation method more closely approximates the distribution of the rejection sampler by drawing 500K samples from each model and comparing them with 500K samples from the rejection sampler itself. We quantify this comparison with the likelihood ratio statistic, evaluating the null hypothesis that the two samples were drawn from the same distribution. Not only does our method more closely emulate that of the rejection sampler, be we see welcome evidence that closeness to the true distribution is correlated with correctness.

4.5.3 Word Inclusion

To explore the word inclusion constraint, for each word in our vocabulary list we sample 1000 sentences that are constrained to include that word using both unsmoothed and WordNet smoothed SpineDep. We compare these results to the "Corpus" model that simply searches the training data and uniformly samples from the existing sentences that satisfy the constraints. This corpus search approach is quite a strong baseline, as it is trivial to implement and we assume perfect correctness for its results.

This experiment is especially relevant to our motivation of language education. The natural question when proposing any NLG approach is whether or not the ability to automatically produce sentences outweighs the requirement of a post-process to ensure goal-appropriate output. This is a challenging task in the context of language education, as most applications such as exam or homework creation require only a handful of sentences. In order for an NLG solution to be appropriate, the constraints must be so strong that a corpus search based method will frequently produce too few options to be useful. The word inclusion constraint highlights the strengths of our method as it is not only highly plausible in a language education setting but difficult to satisfy by chance in large corpora.

	# < 10	# > 100	Corr(%)
Corpus	987	26	100
Unsmooth	957	56	89.0
Smooth	544	586	79.0

Figure 4.5: Using systems that implement the word inclusion constraint, we count the words for which the number of unique sentences out of 1000 samples was less than 10 or greater than 100, along with the correctness of each system.

We make the assumption that for a user to effectively select a desirable output there should be at least ten options, as some of the outputs may be grammatically or semantically incorrect or undesirable for other unquantfiable reasons. We also assume that the ability to produce over 100 possibilities is a good benchmark for producing a wide variety of outputs. Figure 4.5 shows that the corpus search approach fails to find more than ten sentences that obey the word inclusion constraints for most target words. Moreover, it is arguably the case that unsmoothed SpineDep is even worse due to its inferior correctness. With the addition of smoothing, however, we see a drastic shift in the number of words for which a large number of sentences can be produced. For the majority of the vocabulary words this model generates over 100 sentences that obey both constraints, of which approximately 80% are valid English sentences.

Chapter 5

Conclusion

Learning a second language is a increasingly global experience, with English spoken as a second language by a large amount of the world. At the same time, the increased use and acceptence of technology in work, school, and daily life is hard to ignore. This research follows a long standing interest in the interaction of these two growing trends that is shared by previous work in Natural Language Processing, Computationally Assisted Language Learning, and Linguistics.

Throughout this research we have held this motivation in focus and developed novel techniques for the discovery of language transfer hypotheses and vocabulary constrained generation. Our methods for producing a ranked and filtered list of candidate hypotheses is purely data-driven and will scale with the ongoing efforts to collect the data it requires. Our generation system produces sentences that are constrained by vocabulary or word inclusion and semantically plausible with high probability. Together, they greatly reduce the workload and expertise required to incorporate language transfer hypotheses into the second language classroom, which has been shown to increase the quality of the students' education.

First, we discuss what remains between the ideas presented here and some detailed

possibilities for real world applications. We then discuss ideas for future work regarding both extension of our methods and pursuit of the general goal of computationally assisted second language learning.

Completing the Connection

First we consider scenarios where our list of language transfer hypotheses is incorporated into language education curriculum. It is often the case with English as a Second Language (ESL) education that the school is devoted entirely to English education, and has an administrative staff whose task is to determine a school-wide syllabus for classes of varying levels of experience. We propose that a person in such a role would be able to use our methods and judge the quality of each language transfer hypothesis. Often observing ten to twenty sentences that contain the pattern is enough to understand it, a task made easier by bracketing or highlighting the text that is dominated by the TSG rule representing the pattern. Determining if the pattern is an example of negative language transfer is simple given presumed fluency in the language of instruction, although it would be interesting to explore automatically determining this property. The biggest weakness of this proposed system is that in order to fully convince oneself of the integrity of a pattern it is necessary to understand examples from the native language of the transfer phenomena as well as the second, requiring a bilingual staff member for each native language involved.

