Unsupervised Word Segmentation: An Investigation of Sub-word Features*

Daniel Blanchard

March 20, 2011

*The author thanks Amy Blanchard for all her help proofreading this document.

Contents

1 Introduction

Word segmentation is the problem of separating a given utterance into words. In some written languages (e.g., English), this is a trivial task, as there are clear boundaries marked between words; however, this is not always the case. Some widely-used languages (e.g., Chinese and Japanese) do not mark word boundaries when written, and in speech there are no reliable boundaries between words (Cole and Jakimik, 1980). Thus, in these types of languages segmenting words is crucial to understanding what an utterance means.

To illustrate how word segmentation works in languages that do not mark boundaries, we consider the unsegmented English utterance “Therearenospaces.” How would one find the boundaries in such an utterance? The simplest technique is to search for familiar words. Starting from the beginning of the utterance, “The” and “There” are the two common words that could start it, but only one of them leads to a segmentation exclusively containing English words. If we choose “The,” the next possible word is “rear,” but “enospaces” cannot be split into familiar English words. Therefore, the utterance must start with “There” after which “arenospaces” can only be broken into “are no spaces.” The key to this simple method of segmentation is that we already have a lexicon that contains all the words that the utterance could consist of.

In contrast to the previous example, this research addresses the word segmentation problem with no *a priori* lexicon. This problem is inspired by infants, who learn to segment the ambient language with no lexicon; amazingly, they begin to segment speech by six-months of age (Bortfeld et al., 2005). Since infants learn to segment speech with great efficiency and accuracy despite not initially having a lexicon, I am interested in computational models of word segmentation that utilize insights from language acquisition research.

While infants are not born with a lexicon of the language they will be learning, there are many patterns present in natural language that they can use to segment speech. This is because languages all have their own sets of phonotactic constraints, which restrict the sequences allowed in well-formed words (Chomsky and Halle, 1965; Halle, 1978). When infants hear sequences they know to be invalid, they can then posit word boundaries in places that prevent them from occurring. For example, the velar nasal [ŋ] (e.g., “sing” [sɪŋ]) cannot occur following [t] in English. Thus, when nine-month-olds are presented with novel phrases such as “fang tine” [faŋ taɪn], they correctly insert a boundary between the two words (Mattys and Jusczyk, 2001). In addition to these phoneme pairs, infants have also been shown to use their knowledge of allophonic variation (Jusczyk et al., 1999a), stress patterns (Jusczyk et al., 1999b), and syllable transition probabilities (i.e., syllable bigrams) (Saffran et al., 1996) to successfully segment speech.

In the word segmentation literature (e.g., Brent, 1999; Venkataraman, 2001; Goldwater, 2007; Johnson, 2008a), word segmentation systems that lack both a lexicon of the language they will be learning and feedback about when utterances have been segmented correctly are called “unsupervised.”[1] They usually operate on unsegmented phonetic transcriptions of speech (often infant-

[1]While in traditional machine learning literature “unsupervised” algorithms are only required to not have any

directed), and can be considered abstract models of how infants learn to segment speech. Yet, despite the similarities between both the input given to and restrictions faced by unsupervised segmenters and infants, most unsupervised segmenters do not utilize any of the types of sub-word (i.e., word-internal) patterns known to be useful to infants when segmenting speech. It is the primary goal of my research to remedy this oversight and determine in principle what sub-word features are most effective for unsupervised word segmentation algorithms.

1.1 Contributions

In the process of investigating which sub-word features are most useful for unsupervised word segmenters, my dissertation will make five major contributions to the field: the investigation itself, a unified framework for segmentation, evidence that evaluating segmenters with respect to orthographic words is incorrect, new methods for preventing and recovering from early errors, and a highly accurate unsupervised incremental word segmenter. Each of these contributions are discussed in more detail below.

1.1.1 Investigation of Sub-word Features

The main contribution of my dissertation will be establishing in principle what sub-word features are more useful than others, and the types of errors each prevents or causes. This will be particularly useful for anyone who is building an unsupervised word segmentation system, as they will know which features they should use to address particular issues with a given corpus or model. Furthermore, my work will also be the first to examine some sub-word features like long-distance phonotactic patterns, which may turn out to be beneficial for segmenting languages that have vowel harmony (e.g., Finnish). Finally, if syllable-based sub-word features (e.g., syllable n-grams) are especially useful, then my work could also be considered evidence for the necessity of the syllable, a unit whose status is often debated by phonologists.

1.1.2 Framework for Word Segmentation

The framework presented in Section 3.1 is a significant contribution for the following reasons. First, as is shown in Section 2.4, there are many existing incremental and "repeated incremental" batch segmentation models that all employ different features, which makes them seem disparate; however, many can be described within my framework (see Section 3.1), thereby allowing the systematic investigation of the effects of changing the instantiations of the framework's components, and simplifying the process of comparing competing segmentation models. Second, by unifying seemingly disparate segmentation algorithms, the framework makes the similarities that many segmenters share more apparent. Finally, those interested in studying the acquisition of phonotactic patterns may also be interested in the framework, because the models in it can acquire phonotactic patterns from unsegmented text, instead of from words (as is the norm).

feedback as to which responses were correct, in the word segmentation community "unsupervised" also implies a lack of a lexicon. This is because lexicons can be considered to be lists of one word utterances, although syntactically ill-formed ones, where the correct segmentation is known.

1.1.3 Phonological Word Evaluation

All corpora we are aware of that have been used for evaluating segmenters are divided into orthographic[2] words, but my research calls these evaluations into question, which should have an impact on the methodology others use to evaluate their segmenters. Models that make use of phonotactic features (e.g., Fleck, 2008; Blanchard et al., 2010; Daland, 2009) or innate knowledge of syllable structure (e.g., Johnson, 2008b) are searching for phonological words; however, these models have only been evaluated against corpora that were segmented into orthographic words, although they are phonetically transcribed ones. Therefore, a more appropriate way to evaluate these segmenters is to compare their output to a gold standard of phonological words. This will not make the problem easier for these segmenters, as many segmenters split "the" [ðə] from the nouns it precedes even though "the" is not considered a phonological word, but it will make error analysis simpler, since the phonotactic patterns found are already rooted in phonological words.

1.1.4 Early Error Recovery & Prevention

Unsupervised incremental[3] segmenters are especially challenging to develop. As they are incremental, there is very little information for the segmenters to use to make decisions at the early stages of learning. There is also no external feedback to let them know when a particular segmentation is incorrect, because the segmenters are unsupervised. Consequently, if poor decisions are made initially, it may be very difficult for an unsupervised incremental segmenter to recognize an error and prevent it in the future. Even worse, these early errors can trigger more later on. Therefore, the main challenge when developing an unsupervised incremental learner is the prevention of unrecoverable early errors.

As part of my dissertation, I will attempt to develop methods for preventing and recovering from early errors in an incremental learning process. These methods will be a significant contribution not only to the incremental unsupervised word segmentation community, but also to those interested in unsupervised learning in general, because early errors are problematic for both incremental and boot strapping approaches to unsupervised learning.

1.1.5 Accurate Incremental Segmenter

The final major contribution that my work will make is a highly accurate unsupervised incremental word segmenter. The instantiation of my segmentation framework presented by Blanchard et al. (2010) is currently the most accurate unsupervised incremental segmenter on the *de facto* standard corpus for evaluation in the unsupervised segmentation community, the Bernstein-Ratner (1987) corpus. Although this segmenter's results are promising, there is still room for improvement, and in my dissertation I will use the error prevention methods mentioned previously to develop more accurate unsupervised incremental segmenters.

[2]For thorough definitions of ORTHOGRAPHIC and PHONOLOGICAL WORDS see Section 2.1.2.
[3]The terms UNSUPERVISED and INCREMENTAL are defined in Section 2.1.

2 Background & Related Work

There is a substantial body of literature about segmenting natural language into different units. In order to prepare the reader for the discussion of these models and the proposed work, the first section outlines the terminology that will be used throughout this proposal. As I am primarily interested in word segmentation and how it relates to language acquisition, the second section discusses the corpora that are frequently used to evaluate unsupervised word segmentation systems, the third presents some commonly used features by these systems, and the fourth describes the many different instances of unsupervised word segmentation algorithms. The final section discusses segmentation in related problems, such as speech recognition.

2.1 Terminology

Before discussing the state-of-the-art of word segmentation, the reader needs to know some terminology that will be used throughout this proposal. Therefore, the first subsection defines some basic terms relating to machine learning, a broad area of research of which word segmentation is just one small area. The second tries to define what a "word" is for the purposes of my work.

2.1.1 Learning Algorithms

Machine learning algorithms can be broadly categorized as either batch or incremental. Batch algorithms process the entire corpus before outputting a result, whereas incremental algorithms output a result immediately after each item in the corpus is processed. In the word segmentation domain, batch segmenters process all of the unsegmented utterances before outputting a segmented version of the corpus, while incremental ones segment each utterance as they encounter it.

There are advantages and disadvantages to the batch and incremental approaches to word segmentation. Batch models can examine global patterns when determining where to insert boundaries, which allows them to have the same level of success segmenting utterances regardless of where they occur in the corpus. On the other hand, incremental models typically have poor performance for utterances toward the beginning of the corpus due to an initial lack of evidence. Furthermore, initial mistakes can cause a cascade of later errors with an incremental learner. For example, if an incremental segmenter incorrectly learns that "boy" [#bɔɪ#] should actually be segmented as "b oy" [#b#ɔɪ#], it will have learned that "b" [b] is a well-formed English word and will likely pull [b]s out of other words it encounters. Therefore, early errors is the primary concern with incremental learners. If one is only concerned with producing the word segmenter with the highest segmentation accuracy possible, a batch model would be appropriate; however, there is a major drawback to batch segmenters: they are fundamentally inappropriate for models of language acquisitions. This is because batch models assume a learner does not attempt to start segmenting until hearing a relatively large number of utterances. Consequently, batch segmenters require more memory and computational power than similar incremental ones. If incremental segmenters could yield similar performance to batch ones, then the additional resource requirements and incompatibility with language acquisition models would prove unnecessary.

Learning algorithms can also be classified according to the amount of feedback they get. Supervised algorithms are given pairs of unprocessed data and the corresponding results, whereas unsupervised algorithms are only given the data. For word segmenters, supervised models are usually given a lexicon of the words in the corpus, and they must then determine what is the most likely sequences of words from that lexicon that could make up a given unsegmented utterance. Unsupervised models do not start with a lexicon, which makes the problem substantially more difficult. However, if a supervised model encounters words in the corpus that are not in its lexicon (i.e., novel words), then the same techniques that are used for unsupervised segmentation could be used to learn these words.

2.1.2 What Is a Word?

When developing an algorithm to segment utterances into words, one immediately faces a thorny question: What exactly constitutes a "word"? This question has proved difficult for linguists. In a seminal book on morphology, Matthews (1991) waited until page 208 to say, "There have been many definitions of the word, and if any had been successful I would have given it a long time ago, instead of dodging the issue until now."

Here I follow Dixon and Aikhenvald's (2002) illuminating discussion of words in natural language, which proposes that there are PHONOLOGICAL WORDS, GRAMMATICAL WORDS, and ORTHOGRAPHIC WORDS. Grammatical words are defined as consisting of "a number of grammatical elements" that cannot be separated, "occur in a fixed order", and "have a conventional coherence and meaning" (Dixon and Aikhenvald, 2002).[1] Conversely, a phonological word can be defined roughly as a unit of at least one syllable such that there are phonotactic constraints governing its structure, and/or some phonological rules can only apply within or between such units. One example highlighting the difference between the two types of words in English is "it's." "It's" consists of two grammatical words ("it" and "'s"), but only one phonological word ("it's"). This is because "'s" is a CLITIC, which means it has a distinct meaning but cannot stand on its own as a phonological word, as it does not consist of at least one syllable. Orthographic word boundaries are determined by a society's writing conventions, so they do not necessarily line up with either phonological or grammatical words, though they often line up with one or the other (Dixon and Aikhenvald, 2002).