Once a language transfer hypotheses has been accepted for inclusion in the syllabus, it can be used in many ways. The most straightforward is to simply have the teacher explicitly highlight the form resulting from language transfer and present corresponding sentences in the students' native languages. In the case of negative language transfer, L2 sentences exhibiting the error can be used in common exam

questions such as "Which of these sentences are bad English?" or "Correct this sentence". Negative transfer patterns could also be used in translation exercises, both from and to the L2. Positive language transfer can be included in the curriculum by focusing on equivalent grammatical forms or synonyms, encouraging the students to broaden their familiarities. Also, in the case of cloze questions ("fill in the blank"), if a negative language transfer phenomenon consists of a single word or phrase then its inclusion as a selection option is a worthwhile distractor to consider.

Most of these strategies involve example sentences in either the students' L1 or L2, which traditionally are constructed by a textbook author or teacher themselves. Our generative system of exemplar text can easily produce such sentences. Additionally, our word inclusion constraint enables the construction of exercises with multiple questions such that one of a chosen set of featured vocabulary is featured in each question. While according to our human evaluation above these sentences are frequently correct, a human verification step would be necessary not only to weed out incorrect sentences but to apply the teacher's unquantifiable constraints of appropriateness. It is important future work to evaluate generative language technologies in the formulation of exercises to determine if they make tasks like exam creation easier. It would also be interesting to use the output that the teacher labels as incorrect as a source of "correct this sentence" exercises, as they may represent more probable real life errors.

The automatic adaptability of our generative model to any vocabulary allows the use of a dynamic syllabus. It is common practice in ESL schools to have a class textbook that provides vocabulary lists, chapter by chapter. Due to the interests or professional needs of the students, the sequence of chapters in these textbooks is often only loosely respected, skipping chapters or studying them in a modified order. This results in pre-made textbook exercises that feature words or grammar presented

in a previous book chapter that has not been covered by the students. A generated exercise does not suffer from this weakness, as the vocabulary constraint can simply be altered to represent the vocabulary from the chapters that have been studied.

Additionally, in most language classes the interests of the students during in-class activities prompt the introduction of new words that do not appear in the textbook. These words do not appear in the textbook and are not reinforced by pre-made exams and exercises. With our system, as long as the teacher quickly recorded these words they could be automatically integrated into generated exams, homeworks, and activities, allowing their reinforcement and retention.

The final step that deserves discussion is the missing link between the two tasks that we address in this work: given our representation of a language transfer hypotheses, how does one define a system of contexts and outcomes to provide exemplar text? It is a happy coincidence that both our representation of language transfer patterns and the contexts and outcomes of our best performing generation system, SpineDep, are isomorphic to dependency tree syntax. This makes it a simple task to generate exemplars with a slight modification of the word inclusion constraint, which begins with a single context, generates a spine to the root, and then fills in remaining leaves. To force inclusion of a language transfer pattern this process can be seeded with a configuration of contexts and outcomes that represent the pattern, which will be a tree with a root context. A spine can be generated from this root context, and then the leaves filled as before.

Next Steps

While we have managed to automatically complete many steps in the formulation of language transfer hypotheses, our methods are far from perfect. Perhaps the biggest weakness is the set of universal stopword mappings ($\mathcal{M}_L$) used in Section

3.4.2. Some clusters do correspond nicely to the translations across the four languages (English, French, German, and Spanish) that we investigate, in many cases capturing the inflections of the gendered languages. Still, we found it hard to avoid one very large cluster that contains a large number of unrelated stopwords. This should not effect the quality of the output, as this can be thought of as an "other" category, but in this way we lose the chance to capture many potentially interesting patterns. It would be a fairly simple task to curate a gold set of mappings for a small number of languages, which would allow principled experimentation with automatic methods such as ours and quantifiable comparison between systems.

Another understudied topic that has significant impact on our results is the behavior of statistical parsers on non-native text. This involves handling such artifacts as frequent misspellings of words, dropping of pronouns and articles, and irregular punctuation, and it is not clear how to best proceed. By using a parser trained on the Penn Treebank directly, we are trusting that misspelled words are mapped to appropriate unknown word categories so their part of speech can be correctly determined. Also we hope that to some extent it does not matter what some ungrammatical phrase parses to, as long as the same error leads to the same structure across trees. We would be very interested in future work on this topic, as would the general field of NLI.