As the models discussed below operate over phonetically transcribed text, and one of my goals is to examine the contribution phonotactic features make to the segmentation process, the target unit for extraction is the phonological word. This is because phonotactic constraints determine the well-formedness of phonological words, but not grammatical or orthographic words. For example, "'s" lacks a vowel or syllabic sound, so it is not considered a well-formed when heard in isolation. Furthermore, phonological word boundaries always coincide with syllable boundaries, but this is not necessarily the case with grammatical or orthographic words. Thus, any segmenter that uses information about syllables in the corpus to find words must be searching for phonological words.

[1]There are a number of possible exceptions to these criteria, but in general, the definition seems to hold.

2.2 Corpora

Most recent word segmentation research has been evaluated on at least one of the following two corpora: the Bernstein-Ratner (1987) and Demuth (1992) corpora from the CHILDES (MacWhinney and Snow, 1985) database. These two corpora are described in detail below.

2.2.1 Bernstein-Ratner (1987) Corpus

The Bernstein-Ratner (1987, hereafter BR) corpus from the CHILDES database (MacWhinney and Snow, 1985) consists of 9,790 utterances containing 33,399 words of English infant-directed speech. The BR corpus is the same one that Brent (1999), Venkataraman (2001), Goldwater (2007), Fleck (2008), and Johnson (2008b) used to evaluate their models, and it has become the *de facto* standard for segmentation testing ever since it was phonemicized by Brent and Cartwright (1996).

The transcription system described in Brent and Cartwright (1996) makes some unorthodox choices. In particular, complex sounds traditionally transcribed with multiple symbols are transcribed with only one. These include diphthongs and vowels followed by [ɹ]. Another decision was to use different symbols for stressed and unstressed syllabic [ɹ]—that is, there are different symbols for the [ɹ] in "butter" [bʌtɹ̩] and the [ɹ] in "bird"[bɹ̩d]—though stress is not marked elsewhere in the corpus. To alleviate these issues, Blanchard and Heinz (2008) created a modified version of the corpus where the bi-phone symbols were split into two[2] and the syllabic [ɹ] symbols were collapsed into one. Blanchard and Heinz (2008) showed that current segmentation models do worse on the modified BR corpus, because the models have to learn that the diphthong vowels always co-occur without incorrectly grouping them together into their own words.

2.2.2 Sesotho Corpus

Sesotho is a Bantu language spoken mainly in South Africa. Bantu languages are agglutinative, which means that they have long words (both grammatical and phonological) consisting of many morphemes. The writing system for Sesotho is like that of most Bantu languages in that word boundaries are inserted where they would be in most European languages, which makes orthographic words shorter than grammatical or phonological ones.

Johnson (2008a) trimmed the Demuth (1992) corpus from the CHILDES database (MacWhinney and Snow, 1985) of speech between mother-child dyads to include only the child-directed speech. He did not convert the orthography to phonemes, because the writing system for Sesotho is nearly phonemic to begin with.[3] The final corpus contains 8,503 utterances consisting of 21,037 word tokens.

[2]With diphthongs, only those whose first phoneme can occur in isolation in English were split. Therefore, the vowels in "bay" and "boat" were not split.

[3]In addition to vowels, nasals sounds and the lateral liquid [l] can be syllabic in Sesotho. However, these sounds are not marked as such in the transcription, and so we treated all [l] and [n] sounds as non-syllabic.

2.3 Common Features for Word Segmentation

There are myriad features that one could choose to use for word segmentation, but they can generally be broken down into two categories: between-word or sub-word.[4] The between-word features are those such as word n-grams, that aim to capture the influence that neighboring words have on one another. The sub-word features are those that rate how well-formed a word is. Linguists call these types of word-structure-governing features phonotactic constraints (Chomsky and Halle, 1965; Halle, 1978). I describe some common features used for unsupervised word segmentation below.

2.3.1 Familiar Words

When adults face the task of word segmentation, we can simply apply a technique known as Word Spotting to pull out words from the speech stream (Cole and Jakimik, 1980). Essentially, we first identify familiar words in the utterance, and if there are any blocks of sounds leftover that are not currently in our lexicon, we consider them potential novel words. If they follow our language's phonotactic constraints, we add them to our lexicon; if not, we try to further segment the utterance into chunks that do follow phonotactic constraints. For example, if one has learned the word "what's" [wʌts] and encounters the utterance "What's this?" [wʌtsðɪs], it is a reasonable deduction that "what's" [wʌts] is an instance of a familiar word and "this" [ðɪs] is a novel word. This simple technique is the basis for many word segmentation models (e.g., Brent, 1999; Venkataraman, 2001; Blanchard et al., 2010).

Listening for familiar words is an approach to segmentation that infants use as well. Bortfeld et al. (2005) showed that six-month-olds can segment words that follow highly frequent words such as their names or "mommy", which is a good indicator that this feature is useful even in the earliest stages of learning to segment. As for how infants acquire the initial words they use to segment others, Brent and Siskind (2001) argue that infants learn these first words from one-word utterances. If infants are predisposed to consider utterances as words, then they will add entire multi-syllabic utterances to their lexicons at first. Although this results in many initial mistakes, as long as some utterances infants hear consist of one word, this strategy could be enough to bootstrap the lexicon. According to Brent and Siskind (2001), as much as 10% of infant-directed speech is made up of one-word utterances. Furthermore, even the multi-word utterances that are added to the lexicon can be helpful for segmenting future words. For example, if infants hear the utterance "Thank you" [θæŋkju] and incorrectly add the entire utterance to their lexicon, they could use that familiar phrase to segment "say" [seɪ] from the utterance "Say 'thank you"' [seɪθæŋkju].

2.3.2 Phoneme N-grams

A word's well-formedness can be approximated by the well-formedness of the phoneme combinations that make up the word. For example, in English the phoneme pair [kŋ] (orthographically, "kng") never occurs; therefore, words that contain [kŋ] sequences are not well-formed words of English. Hockema (2006) strengthened this claim by showing that phoneme pairs in English have a bimodal distribution. Specifically, pairs occur either frequently within words or frequently across word

[4]Blanchard et al. (2010) refer to "between-word features" as "familiar word cues," and "sub-word features" as "phonotactic cues", but these are just different names for the same classes of features.

boundaries, but not both (e.g., [ŋt] only occurs across word boundaries). For these reasons, a word's constituent phoneme combinations are informative when evaluating a it's well-formedness.

Phoneme combinations are not simply useful for word segmentation in theory; infants have been shown to use phoneme combinations when deciding how to segment novel utterances. Mattys and Jusczyk (2001) showed that nine-month-olds can segment speech by using the difference in probabilities between within-word and across-word consonant clusters. For example, the novel phrase "fang tine" [faŋ taɪn] is segmented as it is because [ŋt] does not occur within English words.

The most straightforward method of evaluating the likelihood of the phoneme combinations that make up words is through a phoneme n-gram model. The basic idea is that a word's probability can be estimated as the product of the probabilities of all phoneme sequences of length n within the word. Phoneme n-gram models have been shown to be useful sub-word features for unsupervised segmenters (e.g., Blanchard and Heinz, 2008; Blanchard et al., 2010).

2.3.3 Syllable N-grams

Another method for approximating a word's well-formedness is by evaluating the syllable combinations within it. While there have been no analogs of Hockema's (2006) phoneme combination study with syllables, there are English examples of cases where syllables have a probability of occurring together within a word, but not across word boundaries. For instance, in the Bernstein-Ratner (1987) corpus of infant-directed speech from the CHILDES database (MacWhinney and Snow, 1985) the most frequently occurring pair of syllables is [o.ke] "okay", and it only occurs word-internally. On the other end of the spectrum is [du.ju] "do you", which exclusively appears across word boundaries. Therefore, the syllable combinations within a potential word are indicators of its likelihood of being a word.

Much like how phoneme combinations have been shown to be useful cues for infants learning to segment speech, syllable transitions have also been shown to be used by infants. Saffran et al. (1996) asserted that infants could segment words of an artificial language based on their syllable probabilities. After only two minutes of listening to randomly repeated occurrences of four three-syllable nonsense words without any silence between them, infants listened significantly longer to three-syllable test words with low probability syllable bigrams versus those with high probability syllable bigrams. Thus, the authors concluded that the infants seem to be able to learn the transitional probabilities of different syllable sequences and can use that knowledge to recognize valid word forms.

There has been some debate about how to implement a segmentation strategy based solely on syllable n-grams, and some drastic claims have been made about the feature's isolated usefulness by Gambell and Yang (2004). However, their results were based on an exceptionally poor strategy for segmenting using syllable n-grams. As a counter-example to the claim that syllable n-grams can be useful on their own, Gambell and Yang created a segmenter that simply inserted boundaries between syllables that had locally minimal conditional probabilities. This approach was inherently flawed, as a string of single-syllable words cannot all be local minima, which prevents them from being segmented. Even a simple approach such as "insert a boundary when the probability is less than $x\%$" would not suffer from such a problem, which indicates that this particular segmentation strategy was nothing but a straw man.

2.3.4 Universal Constraints

Although all of the previously discussed features need to be learned from the data, there are certain universal properties of language that may be integral to the segmentation process. One such property is that phonological words consist of at least one syllable. As syllables require SYLLABIC sounds as their NUCLEI, this constraint can also be thought of as "all words require at least one syllabic sound." In English all vowels are syllabic, but there are also syllabic consonants: [l,n,ɹ] (e.g., "bottle" [batl̩], "button" [bʌtn̩], "butter" [bʌtɹ̩]). As is explained in Section 2.2, the syllabic consonants are transcribed differently from their non-syllabic counterparts in the English corpus we evaluate our models on.[5]

The main motivation for using a "require syllabic" constraint is that many suffixes (e.g., the plural markers [s] and [z] in English) do not contain syllabic sounds, and are easily over-segmented. For example, if the word "ball" [bal] is in the segmenter's lexicon, and then an utterance that is only "balls" [balz] is encountered, a segmenter that uses familiar words is likely to segment the utterance as "ball s" [bal z]. With the constraint in place, these errors can be prevented.

2.4 Unsupervised Word Segmentation

Many unsupervised word segmentation models have been proposed over the past 50 years (e.g., Olivier, 1968; Wolff, 1977; de Marcken, 1995)[6], but they can all be categorized into the following general approaches: Bayesian methods, local statistics, and neural networks. I discuss some representative models from each paradigm below.

2.4.1 Bayesian Models

There are two major categories of Bayesian word segmentation models: MAP (MAXIMUM *A Posteriori*) and hierarchical models. Both models make use of the basic concepts of Bayesian inference, but while the MAP models attempt to find the segmentation that maximizes the posterior probability, the hierarchical models search for a distribution over all possible segmentations.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{2.1}$$

The heart of Bayesian inference lies in Bayes' Theorem (Equation 2.1, where H means hypothesis and E means evidence). For the domain of word segmentation, we consider the evidence to be the corpus and a hypothesis to be a particular segmentation. The probability of a hypothesis absent any considerations of the evidence, $P(H)$, is called the hypothesis' PRIOR PROBABILITY. $P(E|H)$ is known as either simply the conditional probability of the evidence given the hypothesis, or the likelihood if the evidence is the same for all hypotheses. The probability of the hypothesis given the evidence, $P(H|E)$, is called the POSTERIOR PROBABILITY. As word segmentation models just need

[5]The syllabic consonants are plausibly distinguished acoustically from their non-syllabic counterparts (Toft, 2002; Xie and Niyogi, 2006).

[6]Harris's (1954) algorithm for extracting morphemes directly from unsegmented text is often cited as the first example of a word segmentation algorithm; however, it actually finds morphemes, so it is discussed in Section 2.5.3.

to know which segmentation is more likely than all the others, the exact scores are not necessary, so the marginal probability of the evidence, $P(E)$, is dropped from the equation to yield:

$$P(H|E) \propto P(E|H)P(H) \tag{2.2}$$

The equation can actually be simplified even further for word segmentation, because the likelihood of a segmentation is always either 1 or 0 (Goldwater et al., 2009). As long as the segmentation consists of the same phonemes in the same order (i.e., it is the same utterance only with added boundaries), this probability will be 1; otherwise, it will be 0. When searching the hypothesis space, segmenters do not consider inconsistent segmentations where the probability would be zero, so the likelihood term can be dropped from Equation 2.2 and the posterior is simply proportional to the prior:

$$P(H|E) \propto P(H) \tag{2.3}$$

This means that finding the segmentation with maximum prior probability will always yield the segmentation with maximum posterior probability. With these preliminaries established, the following subsections discuss MAP and Hierarchical Bayesian models in more detail.

MAP

All MAP segmenters try to find the segmentation with the maximum posterior probability; however, as we have established that the posterior probability is directly proportional to the prior, MAP segmenters differ from one another only in their methods for calculating $P(H)$ and searching the hypothesis space. For instance, MDL segmenters try to minimize the information theoretic length of the "description" of both the corpus and the lexicon that generated it, and are MAP models with a prior that gives hypotheses exponentially decreasing probabilities as they increase in length (Goldwater, 2007).

de Marcken (1995) created an MDL segmenter that is rather unique in its approach to language structure. His segmenter treated language as hierarchical structure of unspecified number of layers: at the lowest level were phonemes, and at the highest were utterances with morphemes, words, and phrases falling somewhere in between. Starting from an unsegmented corpus, his model first looked for co-occurrences of phonemes, and then recursively looked for co-occurrences of progressively larger types of units. The model fits into the MDL paradigm, because it returned the hierarchical structures that most compactly describe the corpus. Unfortunately, it is quite difficult to compare the performance of de Marcken's model to other segmenters, since no layer of the structure it returned was actually marked as the "word layer." When he evaluated the model's accuracy, de Marcken considered a word as successfully segmented if it appeared at any one of the layers; hence it would have an unfair advantage in any comparative evaluations with other models. Nevertheless, de Marcken is an important example of a MDL segmenter.

The first MDL segmenter aimed at modeling infant language acquisition was put forth by Brent and Cartwright (1996). It operated on a phonemic version of the Bernstein-Ratner (1987) corpus, which has become the *de facto* standard for evaluating unsupervised word segmenters, and was the first model to attempt to use phonotactic information to segment speech. Unfortunately, the algorithm was so computationally demanding that it could only be run on a small subset of the corpus (roughly 170 utterances out of 9790); therefore, the algorithm itself was not very influential, but the idea of examining the phoneme clusters appearing at word boundaries to help determine words' well-formedness was an important one.

The earliest example of a word segmentation algorithm was created by Olivier (1968), but it had some strange quirks (e.g., it had to process the corpus in chunks of exactly 480 characters, even if that meant a word was split incorrectly), so here I present the details of Batchelder's (2002) improved version of the model, BootLex, instead.[7] BootLex works differently than most of the models I describe in that it is a merging algorithm rather than a dividing one (i.e., its null hypothesis is that there is a boundary between every character in an utterance, while others assume utterances are single words by default), and the lexicon it builds is nearly all the evidence it uses to find words. When first processing a corpus, BootLex does an initial scan to find out the true average word length (in phonemes) of the corpus and the set of phonemes used in the corpus. Then, all phonemes are added to the lexicon with an initial count of one. BootLex then iterates through the corpus one utterance at a time, selecting the segmentation that with the highest product of lexical frequencies, adding all "word" pairs in each utterance to its lexicon as single words (if they are not already present), and incrementing the lexical counts of all words in the utterance by one. To prevent words from getting excessively long, segmentations with average word lengths higher than the true mean have their scores discounted.[8] In sum, BootLex is a merging MAP segmenter that uses a lexicon and an optimal length penalty to segment.

The model put forth by Venkataraman (2001) takes a dividing approach to segmentation, and searches unsegmented utterances for the sequence of words[9] with the highest probability. When a word in a segmentation is in the lexicon, its probability is simply equal to its lexical frequency; however, if the model considers a segmentation with a novel word, the word's probability is based on the product of the probabilities of its constituent phonemes. That is, novel words with more frequent phonemes are ranked more highly than those with more infrequent ones (e.g., a word with an "n" [n] in it would be ranked higher than a similar one with a "ng" [ŋ]). Because the model initially starts with an empty lexicon and a uniform distribution over the phonemes, the first utterance it encounters is added to the lexicon as a single word, and the frequencies of the phonemes within the word are updated. The process then repeats for the entire corpus. This simple approach was the most accurate unsupervised word segmenter[10] until Goldwater's (2007) came out, and only uses a lexicon and phoneme unigrams to segment.

MBDP-1 (Brent, 1999) is a predecessor of Venkataraman's (2001) model that is nearly identical to it. Brent explains MBDP-1 by describing a probabilistic model of how one would generate a corpus, and then showing how to calculate the probabilities of particular segmentation given some assumptions about the way the unsegmented text was generated. As I show in Section 3.1, while MBDP-1 was presented in an entirely different manner than Venkataraman's (2001) model, they only actually differ with respect to the way their phoneme counts are initialized and some minor variation in how they calculate word scores: MBDP-1 initially sets all phoneme counts to zero and explicitly gives preference to segmentations containing more frequent words and occurring

[7]Batchelder (2002) explains in detail the differences between the two models, and notes that MK10 (Wolff, 1977) is also closely related to BootLex.

[8]The exact equation for this appears to be missing from Batchelder's (2002) article, as it is supposed to be specified in footnote 16, but the footnotes jump from 15 to 17.

[9]Venkataraman (2001) also considered word bigrams and trigrams, but his results showed no improvement with either over the unigram baseline; therefore, I only discuss his unigram model here.

[10]Tied with MBDP-1, which is discussed below.

later in the corpus, whereas Venkataraman's model uniformly initializes its phoneme counts and has no extra penalties as part of its word scores. As the models are nearly identical and I am primarily concerned with the sub-word features each model uses, further discussion of MBDP-1 is unwarranted.[11]

The MAP segmenters discussed above did not make use of many sub-word features. Primarily, they used lexicons (or in de Marcken's (1995) case a grammar) to find the most probably sequences that have already been seen. The only sub-word features any of the models above use are phoneme n-grams[12] (of length 1 for all but Brent and Cartwright's (1996) model) and a constraint on average word length (Batchelder, 2002). However, some of these models could easily be extended to examine more robust sub-word features, as is discussed in Section 3.1.

Hierarchical

While the MAP models were concerned with just finding the single best for each utterance, the hierarchical Bayesian models discussed below find the distribution over all possible segmentations for entire corpus. With the hierarchical approach the corpus is segmented repeatedly until the distribution stabilizes (or some arbitrary number of iterations passes), and then the maximum scoring segmentation is output. The two most well-known examples of this type of segmenter are presented below.

Goldwater et al.'s (2009) segmenter uses an underlying generative model, much like MBDP-1 does, only her language model is described as a Dirichlet process. While this model uses a unigram phoneme distribution like all of the segmenters above, it considers segmented utterances to be sequences of words bigrams. A word bigram model is useful in that it prevents the segmenter from assuming that frequent word pairs are not simply one word, which Goldwater et al. observed happen with a unigram version of their model. The segmenter uses a Gibbs sampler augmented with simulated annealing to sample from the posterior distribution of segmentations and determine the most likely segmentation for each utterance. A key difference between this approach and the MAP models mentioned previously is that this makes many (usually 1000) passes over the corpus before outputting a final segmentation. We direct the reader to Goldwater et al. (2009) for details.

A related hierarchical model was implemented by Johnson (2008b,a), only instead of using a word bigram model it uses a general adaptor grammar for its language model. The main contribution of this approach is that it allows one to easily add new features to the model simply by adding more rules to the grammar. For example, in one adaptor grammar Johnson (2008a) specified that words consist of syllables which consists of phonemes, and in another he gave an even more detailed description of syllables by saying they consist of optional consonants at the beginning and end (i.e., an onset and a rime) with a vowel in the middle (the nucleus). This segmenter is the only one I am aware of that has any innate knowledge of syllable structure beyond a rule that syllables requiring vowels.

2.4.2 Connectionist Models

Cognitive scientists studying word segmentation and how it relates to language acquisition have typically used connectionist models (i.e., neural networks). The most frequently cited example of

[11]The interested reader is referred to Brent (1999) for a more detailed description.
[12]N-grams can simply be thought of as sequences of phonemes of length n.

this approach is from Christiansen et al. (1998), although there are other variants (e.g., Cairns et al., 1997; Christiansen et al., 2005). All of these approaches use a special type of neural network known as a Simple Recurrent Network (Elman, 1990), which is a standard feed-forward network except that it has extra connections that copy the output of one run through the network back as input for the next run. This allows the SRN to incrementally process the corpus and learn from the words it has segmented previously. As for the sub-word features, Christiansen et al.'s (1998) segmenter is particularly interesting in that it converted the phonetic transcription it processed to a series of phonological features[13] that included stress information; therefore, it could describe the input in finer detail than any of the other segmenters mentioned previously. Despite this intriguing way of handling the input, neural network models have been shown to output poorer segmentations than other types of models (Brent, 1999), so a more detailed discussion of this paradigm is omitted.

2.4.3 Local Statistics

A common approach for word segmentation algorithms to take is to calculate some sort of local statistic (e.g., mutual information, transitional probability, conditional entropy) at every possible boundary point in an unsegmented utterance and insert boundaries either at local minima or when the statistic is less than some threshold. Models using local minima have not been very successful (Brent, 1999; Gambell and Yang, 2004), because this method makes it impossible for single-unit words to be segmented, as two minima cannot occur contiguously. Of the segmenters that have used thresholding successfully (Cohen and Adams, 2001; Swingley, 2005; Fleck, 2008; Daland, 2009; Hewlett and Cohen, 2009), two use especially effective features: WordEnds (Fleck, 2008) and Bootstrap Voting Experts (Hewlett and Cohen, 2009).[14] These two models are described in detail below.

WordEnds (Fleck, 2008) segments words by calculating the probability of a boundary given a left and right context (i.e., prefix or suffix) of arbitrary length and inserting word boundaries when their joint probability exceeds 0.5. These boundary probabilities are essentially calculated using phoneme n-grams without a fixed value for n; n is set to the largest value for which a particular context has occurred during training more than some threshold (10 in Fleck's experiments).[15] WordEnds makes three passes over the corpus to determine final word boundaries. On the first pass, WordEnds attempts to bootstrap the word boundary probability estimation by calculating the probability of contexts that occur at **utterance** boundaries, and setting the probability of these contexts given **word** boundaries to be a constant high value if the utterance boundary probabilities exceed a certain threshold (0.003 in Fleck's experiments). With these word boundary probability estimates established, it makes a second pass through the corpus where it inserts word boundaries whenever the estimated probability exceeds 0.5. WordEnds then uses the newly inserted word

[13]Phonological features describe the articulatory or acoustic properties of speech sounds and are useful for comparing speech sounds. For example, [t] and [d] agree on most features except VOICE with [t] being a voiceless sound and [d] being voiced.

[14]Daland's (2009) segmenter also uses a particularly useful sub-word feature, phoneme bigrams, but there is no unsupervised version of his model.

[15]Fleck (2008) noted that if no context has occurred at least ten times, a single-character context is used. There is also a maximum context length parameter, which was set to five during the first pass and four during the third pass.

boundaries to re-estimate the context probabilities by calculating the standard MLE probabilities during the third pass. For the final pass, WordEnds again inserts word boundaries whenever the estimated probability of a word boundary given the right and left contexts exceeds 0.5. This multi-pass, variable-length phoneme n-gram approach was rather successful, and it outperformed even Goldwater's (2007) segmenter on most corpora it was evaluated on.

To prevent some "obvious" errors in WordEnds' segmentation, in her experiments Fleck (2008) employed a simple morphological processor called Mini-morph that ran through the output from WordEnds. Essentially, Mini-morph calculates the ratio of the number of times a word has occurred in the segmented corpus by itself to the number of times it has appeared as a prefix or suffix of another word; words are either split or merged based on the value of this ratio. While no one else has used Mini-morph to correct segmentation errors, there is nothing specific to WordEnds about the Mini-morph algorithm, so it could theoretically be used to spot and correct errors in any segmenter's output.

One of the most popular word segmenters (e.g., Cheng and Mitzenmacher, 2005; Miller and Stoytchev, 2008) that uses local statistics is Voting Experts (Cohen and Adams, 2001; Cohen et al., 2007). In the basic form of the algorithm, two "experts" vote on where to insert boundaries within each utterance with one voting to insert boundaries after sequences with low internal entropy[16], and the other voting to insert boundaries after sequences with high boundary entropy[17] (Hewlett and Cohen, 2009). Sequences have low internal entropy if their constituents reliably predict one another and have high boundary entropy if the final elements **cannot** reliably predict what follows them. Voting Experts inserts boundaries whenever the number of votes for a particular position exceeds a certain threshold. While this approach is rather straightforward, the base model is not an incremental model, because it first scans the entire corpus to gather the statistics necessary to calculate the two different types of entropy before inserting any boundaries. However, recently Hewlett and Cohen (2009) implemented incremental versions of both the baseline and an enhanced version of it called Bootstrap Voting Experts.