Generation of plausible text without a semantic goal is a field where much work remains. A particular extension to our work would be to develop methods to investigate the rate of plausibility over several vocabulary lists and improve the plausibility rate across the board. One technique for this is active learning, where the user would be asked to simply give or deny approval to generated text, which would then feed back into the system. This brings to mind early machine learning work in grammatical version spaces, which has much to say about such grammar induction algorithms. If it could be shown that through a small amount of effort the plausibility could be

significantly increased, this would fit our paradigm of language education in a ESL company setting perfectly, where a teacher is a member of the process whose time commitment should be minimized.

There are also possibilities for the use of the algorithms we develop on other NLP tasks. The multi-grammar Tree Substitution Grammar induction model that we present is general purpose in nature, and can produce many useful models by appropriate parameter settings, such as a naive Bayes model or Latent Dirichlet Allocation topic model in either its canonical or supervised form. While in many implementations of these models the language model component is a simple bag of words, our code allows the use of the TSG as the language model and it would be interesting to see if the addition of syntactic information aids any of the vast number of experiments that use these well-known models. Our work with constrained generative grammars also has application in other areas, as it fits the general scenario where composition of correct text is required but the communicative goal is not explicit. This includes the previous work mentioned above in poetry and lyric generation, but also has potential in research tasks like abstractive captioning of images. Another potential area of outside application is in speech recognition, which relies on good language models. It is possible that learner language models that condition on the speakers native language would produce better L2 language models that more accurately represent the speaker. To facilitate such future work, we release several software packages that make up the bulk of the work described here, with links are provided in the introduction.

The General Future

In our experiments, we have focused on a specific aspect (language transfer) of a much larger group of interconnected issues regarding second language acquisition.

When detected across multiple languages, language transfer becomes relevant to questions regarding the effects of native language on cognition such as the Sapir-Whorf hypothesis. It is possible that when viewed on the level playing field of a common second language and essay prompt, the behavior of different L1 groups is made clearer or more easy quantified.

Computationally Assisted Language Learning (CALL) will no doubt remain an excellent real world application for NLP, and its increasing prevalence will grow the set of interesting problems that exists today. A key enabler for this type of research would be to foster the partnership between the educators who hold the relevant data and the academic community that wants to analyze it. While private software companies may be unwilling to part with such data, certainly the second language education departments of major universities could design a framework for sharing data with the cooperation of interested NLP groups. A large step in the right direction would be a shift in language education testing to fully computer based methods, a trend that is already underway with the recent success of online education services such as Coursera.

While CALL is primarily focused on techniques that effect student performance on exams, NLP may find more common ground in the design of the syllabus. These questions are important, as they determine what should be taught rather than how it should be presented. For example, it is worth considering the design of vocabulary lists in the chapters of a textbook. Not only should the individual vocabulary lists be conceptually related such that several vocab words can naturally appear in a single sentence, but successive lists should also exhibit this property to some extent. This allows the continued reinforcement of previous lists in natural text. Also, it can be beneficial to dynamically incorporate current events or class interests into a syllabus, but careful analysis is required to determine if the new text is appropriate to the

knowledge of the students.

The generation of classroom exercises is also a field where much work can be done. Our incorporation of the vocabulary constraint is a bare minimum for such systems, and our word inclusion constraint facilitates many natural language education exercises, but many other types of exercise exist. One closely related topic is the generation of sentences that feature some syntax or word collocation, which is possible through extension of our model. A more complex extension would be to incorporate dialog modeling, attempting to produce the simple dialog that are ubiquitous in language education textbooks. This much more complex problem would be approachable in a similar spirit to our work, using large amount of in and out of vocab dialog to determine its compositional nature and building a vocabulary constrained generation engine from these components.

Bibliography

Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering 14*(4), 431–455.

Blanchard, D., J. Tetreault, D. Higgins, A. Cahill, and M. Chodorow (2013). Toefl11: A corpus of non-native english. Technical report, Educational Testing Service.

Blunsom, P. and T. Cohn (2010). Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*, pp. 1204–1213.