Bootstrap Voting Experts repeatedly segments the entire corpus, each time decrementing the voting threshold until a specified minimum value has been reached. It uses an extra expert, the KNOWLEDGE EXPERT, that uses the boundaries from the previous iteration to gather the necessary information to calculate the sum of the internal entropies of the sequences on either side of a potential word boundary, including the word boundary symbol, #. To use the example Hewlett and Cohen (2009) gave, if a potential split has the sequences "it" and "was" on either side of it, the Knowledge Expert will calculate the internal entropy of "it#" and "#was" and vote for that split if the sum of those entropies is a locally minimal. One interesting fact about this approach is that the initial seed segmentation the model uses is simply the output of the original Voting Experts when the voting threshold is set very high. This means that the initial segmentation has a very high boundary precision—because it only inserts boundaries when it is very confident they will be right—and with each subsequent pass through the corpus the model is made less conservative to increase boundary recall. To make the segmenters incremental, Hewlett and Cohen modified them to update after each utterance was processed the statistics necessary to calculate the different entropies used by the experts. In Hewlett and Cohen's experiments, the incremental versions of both Voting Experts and Bootstrap Voting Experts proved to be quite accurate (boundary $F_1 > 70\%$

[16] $H_I(seq) = -log(p(seq))$

[17] $H_B(seq) = -\sum_{c \in S} p(c|seq) log(p(c|seq))$ where S is the set of successors to seq.

on the BR corpus) even after processing relatively few (< 1000) utterances in their experiments. This indicates that the entropy features used by the Voting Experts models become reliable rather quickly.

An interesting application of the Voting Experts algorithm was attempted by Miller and Stoytchev (2009): run Voting Experts on the sequence of discrete symbols (mostly phones) returned by an unsupervised automatic phone recognizer (Iwahashi, 2006; Brandl et al., 2008). This alternative approach to speech recognition was completely unsupervised, and the segmentation capabilities the system demonstrated on datasets from a well-known infant segmentation experiment by Aslin et al. (1998) almost perfectly matched the infant results. Although these results were obtained on very small datasets (approximately 2 minutes of uninterrupted nonsense syllables), Miller and Stoytchev's method does appear to be able to segment words directly from the speech stream, which was not previously possible in an unsupervised fashion. The standard supervised approaches to speech recognition and word segmentation are discussed in the next section.

2.5 Other Types of Language Segmentation

There are many Natural Language Processing tasks that involve segmenting larger linguistic units into smaller ones. Each of these tasks has its own unique set of proposed solutions, but in general most of the approaches to solving these problems are supervised and use sub-word features at the phoneme or character level. The types of language segmentation most related to unsupervised word segmentation are: speech recognition, supervised word segmentation, and morpheme segmentation. All of these problems are discussed below.

2.5.1 Speech Recognition

Because words in spoken language form a continuous stream, automatic speech recognition systems must confront the problem of word segmentation. Most modern speech recognition systems use first-order Hidden Markov Models (HMMs) to identify the correct sequence of words given audio input. HMMs are essentially probabilistic automata where the current state can be predicted solely from the previous (like a bigram model), and the states represent "hidden" properties associated with certain observable features. For example, the HMMs used at the lowest level of speech recognition usually model phones: they have three states to represent the beginning, middle, and end portions of the phone, and the states are assigned probabilities conditioned on the acoustic properties the speech stream exhibits during each state. The probabilities for the HMMs are calculated by training the models on many different labelled examples of the phones in question, which makes this a supervised learning problem. To construct a model to recognize words within a particular lexicon, the pronunciations of all the words are looked up in a pronouncing dictionary, and then word HMMs are constructed by concatenating the corresponding phone HMMs together. To account for the transitions between words, the final model is constructed by taking all the word HMMs and adding transitions at the beginning and end of each word HMM such that any word can transition to any other (with some bigram probability). Once the final model has been constructed the Viterbi beam search algorithm is used to find the most likely series of states (and thereby words) that account for all the words in a given utterance.[18] Beam search is simply a version of

[18]See Jurafsky and Martin (2008) for details.

the standard Viterbi algorithm that removes unlikely paths from consideration; similarly, a version of the basic Viterbi search is used by some of the unsupervised word segmentation systems I have presented (e.g., Brent, 1999; Venkataraman, 2001). Thus, it is not the search algorithm itself that distinguishes speech recognition from word segmentation, but rather the underlying model: word segmenters are completely confident in what the phones within the utterances are, whereas speech recognizers require the additional HMM layer to try to determine what phones are being spoken in the first place. If we replaced the phone HMMs in a speech recognizer with phoneme bigram models, the problem is just an instance of supervised word segmentation, which I discuss in the next section.

2.5.2 Supervised Approaches to Word Segmentation

Most supervised words segmenters are used to segment orthographic text in languages that do not insert whitespace between words (e.g., Chinese); therefore, many models primarily use large lexicons to extract as many words as possible, and fall back on sub-word features and other statistics when encountering novel words. Since the sub-word features are at the character level, the patterns they exhibit could be very different than those between phonemes in an a phonetic corpus, which means that features that work for supervised models may not work well for unsupervised models operating on phonetic corpora. Therefore, I omit a more detailed discussion of supervised approaches, and refer readers to Lu (2006) for a review of Chinese word segmentation methods.

2.5.3 Morpheme Segmentation

Although it is sometimes referred to as "word segmentation," morpheme segmentation is a distinct problem from word segmentation. In morpheme segmentation, words are broken up into their component morphemes, and the segmenter is usually given words as input. Word segmentation and morpheme segmentation are often conflated; hence many papers cite Harris (1954) as the first to study word segmentation, when he was the first to study morpheme segmentation. The majority of morpheme segmentation algorithms use MDL and are concerned primarily with reducing redundancy in the lexicon, where the lexical items are strings of phonemes thought to be morphemes. For a thorough review of morpheme segmentation algorithms and their differences from word segmentation ones, refer to Goldsmith (2009).

3 Progress

In this chapter, I outline my current research accomplishments. The first section explains the framework that will be used to facilitate my investigation of sub-word features, and the second discusses the experiments I have conducted thus far to compare the utility of the sub-word features described in Section 3.2.2.

3.1 PHOCUS Framework

Here I present a general framework for word segmentation, called PHOCUS (PHOnotactic CUe Segmenter[1]), which enables the systematic investigation of different segmentation techniques through changing the instantiations of its components. PHOCUS generalizes the segmentation process in such a way that it unifies many existing segmenters.

3.1.1 Components

In the PHOCUS framework, an incremental word segmenter has the components shown in Figure 3.1: an Evidence Initializer, an Utterance Enumerator, a Learned Segmentation Model, and an Evidence Updater. First, the unsegmented corpus is passed to the evidence initializer, which sets the initial state of all the features for the segmenter. The corpus is then forwarded to the Utterance Enumerator, which simply selects an utterance from the corpus to be segmented and passes it on to the Learned Segmentation Model. The Learned Segmentation Model is the heart of the incremental segmenter, and is discussed in great detail below. For now, I treat it as a "black box" that takes in an unsegmented utterance and yields a segmented one based on the current state of the evidence the segmenter has acquired. This segmented utterance is then both outputted and passed to the Evidence Updater, which updates the states of the segmenter's features to account for any new evidence in the current segmentation. For example, an Evidence Updater may update the lexical frequencies of all the words that have been segmented. After this maintenance step has been taken, control transfers back to the Utterance Enumerator, which yields another utterance to segment, if any are remaining. Otherwise, the segmenter exits and the segmentation process is complete. It is important to note that while the definitions of all of the components in the system are rather flexible, for a system to be incremental it must only process the corpus once.[2]

[1]The name comes from earlier work (Blanchard et al., 2010) that was written for a language acquisition audience. What I refer to here as "sub-word features" are referred to in the language acquisition literature as "phonotactic cues."

[2]One potential exception to this is the Evidence Initializer, as we often want the model's initial n-gram distributions to be uniform, which requires knowing the set of phonemes used in the corpus. As different corpora often use different transcription systems, the model must run through the corpus once to know all of the phonemes it includes.

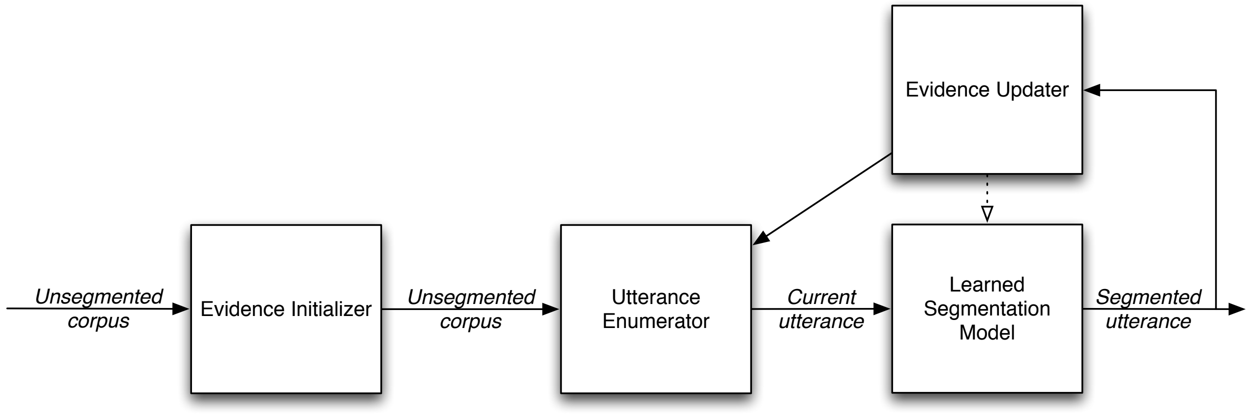

Figure 3.1: Incremental Segmenter

As shown in Figure 3.2, the Learned Segmentation Model takes an unsegmented utterance and passes it to the Candidate Proposer to get a list of possible segmentations. In all models we examine, the Candidate Proposer lists all possible segmentations, but this is not technically necessary, as there may be some innate bias one would want to put in a segmenter to avoid certain possible segmentations (e.g., only consider segmentations that consist of less than n words). These possible segmentation are then enumerated by the Candidate Enumerator, which either selects a candidate and passes it to the Candidate Evaluator or, if there are no candidates left, passes control straight to the Top Candidate Selector. The Candidate Evaluator is what assigns the scores to a given segmentation, and passes the score on to the Top Candidate Selector, which accumulates all of the scores until it is time to choose the best one. In every model we examine, the Top Candidate Selector simply chooses the segmentation with the highest score.

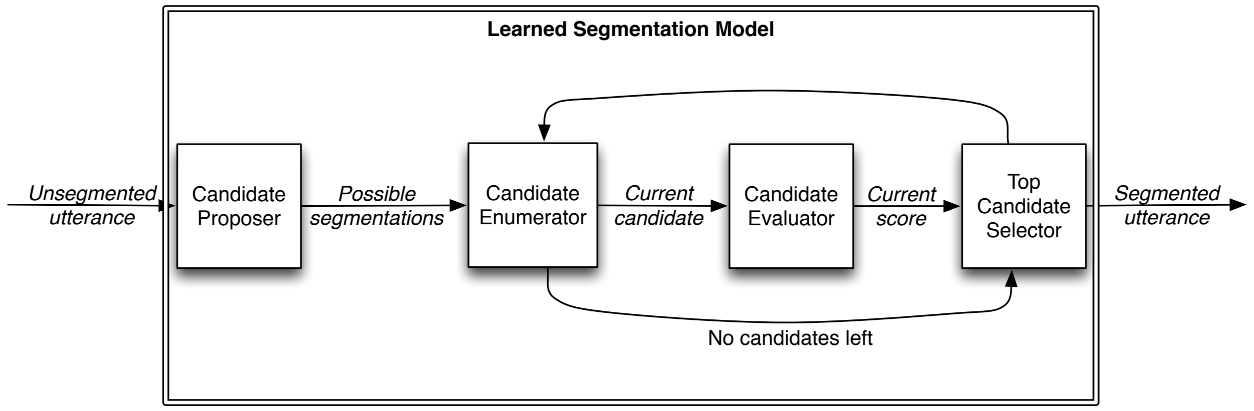

Figure 3.2: Learned Segmentation Model

The Candidate Evaluator returns a score for a possible segmentation by evaluating the language model constituents in the segmentation (e.g., word n-grams). First, all of constituents in the segmentation are enumerated, and then each constituent is passed on to the Language Model

CONSTITUENT EVALUATOR. The resulting scores are then combined by the CANDIDATE SCORE COMBINER. In most models we examine, the score combiner simply multiplies all of the constituent scores (e.g., word n-gram probabilities) together to yield a score for the current segmentation.