Blunsom, P., T. Cohn, S. Goldwater, and M. Johnson (2009, August). A note on the implementation of hierarchical dirichlet processes. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, Suntec, Singapore, pp. 337–340. Association for Computational Linguistics.

Bod, R. (1992). A computational model of language performance: Data oriented parsing. In *COLING*, pp. 855–859.

Brooke, J. and G. Hirst (2011). Native language detection with 'cheap' learner corpora. In *Conference of Learner Corpus Research (LCR2011)*, Louvain-la-Neuve, Belgium. Presses universitaires de Louvain.

Brooke, J. and G. Hirst (2012, December). Robust, Lexicalized Native Language

Identification. In *Proceedings of COLING 2012*, Mumbai, India, pp. 391–408. The COLING 2012 Organizing Committee.

Capel, A. (2012). The english vocabulary profile. *http://vocabulary.englishprofile.org/*.

Carroll, G. and E. Charniak (1992). Two experiments on learning probabilistic dependency grammars from corpora. Technical Report CS-92-16, Brown University, Providence, RI, USA.

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

Cohn, T. and P. Blunsom (2010a). Blocked inference in bayesian tree substitution grammars. In *ACL (Short Papers)*, pp. 225–230.

Cohn, T. and P. Blunsom (2010b). Blocked inference in bayesian tree substitution grammars. pp. 225–230. Association for Computational Linguistics.

Colton, S., J. Goodwin, and T. Veale (2012). Full face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, pp. 95–102.

Gatt, A. and E. Reiter (2009). Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, Stroudsburg, PA, USA, pp. 90–93. Association for Computational Linguistics.

Goodman, J. (1999, December). Semiring parsing. *Comput. Linguist. 25*(4), 573–605.

Granger, S., E. Dagneaux, and F. Meunier (2002). *International Corpus of Learner English : Version 1.1 ; Handbook and CD-ROM*. Louvain-la-Neuve: Pr. Univ. de Louvain.

Greene, E., T. Bodrumlu, and K. Knight (2010). Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, Stroudsburg, PA, USA, pp. 524–533. Association for Computational Linguistics.

Jarvis, S., Y. Bestgen, S. A. Crossley, S. Granger, M. Paquot, J. Thewissen, and D. McNamara (2012). The Comparative and Combined Contributions of n-Grams, Coh-Metrix Indices and Error Types in the L1 Classification of Learner Texts. In S. Jarvis and S. A. Crosley (Eds.), *Approaching Language Transfer through Text Classification*, pp. 154–177. Multilingual Matters.

Jarvis, S. and S. Crossley (Eds.) (2012). *Approaching Language Transfer Through Text Classification: Explorations in the Detection-based Approach*, Volume 64. Bristol, UK: Multilingual Matters Limited.

Jiang, L. and M. Zhou (2008). Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pp. 377–384. Association for Computational Linguistics.

Johnson, M. (2011). How relevant is linguistics to computational linguistics?. *Linguistic Issues in Language Technology*.

Klein, D. and C. D. Manning (2003). Accurate unlexicalized parsing. In *ACL*, pp. 423–430.

Klein, D. and C. D. Manning (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42Nd Annual Meeting*

on Association for Computational Linguistics, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. *MT Summit*.

Koppel, M., J. Schler, and K. Zigdon (2005). Automatically determining an anonymous author's native language. *Intelligence and Security Informatics*, 41–76.

Lado, R. (1957). *Linguistics across cultures: applied linguistics for language teachers*. University of Michigan Press.

Laufer, B. and N. Girsai (2008). Form-focused instruction in second language vocabulary learning: A case for contrastive analysis and translation. *Applied Linguistics 29*(4), 694–716.

Marneffe, M. C. D., B. Maccartney, and C. D. Manning (2006). Generating typed dependency parses from phrase structure parses. In *In Proc. Intl Conf. on Language Resources and Evaluation (LREC*, pp. 449–454.

McDonald, R. T., J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Täckström, C. Bedini, N. B. Castelló, and J. Lee (2013). Universal dependency annotation for multilingual parsing. In *ACL (2)*, pp. 92–97.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *CoRR abs/1301.3781*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM 38*, 39–41.

Miller, M. I. and J. A. Osullivan (1992). Entropies and combinatorics of random branching processes and context-free languages. *IEEE Transactions on Information Theory 38*.

Mimno, D. M. and A. McCallum (2012). Topic models conditioned on arbitrary features with dirichlet-multinomial regression. *CoRR abs/1206.3278*.

Mitchell, M., K. van Deemter, and E. Reiter (2013). Generating expressions that refer to visible objects. In *HLT-NAACL*, pp. 1174–1184.

Mostow, J. and H. Jang (2012). Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pp. 136–146. Association for Computational Linguistics.

Murphy, S. (2005). Second language transfer during third language acquisition. *Teachers College, Columbia University Working Papers in TESOL & Applied Linguistics 3*(1).

Och, F. J. and H. Ney (2003, March). A systematic comparison of various statistical alignment models. *Comput. Linguist. 29*(1), 19–51.

Özbal, G., D. Pighin, and C. Strapparava (2013, August). Brainsup: Brainstorming support for creative sentence generation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, pp. 1446–1455. Association for Computational Linguistics.

Peng, H., F. Long, and C. Ding (2005, August). Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell. 27*(8), 1226–1238.

Petrov, S., L. Barrett, R. Thibaux, and D. Klein (2006, July). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International*

Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, pp. 433–440. Association for Computational Linguistics.

Post, M. and D. Gildea (2009). Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pp. 45–48. Association for Computational Linguistics.

Ramakrishnan A, A., S. Kuppan, and S. L. Devi (2009). Automatic generation of tamil lyrics for melodies. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pp. 40–46. Association for Computational Linguistics.

Sangati, F. and W. Zuidema (2011, July). Accurate parsing with compact tree-substitution grammars: Double-dop. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK., pp. 84–95. Association for Computational Linguistics.

Shindo, H., Y. Miyao, A. Fujino, and M. Nagata (2012). Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *ACL (1)*, pp. 440–448.

Socher, R., J. Bauer, C. D. Manning, and A. Y. Ng (2013). Parsing With Compositional Vector Grammars. In *ACL*.

Sumita, E., F. Sugaya, and S. Yamamoto (2005). Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, pp. 61–68. Association for Computational Linguistics.

Swanson, B. and E. Charniak (2013, June). Extracting the native language signal for second language acquisition. In *Proceedings of the 2013 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, Georgia, pp. 85–94. Association for Computational Linguistics.

Teh, Y. W., M. I. Jordan, M. J. Beal, and D. M. Blei (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association 101*(476), 1566–1581.

Tetreault, J., D. Blanchard, and A. Cahill (2013, June). A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, GA, USA. Association for Computational Linguistics.

Tono, Y., Y. Kawaguchi, and M. Minegishi (Eds.) (2012). *Developmental and Crosslinguistic Perspectives in Learner Corpus Research*. Tokyo University of Foreign Studies 4. Philadelphia: John Benjamins.

Tsur, O. and A. Rappoport (2007, June). Using Classifier Features for Studying the Effect of Native Language on the Choice of Written Second Language Words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, Prague, Czech Republic, pp. 9–16. Association for Computational Linguistics.

Tuv, E., A. Borisov, G. Runger, K. Torkkola, I. Guyon, and A. R. Saffari (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. *JMLR*.

Valitutti, A., H. Toivonen, A. Doucet, and J. M. Toivanen (2013). "let everything turn well in your wife": Generation of adult humor using lexical constraints. In *ACL (2)*, pp. 243–248.

Wong, S.-M. J. and M. Dras (2009, December). Contrastive Analysis and Native Language Identification. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, Sydney, Australia, pp. 53–61.

Wong, S.-M. J. and M. Dras (2011, July). Exploiting Parse Structures for Native Language Identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, UK., pp. 1600–1610. Association for Computational Linguistics.

Wong, S.-M. J., M. Dras, and M. Johnson (2011, December). Topic Modeling for Native Language Identification. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, Canberra, Australia, pp. 115–124.

Wu, D., K. Addanki, M. Saers, and M. Beloucif (2013). Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation. In *EMNLP*, pp. 102–112.

Yamangil, E. and S. M. Shieber (2010). Bayesian synchronous tree-substitution grammar induction and its application to sentence compression. In *ACL*, pp. 937–947.

Yu, L. and H. Liu (2004, December). Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res. 5*, 1205–1224.