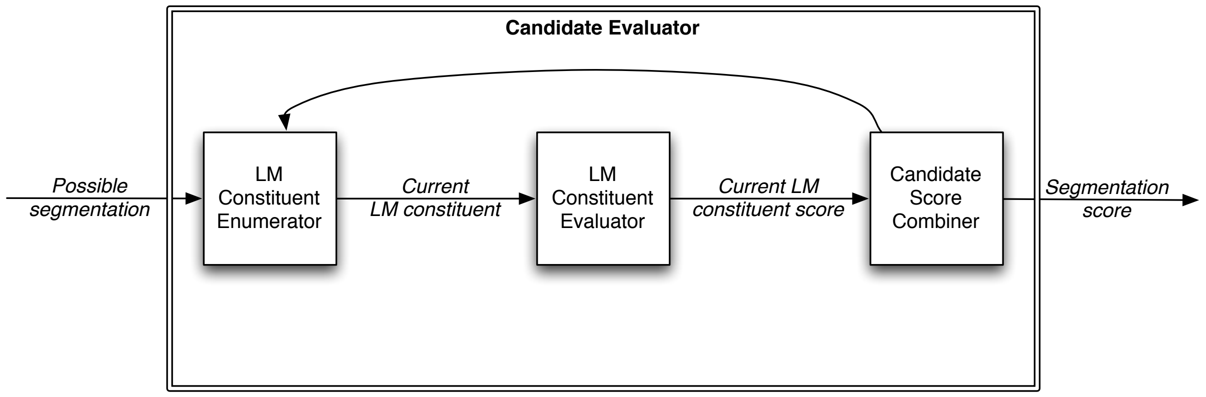

Figure 3.3: Candidate Evaluator

The Language Model Constituent Evaluator calculates two kinds of scores for a given constituent: word scores and language model scores. First, the words in the constituents are sent to the WORD EVALUATOR, and the resulting scores accumulated by the LANGUAGE MODEL SCORE COMBINER. The constituent (e.g., n-gram) itself is also evaluated by the LANGUAGE MODEL EVALUATOR. In most segmenters, the Language Model Evaluator scores n-grams (in most cases, unigrams) on the basis of their relative lexical frequency, although alternative proposals are discussed in later sections. Regardless of the method for determining the scores, the language model scores evaluate words on their contexts, while the word scores are for individual words.

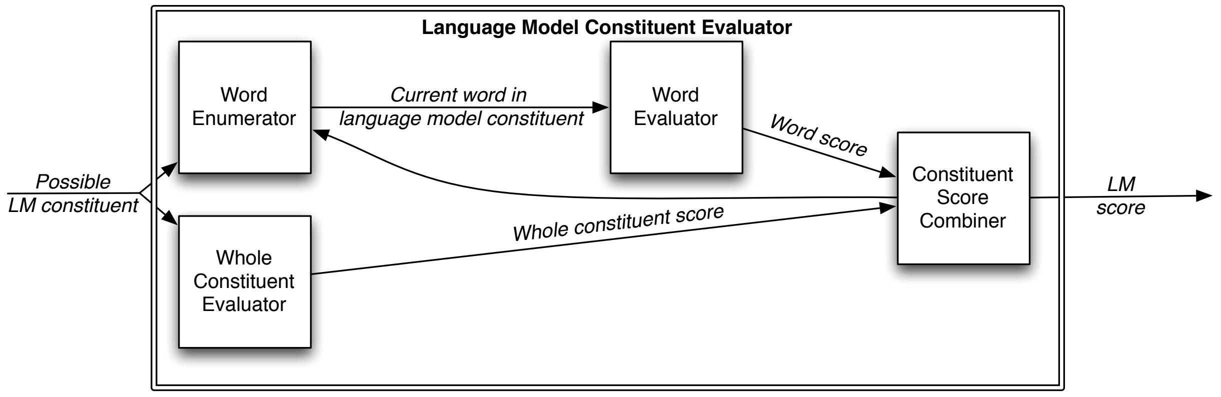
Language Model Constituent Evaluator
Word Enumerator
Current word in language model constituent
Word Evaluator
Word score
Possible LM constituent
Constituent Score Combiner
LM score
Whole constituent score
Whole Constituent Evaluator

Figure 3.4: Language Model Constituent Evaluator

The Word Evaluator consists of an unspecified number of word feature evaluators, and a WORD SCORE COMBINER that puts all of the scores together. Some possible evaluators are shown in Figure 3.5. In general, the sub-word features examined here score the word on the basis of its different components (e.g., phonemes, syllables). The specific features that we examine in the current experiments are discussed in Section 3.2.2. The feature most prevalently used by existing segmentation models is a unigram phoneme evaluator, which scores a word based on the product of the relative frequencies of its phonemes. The primary goal of my research is to pinpoint what features are most beneficial when added to the Word Evaluator, and to determine what mechanism is appropriate for combining those scores.

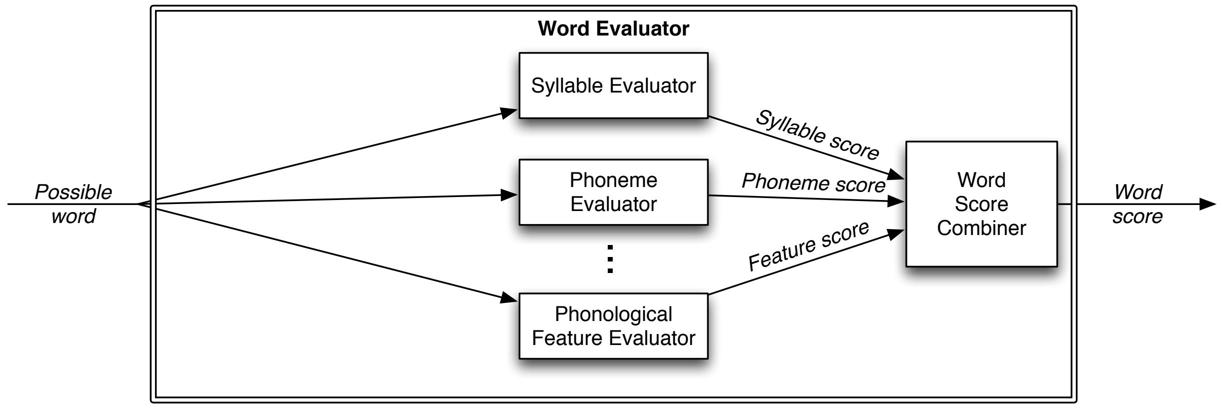

Figure 3.5: Word Evaluator

The PHOCUS framework is not limited to incremental models, as it can also describe batch models that repeatedly apply an incremental segmentation procedure ("repeated incremental segmenters"). This way we can evaluate whether certain word features are only helpful in batch segmenters, and it allows the framework to unify even more models. As shown in Figure 3.6, there are two additional components to the framework when dealing with batch segmenters: the Global Evidence Updater and the Halt Condition Checker. With the Global Evidence Updater, the model can update the evidence for features that can only be evaluated after the entire corpus has been initially segmented. The Global Evidence Updater can also be used to modify the score combination functions, so that each run of the incremental segmenter can evaluate segmentations however the model designer would like. The Halt Condition Checker simply prevents the incremental segmenter from being called again once the halting condition has been met. For example, one possible halting condition is "Has the incremental segmenter produced two identical segmentations for the corpus in a row?" These two simple additions allow for batch models in the PHOCUS framework.

3.1.2 Instantiations

There are many existing models that fit into the PHOCUS framework. The most obvious ones are MBDP-1 (Brent, 1999), MBDP-Phon (Blanchard and Heinz, 2008), the model of Venkataraman (2001), BootLex (Batchelder, 2002), and the PHOCUS models described by Blanchard et al. (2010), but other existing models may also fit into the framework.[3] Table 3.1.2 describes how the framework

[3]Hewlett and Cohen's (2009) incremental implementation of Voting Experts (Cohen and Adams, 2001; Cohen et al., 2007) seems like it would fit into the framework, but it is not clear exactly what the initial state of their model

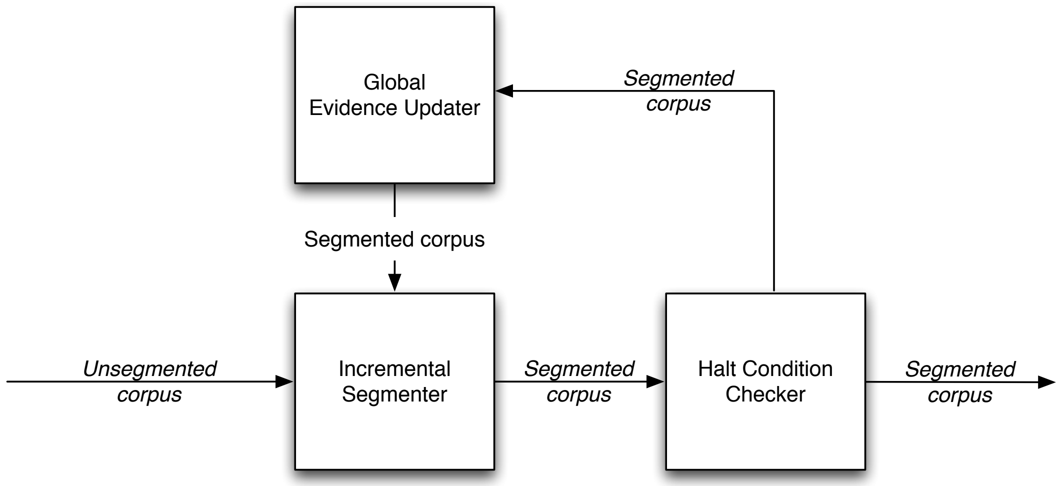

Figure 3.6: Batch Segmenter

must be instantiated for each of the previously mentioned models, and it illustrates the many things that these incremental models have in common. To save space, Table 3.1.2 was written in the following notation. To write that a score a is used unless it is equal to zero, in which case b is used, we write $a \prec b$. $L(w)$ is the relative lexical frequency of a word, while $l(w)$ is the word's raw lexical count. Similarly, $P(x)$ is the relative frequency of a particular phoneme n-gram, and $p(x)$ is the n-gram's raw count. For the total phoneme n-gram score of a word, we write $\mathcal{P}(w)$. $s(w)$ is the well-formedness score (i.e., the score yielded by the Word Score Combiner). There are two score adjustments that are used by the MBDP models, which are $Z(w)$ and $Y(w)$. The former is an adjustment to further penalize the score of infrequent words regardless of the size of the lexicon, while the latter is an adjustment based on the size of the lexicon. $S(w)$ is a binary-valued "require syllabic" evaluation function, which returns one if w contains a syllabic sound (e.g., a vowel), and zero otherwise. BootLex also has an extra score adjustment, $B(S)$, that decreases the scores of segmentations whose mean word length is greater than the true mean length for the current corpus. Finally x_n is used to indicate a phoneme n-gram of size n, with x abbreviating x_1.

While Brent (1999) and Venkataraman (2001) describe their models in entirely different ways, it has been noted that their performance is nearly identical when Venkataraman's model uses a word unigram language model (Venkataraman, 2001; Batchelder, 2002; Blanchard et al., 2010). As shown in Table 3.1.2, these performance similarities should be expected, because the only differences between the two models are that MBDP-1 does not start with a uniform phoneme distribution, and that the Phoneme Evaluator and Language Model Evaluator components for MBDP-1 both have extra score adjustment terms.[4] Even though the expositions Brent (1999) and Venkataraman (2001) present are very different, their models are identical in every other way, a fact not easily noticed outside of the PHOCUS framework.

As can be seen in Table 3.1.2, all of the segmenters except BootLex are highly similar. The major differences between BootLex and the other models lie in how they handle novel words. The

is; thus I omit it.

[4]The score adjustments $Z(w)$ and $Y(x)$ in MBDP-1 are what account for the Bayesian prior.

other models use phoneme n-grams to evaluate novel words' well-formedness, whereas BootLex never proposes a segmentation that contains novel words. This is because BootLex initially adds all of the phonemes in the corpus to its lexicon, thereby giving it some "words" that it can use to segment. The other major difference between the two approaches is that BootLex is a clustering algorithm and adds all word bigrams that are not currently in the lexicon to its lexicon (as single words) during the Evidence Updater step. Clustering segmenters such as BootLex use the baseline strategy that given an utterance and no other information, it should be split into all of its individual phonemes and they may later be combined; dividing segmenters like MBDP-1 assume utterances are a single word given nothing else to go on and these words are later divided. It is worth noting that these two different approaches can both be described easily within the PHOCUS framework.

Model	Evidence Initializer	Evidence Updater	Language Model Evaluator $N(w)$	Language Model Score Combiner	Phoneme Evaluator $\mathcal{P}(w)$	Word Score Combiner $s(w)$
MBDP-1	$\forall(x \in \Sigma)p(x) = 0$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x \in w)p(x) = p(x) + 1$.	$L(w)Z(w)$	$N(w) \prec s(w)$	$\prod_{x \in w} P(x)Y(x)$	$\mathcal{P}(w)$
MBDP-Phon	$\forall(x \in \Sigma)\forall(y \in \Sigma)p(xy) = 0$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x_2 \in w)p(x_2) = p(x_2) + 1$.	$L(w)Z(w)$	$N(w) \prec s(w)$	$\prod_{x_2 \in w} P(x_2)Y(x_2)$	$\mathcal{P}(w)$
Venkataraman	$\forall(x \in \Sigma)p(x) = 1$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x \in w)p(x) = p(x) + 1$.	$L(w_3) \prec L(w_2) \prec L(w)$	$N(w_n) \prec \prod_{w \in w_n} s(w)$	$\prod_{x \in w} P(x)$	$\mathcal{P}(w)$
BootLex	$\forall(x \in \Sigma)l(x) = 1$	$\forall(w \in S)l(w) += 1$. $\forall(w_2 \in S)$(If $l(w_2) = 0$) then $l(w_2) = 1$)	$L(w)$	$N(w)B(S)$	—	—
PHOCUS-1	$\forall(x \in \Sigma)p(x) = 1$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x \in w)p(x) = p(x) + 1$.	$L(w)$	$N(w) \prec s(w)$	$\prod_{x \in w} P(x)$	$\mathcal{P}(w)$
PHOCUS-2	$\forall(x \in \Sigma)\forall(y \in \Sigma)p(xy) = 1$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x_2 \in w)p(x_2) = p(x_2) + 1$.	$L(w)$	$N(w) \prec s(w)$	$\prod_{x_2 \in w} P(x_2)$	$\mathcal{P}(w)$
PHOCUS-3	$\forall(x \in \Sigma)\forall(y \in \Sigma)\forall(z \in \Sigma)p(xyz) = 1$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x_3 \in w)p(x_3) = p(x_3) + 1$.	$L(w)$	$N(w) \prec s(w)$	$\prod_{x_3 \in w} P(x_3)$	$\mathcal{P}(w)$
PHOCUS-3s	$\forall(x \in \Sigma)\forall(y \in \Sigma)\forall(z \in \Sigma)p(xyz) = 1$	$\forall(w \in S)l(w) += 1$. If $l(w) = 1$ then $\forall(x_3 \in w)p(x_3) = p(x_3) + 1$.	$L(w)$	$N(w) \prec s(w)$	$\prod_{x_3 \in w} P(x_3)$	$S(w)\mathcal{P}(w)$

Table 3.1: PHOCUS Instantiations

Example Models

To illustrate how exactly some models are instantiated in the PHOCUS framework, here I discuss the similarities and differences between PHOCUS-1, PHOCUS-2, PHOCUS-3, and their counterparts PHOCUS-1s, PHOCUS-2s, and PHOCUS-3s from our earlier work (Blanchard et al., 2010). All of these models evaluate use the same simple back-off approach for their Language Model Score Combiner: if a word is already present in the lexicon, the Language Model Evaluator score is used; otherwise, the Word Score Combiner score is. The number in the names of these models simply specifies the length of the phoneme n-grams used by the Phoneme Evaluator, and the models that end with "s" require every word to contain at least one syllabic sound by having the Word Score Combiner multiply the Phoneme Evaluator score by one or zero depending on the presence or absence of a syllabic sound. The Language Model Evaluator in all of these models simply scores words based on their relative lexical frequency. All of these PHOCUS models start with an initial uniform distribution over the phoneme n-grams by having the Evidence Initializer set the count of every possible n-gram of the size the Phoneme Evaluator uses to one before the first utterance is segmented. After a particular segmentation has been chosen, for every word in the segmentation the Evidence Updater increases the lexical count by one and the counts of every n-gram in the word by the number of times they occur in the word. The only way in which these PHOCUS models differ from each other are with respect to the phoneme n-gram size the Phoneme Evaluator uses and whether or not words are required to contain a syllabic element.

Evaluation of PHOCUS

I have established that PHOCUS unifies the segmenters of Brent (1999), Batchelder (2002), Venkataraman (2001), and Blanchard et al. (2010); however, there are other segmenters (e.g., Hewlett and Cohen's (2009) incremental implementation of Voting Experts (Cohen and Adams, 2001; Cohen et al., 2007)) that should fit into the framework whose exact instantiations have yet to be determined. In my dissertation I will examine which state-of-the-art segmenters can be instantiated within the PHOCUS framework to allow for easier comparison between models. As it is unclear how one could mathematically prove that the PHOCUS instantiations of these models are correct, I will have to verify the accuracy of them by comparing the outputs from running the PHOCUS and original versions of each segmenter on the same corpus. Those with identical segmentation can be considered to be correctly instantiated. This will concretely establish that the PHOCUS framework unifies many seemingly disparate models.

3.2 Preliminary Experiments on Feature Utility

To evaluate the performance of the sub-word features mentioned in Section 3.2.2, we run two experiments. First, to get an idea of how the features work in an ideal situation we run the PHOCUS models with 90% of the corpus as training data (i.e., the models know where the word boundaries go for those utterances) via ten-fold cross validation. The corpus is split into ten equal-sized subsets, and then each segmenter is trained on nine sets in a supervised fashion and then tested on the remaining set. The sets are then rotated and the process is run again, so in the end every set has been used for both training and testing. The second experiment tests the segmenters in progressively more realistic scenarios, by training on smaller and smaller portions of the corpus before being tested on the remaining portions. In this experiment each segmenter is allowed to continue to update its phoneme or syllable n-gram counts during the testing phase. This is because when the segmenter is not given any training data, it will not be able to segment at all if it cannot update its counts; we allow updating in all trials of this experiment for consistency. The combination of these two experiments allows us to draw conclusions about how the sub-word features perform with varying amounts of training data.

3.2.1 Experimental Setup

In each of the experiments, we run PHOCUS models that exclusively use either phoneme n-grams or syllable n-grams to score both novel and familiar words. We vary the length of n-grams from one to three for each case, and we use the phoneme n-grams with and without the "require syllabic" constraint.[5] In both experiments, we disable the familiar word feature (i.e., the lexicon), so that the segmentation relies purely on the sub-word feature we are evaluating. The main reason for making this choice is that even with a relatively small amount of training data (10% of the corpus), a model without any sub-word features[6] that uses the lexicon for word spotting will yield a word F_1 of 72.07%,[7] and since the models currently use a simple back-off model from lexical to sub-word scores, the differences in performance of the sub-word features is less obvious when the familiar words feature is enabled. Thus, to allow for a more informative comparison of the sub-word features, we disable the lexicon in these experiments, but for completeness-sake we have included results with the lexicon enabled in Appendix 3.A.

For both n-gram features in both experiments, we calculate the scores for each word using the chain rule as:

$$P(\#x_1 \dots x_m\#) = P(\#) \times P(x_1|\#) \times P(x_2|\#x_1) \times P(x_3|x_1x_2) \times \dots \times P(\#|x_{m-1}x_m) \qquad (3.1)$$

for trigrams,

$$P(\#x_1 \dots x_m\#) = P(\#) \times P(x_1|\#) \times P(x_2|x_1) \times \dots \times P(\#|x_m) \qquad (3.2)$$

for bigrams, and

$$P(\#x_1 \dots x_m\#) = P(\#) \times P(x_1) \times P(x_2) \times \dots \times P(x_m) \times P(\#) \qquad (3.3)$$

for unigrams, where x_i is either a phoneme or syllable. As word boundary symbols are added to the beginning and end of every word scored, and the word scores are multiplied together to yield a score for the entire segmentation, we drop the initial $P(\#)$ from our calculation; otherwise, each inter-word boundary would be scored twice. This does not affect the calculation for the initial word in the segmentation, because the probability of a word boundary at the beginning of the utterance is 100%. To prevent word probabilities from being 0% when there is little training data, we smooth the n-gram models by setting the initial counts for all possible n-grams to a small constant number (0.0001).

[5]We do not evaluate the syllable n-grams without the "require syllabic" constraint, because all syllables require a syllabic element; we want all of the n-grams to consist of syllables, not a mixture of syllables and non-syllables.

[6]All novel words are given a score of zero in this case.

[7]The F_1 is 72.07% when the lexicon is allowed to be updated after the training portion is over, and 64.50% when it is not.

3.2.2 Features Implementations

As was mentioned in Section 2.3, there are two main types of features for word segmentation: "between-word" and "sub-word." I primarily investigate sub-word features because not only do the sub-word (i.e., phonotactic) features more directly evaluate a potential word's well-formedness, but also there are many studies that show infants seem to use these types of features to segment speech (e.g., Saffran et al., 1996; Jusczyk et al., 1999a; Mattys and Jusczyk, 2001). I describe how each feature that will be included in these experiments are implemented below.

Phoneme N-grams

The most straightforward method of evaluating the likelihood of the phoneme combinations that make up words is through a phoneme n-gram model. The basic idea is that a word's probability can be estimated as the product of the probabilities of all phoneme sequences of length n within the word. Phoneme n-gram models have been shown to be useful sub-word features for unsupervised segmenters (e.g., Blanchard and Heinz, 2008; Blanchard et al., 2010).

There are many ways in which n-gram probabilities can be estimated. The most commonly used method in the NLP community is to calculate the conditional probability of a phoneme given the $n - 1$ phonemes preceding it. This technique is especially relevant when one is trying to predict what the next phoneme will be given the ones before it. As we are really interested in a measure of how likely a given sequence of phonemes is a word, rather than how likely a given sequence of phonemes is to occur, an alternative way of estimating phoneme n-gram probabilities may be appropriate for this task. To determine the probability that a particular phoneme n-gram occurs within a word, versus across word boundaries, the count of the n-gram within words is divided by the number of times that phoneme sequence has occurred in the unsegmented corpus. Unfortunately, these probabilities are not independent, so it is unclear how to appropriately combine them.[8] In light of this issue, in the experiments reported below, I use the standard conditional probability approach.

Syllable N-grams

In this work syllable n-grams are used to score segmentations in the same manner as any other sub-word feature in the PHOCUS framework: all of the words in every possible segmentation are evaluated on the basis of their constituent syllable n-grams, and the segmentation with the highest combined score is chosen. In the current implementation of the syllable n-gram feature, n-gram scores are combined by taking the product of the conditional probabilities of all the syllable n-grams within a potential word.

When using syllable n-grams for segmenting, the segmenter must first be able to detect the syllable boundaries in the unsegmented corpus. I use a simple syllabification algorithm that could be found in nearly any elementary phonology textbook (see Algorithm 1), but the question of when to apply the algorithm arises. Is it appropriate to syllabify the entire **segmented** corpus in advance and then remove the word boundaries? Or is it necessary to syllabify straight from

[8]Daland (2009) and others have used this subsequence take on joint bigram probabilities for segmentation, but crucially in those models the segmenter inserts boundaries between phonemes that have low probability. They do not calculate a well-formedness score for the words that the segmentation generates (i.e., the probabilities are considered separately and do not have to be combined).

the unsegmented corpus, because the other approach will have every word boundary marked by a syllable boundary? As the goal for our model is to pull out phonological words, which consist only of entire syllables (Dixon and Aikhenvald, 2002), and syllables are readily identifiable from the audio signal (Villing et al., 2004), then it seems reasonable to syllabify the corpus in advance. However, if the true syllable boundaries are known in advance, an entirely different baseline strategy would be appropriate, where the task is to find and remove the extra boundaries, rather than to insert ones where there are none. For example, if every syllable boundary is treated as a word boundary, this simple model will yield 77.18% word F_1 on the Bernstein-Ratner (1987) corpus discussed in Section 2.2.1, as all but 20% of the syllable boundaries are also word boundaries. However, Algorithm 1 performs poorly when trying to syllabify the entire utterance at once, because it is intended for individual words. Syllabifying the corpus in advance with this algorithm, and then treating every syllable boundary as a word boundary only yields a word F_1 of 46.42% on the same corpus. Therefore, giving the segmenter the true syllable boundaries to start with seems overly generous to the segmenter, and I implement the syllable n-grams such that potential words get syllabified when they are scored by the syllable n-gram evaluator.[9]

Algorithm 1 Syllabification

```
Posit syllables for all vowels in word.
while There are consonants in word that are not assigned to syllables do
  For every syllable, add the first unassigned consonant to its left (if any) to the syllable.
  For every syllable, add the first unassigned consonant to its right (if any) to the syllable.
end while
```

Require Syllabic Constraint

As I want to directly eliminate any potential words that do not have at least one syllable, I implement a "require syllabic" constraint as a bit feature. That is, if a word contains at least one syllabic sound, it is given a score of one and zero otherwise. The Word Score Combiner then multiplies the scores of the other features by the "require syllabic" score, which prevents segmentations with words that do not consist of at least one syllable from being selected.

3.2.3 Evaluation Metrics

As a general guide to a segmenter's performance, we used a standard metric: a combination of PRECISION and RECALL, known as the F_1 SCORE. Precision (also known simply as ACCURACY in the cognitive science community) is the percentage of items identified that are correct. Recall (also known as COMPLETENESS) is the percentage of correct items identified. To illustrate the difference between these two measures, a segmentation system could achieve a boundary precision of 100% by simply inserting one correct boundary into the entire corpus, because 100% of the boundaries it inserted would be correct (although lacking all others). On the other hand, a segmentation model could achieve a boundary recall of 100% by inserting word boundaries between every phoneme in

[9]However, I do smooth the syllable n-gram model by getting a list of all syllables types in the segmented corpus, and using the list to initialize the counts of all possible syllable n-grams to a small constant number (0.0001 in the experiments below).

the corpus, because it would insert all of the correct boundaries (in spite of many extras). It is clear that neither precision nor recall is sufficient, and so the harmonic mean (F_1) is used.[10] We follow earlier researchers in reporting precision, recall and F_1 scores for word identification (as opposed to boundary) since words are the ultimate goal of the segmentation process (Brent, 1999; Goldwater, 2007).

In addition to F_1, knowing the kinds of errors the segmenters make can be very informative. This is because the contrasts among the outputs of different segmenters are not obvious from just F_1, and since we are comparing the utility of various sub-word features, being able to determine the benefits of each feature is important. To that end, a segmenters' errors can be classified into three natural classes: over-segmentations, under-segmentations, and mixed errors. Consider the utterance "you see the doggy" [#ju#si#ðə#dɔgi#]. OVER-SEGMENTATION ERRORS are those when the segmenter segments a true word into multiple words (e.g., the segmenter segments [dɔgi] as [#dɔ#gi#]). UNDER-SEGMENTATION ERRORS are those when the segmenter segments a sequence of true words as a single word (e.g., the segmenter guesses [#ðədɔgi#] is a single word). MIXED ERRORS are those when the segmenter segments a word which is both under-segmented and over-segmented (e.g., [#ədɔg#i#]).

3.2.4 Experiment 1: Supervised

To see how the phoneme n-gram and syllable n-gram sub-word features contributed to the segmentation process with nearly perfect knowledge of the true corpora, we trained nine PHOCUS models on 90% of each corpus: three phoneme n-gram models (varying n from 1 to 3) without the "require syllabic" constraint, three with the "require syllabic" constraint, and three syllable n-gram models (again, varying n from 1 to 3) with the "require syllabic" constraint. There was no need to run the syllable n-gram model without the "require syllabic" constraint, because a syllabic element is necessary to form a syllable; therefore, an n-gram model of syllables already requires that there be a syllabic sound in each word. The reason to include the "require syllabic" feature is that it can prevent over-segmentations that consist only of consonants (e.g., segmenting the plural ending "s" off a word). All models were evaluated using ten-fold cross-validation on both the Bernstein-Ratner (1987) and Sesotho (Demuth, 1992) corpora (see Section 2.2). The phoneme and syllable n-gram probabilities were calculated on the basis of the word tokens present in the training data (as opposed to the word types), so every time a word was encountered in the training data the counts for its constituent n-grams were updated. Unlike when these PHOCUS models are run without any supervision, in this experiment the n-gram counts are not updated on the test set in keeping with standard methodology for supervised model cross-validation.[11]

Results

As shown in Figure 3.7, the phoneme trigram segmenter performed substantially better than the other phoneme models on the modified BR corpus when run with ten-fold cross-validation. This is because unlike the unigram and bigram models, the trigram model can learn what sound combinations can start and end words, which means that a word with unlikely phoneme combinations

[10] $F_1 = \frac{2 \times precision \times recall}{precision + recall}$.

[11] This does not noticeably harm the models' performance, as can be seen by comparing the F_1 for each model in this experiment to the 90% training trial in Experiment 2.

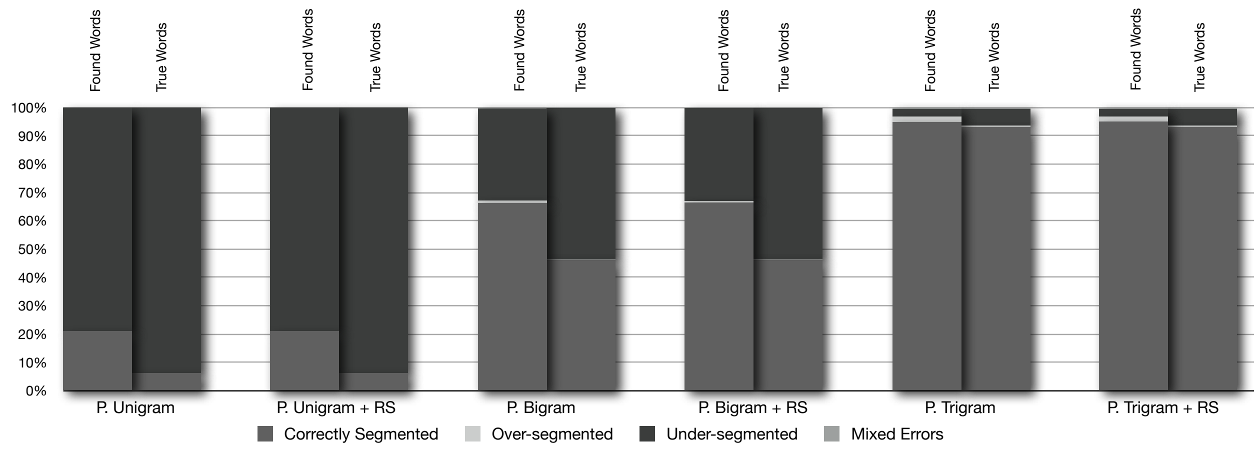

Figure 3.7: Found word and true word errors for phoneme models on modified Bernstein-Ratner corpus. "Correct" for found words is precision, and "correct" for true words is recall.

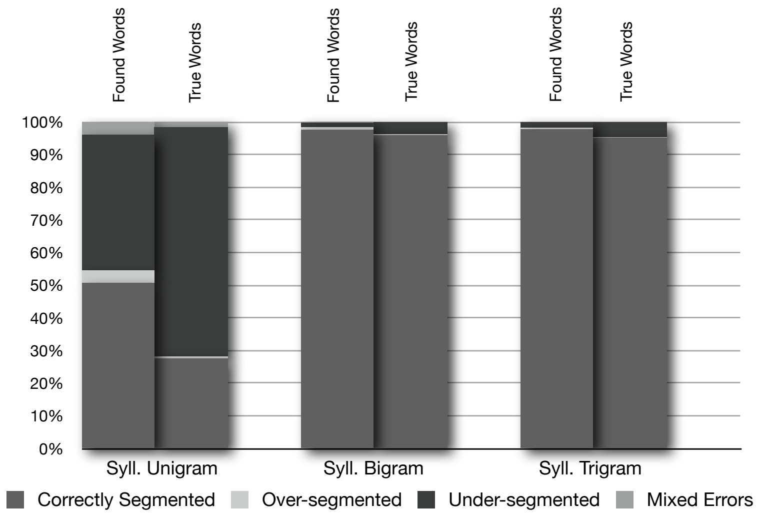

Figure 3.8: Found word and true word errors for syllable models on modified Bernstein-Ratner corpus. "Correct" for found words is precision, and "correct" for true words is recall.

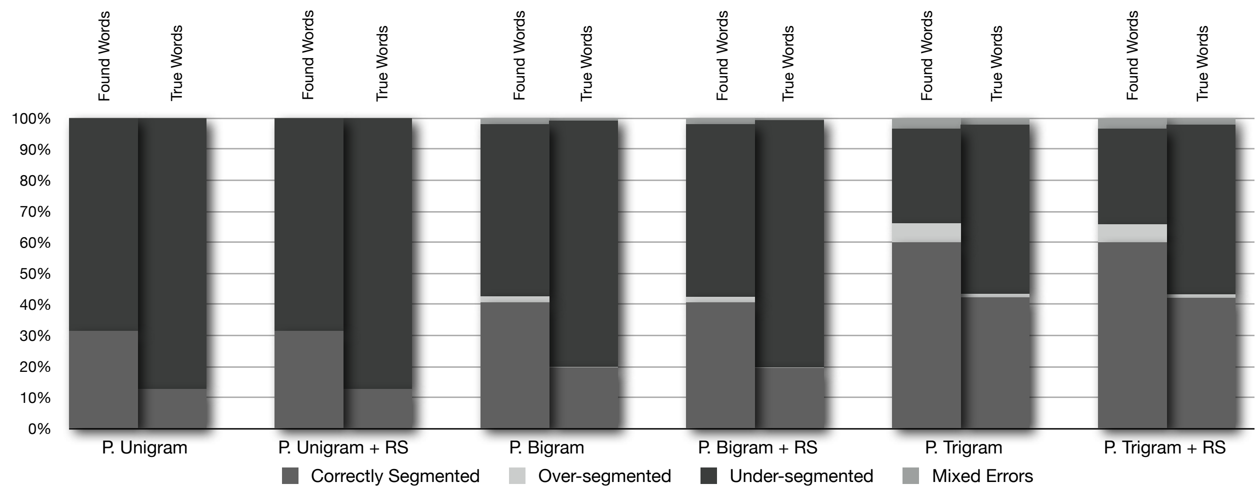

Figure 3.9: Found word and true word errors for phoneme models on Sesotho corpus. "Correct" for found words is precision, and "correct" for true words is recall.

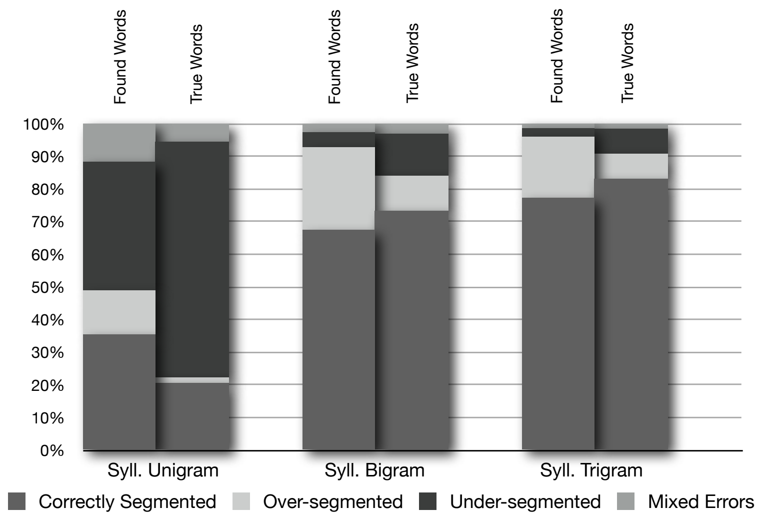

Figure 3.10: Found word and true word errors for syllable models on Sesotho corpus. "Correct" for found words is precision, and "correct" for true words is recall.

at the boundaries will not be selected. The bigram model comes close to this, but it can only learn what individual sounds start and end words, as the word boundary symbol is treated as a phoneme. The phoneme unigram model contributes nothing to the segmentation process when used without a lexicon; no word boundaries were inserted into the corpus whatsoever. When examining the errors the phoneme models make, we see that the unigram and bigram models almost exclusively under-segment, while the trigram model over-segments only slightly less than it under-segments. Furthermore, Figures 3.9 and 3.10, show that the performance of all the segmenters suffers greatly when moving to the Sesotho corpus (Demuth, 1992; Johnson, 2008a). The reasons for these performance differences are discussed in the next section.

Analysis

With the phoneme unigram model, longer novel words are preferred to shorter ones. This is because the unigram scores are multiplied together to yield the segmentation score, and adding a word boundary to a segmentation makes it one "phoneme" longer (e.g., [#wʌts#ðæt#] is one phoneme longer than [#wʌtsðæt#]). Consequently, any segmentations with word boundaries have lower scores than those without them, and the segmenter does not insert a single boundary into the corpus.

The large number of under-segmentation errors the phoneme bigram model makes are a consequence of the bigram model only inserting boundaries between pairs of phonemes that do not occur in the training input. This happens because, much like with the unigram model, adding an extra word boundary makes a segmentation contain an extra bigram, and as the bigram scores are multiplied together to yield the segmentation score, the total segmentation score for the longer segmentation will usually be lower. For example, two possible segmentations for "hi mommy" [haɪmami] are [#haɪmami#] and [#haɪ#mami#]. As word scores and bigram scores are both multiplicatively combined, the score for each segmentation is simply the product of the bigram probabilities in the segmentation. Therefore, if we cross-out all common terms in the segmentation scores for [#haɪmami#] and [#haɪ#mami#], we see that the only terms that remain are $P(\text{m}|\text{ɪ})$ for [#haɪmami#] and $P(\#|\text{ɪ}) \times P(\text{m}|\#)$ for [#haɪ#mami#]. Unless $P(\#|\text{ɪ}) \times P(\text{m}|\#)$ is greater than $P(\text{m}|\text{ɪ})$, [#haɪmami#] will be the highest ranking segmentation, and with the BR corpus $P(\text{m}|\text{ɪ})$ is the higher probability. The bigram segmenter only over-segments in cases where there is a multi-syllabic word that contains extremely rare phoneme combinations that occur at syllable boundaries. For example, "bedspread" [#bɛdsprɛd#] is the only word in the corpus that contains sequence [ds], and [d] and [s] are both frequently seen at word boundaries, so the segmenter over-segments "bedspread" [#bɛdsprɛd#] as "bed spread" [#bɛd#sprɛd#]. As this is a rare problem, all but a few of the errors the phoneme bigram segmenter makes are under-segmentations.

Much like the phoneme bigram model, the phoneme trigram model only inserts boundaries to eliminate trigrams that did not occur in the training data; however, the phoneme trigram model is much more successful segmenting the corpora. This is because most of the trigrams in the training data exclusively occur within words, whereas more of the bigrams occurred both within and between words. For instance, "hi mommy" [haɪmami] is correctly segmented in the phoneme trigram case because [ɪma] never occurred in the training data, but was incorrectly segmented in the bigram case because [ɪm] did occur. The trigram model makes both over- and under-segmentation errors, but both only occur in cases of rare phoneme sequences. On the over-segmentation side, "muffin"

[#mʌfɪn#] is incorrectly broken up into [#mʌf#ɪn#] because the trigram [ʌfɪ] never occurred in the training data. The opposite type of rarity causes under-segmentations like “I don’t” [#aɪ#dont#] as [#aɪdont#]. In this case, a very rare trigram, [ɪdo] occurred once in the training data, but that prevents the boundary from being inserted as the segmentation with more trigrams will score lower due to the multiplicative way the word scores are combined. In summary, the phoneme unigram and bigram models greatly under-segment the corpus because segmentations with more word boundaries contain more phonemes, whereas the trigram model only errs when it encounters relatively rare phoneme combinations that are well-formed in English.

It is interesting to note that the “require syllabic” feature did not improve the performance of the phoneme n-gram segmenters examined in this experiment, but there is a simple explanation for this result. The phoneme unigram and bigram models exclusively under-segmented the corpus, which is not something the “require syllabic” feature can fix, as it only prevents the over-segmentation of consonants. The phoneme trigram case is the only one where any significant amount of over-segmenting occurred, but none of these were words that did not contain any syllabic sounds. This is most likely because the segmenter did not have a lexical component, and it is word spotting that often leads to the over-segmentation of non-syllabic affixes such as [s] and [z] in English.

As shown in Figure 3.8, the syllable bigram and trigram models have roughly the same word F_1 on the modified BR corpus, and both have over twice the word F_1 of the unigram model. This is because the syllable unigram model suffers from the same problem as the phoneme unigram model: it cannot learn which sound combinations at word boundaries are well-formed. However, unlike the phoneme unigram model, it still inserts some word boundaries because it ranks words made up of more frequent syllables higher than those with less frequent ones. For example, if the segmenter encounters the words “let’s eat” [lɛts it], that segmentation will be preferred over “let seat” [lɛt sit], because “eat” [it] is a more common syllable than “seat” [sit]. The syllable bigrams and trigrams can learn what syllables are appropriate at word boundaries, and perform roughly the same, and both mainly under-segment, but for very different reasons than their phoneme counterparts. Although the syllable bigram and trigram models have the same problem as the phoneme models in that adding a word boundary to a segmentation makes it have an extra n-gram in it, this does not manifest itself in any significant way. As most of the words in BR corpus (84%) are monosyllabic, most syllable n-gram never occur in the training data, so segmentations which contain even infrequent n-grams (e.g., “#strings” [#.stɹɪŋz]) that have been seen will be preferred over those with unseen bigrams (e.g., “those.strings” [ðoz.stɹɪŋz]). There are only two cases in which the syllable bigram and trigram models will under-segment: (1) when a syllable n-gram occurs in the training data and is then seen in the test set as two (or three, for trigrams) separate words (e.g., “along” [#.ə.lɔŋ.#] occurs in the training data, but then the phrase “a long” [#.ə.#.lɔŋ.#] is encountered at test), and (2) when a word has never been encountered in the training data. For example, if “strings” [#.stɹɪŋz.#] did not occur in the training data, then the bigrams [#.stɹɪŋz] and [stɹɪŋz.#] (or the trigram [#.stɹɪŋz.#]) will not have either, so any segmentation that gloms [stɹɪŋz] onto another word (e.g., “those.strings” [#.ðoz.stɹɪŋz.#]) will be preferred over one with [stɹɪŋz] by itself, because it will consist of fewer n-grams. The only over-segmentations the syllable n-gram models make are when they encounter words with unseen n-grams that contain n-grams which start and end words. One such case is “answer” [#.æn.sɹ̩.#] getting over-segmented as “an swer” [#.æn.#.sɹ̩.#] because [æn] frequently occurs as its own word and [sɹ̩] starts words (e.g., “circles” [#.sɹ̩.kl̩z#]). In summary, the syllable bigram and trigram models perform about the same and mostly under-segment.

As shown in Figures 3.9 and 3.10, the performance of all the segmenters suffers greatly when moving to the Sesotho corpus (Demuth, 1992; Johnson, 2008a). In particular, the top-performing segmenters have many more over-segmentation errors than with the Bernstein-Ratner corpus, which is largely due to the prevalence of words consisting solely of vowels in the Sesotho corpus. For example, [e] is the second most frequent word in the corpus, and it is short enough that the phoneme trigram model stores the entire word including word boundaries, which leads to over-segmentation problems. The syllable model suffers from the same sort of problem, as it learns that [e] is very good to both begin and end words. While the syllable trigram model also frequently over-segments [e], overall it segments more accurately than the other because it can learn more of the long syllable patterns present in Sesotho. Sesotho is an agglutinative language, which means that words in it may be very long (e.g., there are 919 five-syllable word tokens in the corpus and 1 ten-syllable word). These makes encountering single-vowel words like [e] in the training data especially problematic in Sesotho, because there are even more syllables to opportunities to over-segment these long words. Furthermore, because Sesotho is agglutinative, the corpus has nearly three times as many word types as the BR corpus (3870 versus 1321), so the syllable or phoneme n-gram models are much less likely to have learned all of the relevant patterns in the training data.[12] This in turn leads to the vowels being extracted from long words consisting of unfamiliar syllable and phoneme n-grams. Overall, it does not seem like any of the sub-word features examined here are individually sufficient for successfully segmenting Sesotho; although, the syllable trigram model comes closest with a word F_1 of close to 80%.

This experiment examined how the sub-word features discussed previously perform in ideal scenarios where they have plenty of training data. I have shown that both phoneme and syllable unigram models are very uninformative for inserting word boundaries even with large amounts of training data, because adding a word boundary to a segmentation greatly penalizes the unigram score for that segmentation. This lengthening problem is also present in the phoneme bigram and trigram models, but to a lesser degree, as boundaries are still inserted between pairs or triples of phonemes that did not occur in the training data, and many of the n-grams that would occur if two words were under-segmented do not occur in the training data. In general, the syllable n-gram models outperform the phoneme models, partly because potential words need to contain syllable n-gram that occurred in the training data, and under-segmentation often leads to incorrect syllabification (e.g., "what's that" [#wʌts#ðæt#] would be syllabified as [#.wʌt.sðæt.#] if it were under-segmented). Taking both word F_1 and the types of errors each model makes into account, the phoneme and syllable trigram models performed the best on both corpora.

3.2.5 Experiment 2: How Much Training Is Necessary?

To get an idea of how much training data is necessary for the sub-word features to be useful, we conduct another experiment with the following departures from the previous. First, the segmenters are trained on progressively smaller portions of the corpora. That is, instead of training on nine subsets and testing on one, the number of training subsets varies from nine to one, while the test set

[12] It is worth investigating whether these problems can be overcome with more training data, as the Johnson (2008a) version of the (Demuth, 1992) Sesotho corpus we use is only a subset of the entire corpus. Johnson wanted the corpus to be roughly the same length as the BR corpus.

consists of the remainder.[13] The models are also run once without any training data in a completely unsupervised fashion. Second, as we are measuring the way performance changes over many trials, in Figures 3.11 and 3.12 we only report word F_1 scores in the graphs for this experiment. Third, we only look at the phoneme n-gram segmenters when they also have the "require syllabic" feature. This is not only to make the graphs easier to follow, but also because the "require syllabic" feature makes little difference in F_1—except when the segmenters have a lexicon (see Blanchard et al. (2010)), which they do not in these experiments. Finally, for consistency with the unsupervised trial, the n-gram models are allowed to update their counts during testing (i.e., if an n-gram is encountered during test, its count is incremented once per occurrence). With the exception of the aforementioned changes, the setup for this experiment is identical to that of Experiment 1: the probabilities for the syllable and phoneme n-grams are still calculated as described in Section 3.2.1, the lexicon is disabled, and the models are run on the BR and Sesotho corpora.

Results

As can be seen in Figure 3.11, all of the models have relatively stable performance on the modified BR corpus for the semi-supervised trials; however, when the models are run completely unsupervised, there is a large drop in word F_1 score for all but the phoneme and syllable unigram segmenters.

The results of running the segmenters with progressively smaller amounts of training data on the Sesotho corpus are shown in Figure 3.12. The rates of the performance drops on Sesotho seem very similar to those of the ones on the BR corpus, with the exception of the syllable bigram and trigram segmenters.

Analysis

As can be seen in Figure 3.11, all of the models have relatively stable performance on the modified BR corpus for the semi-supervised trials; however, when the models are run completely unsupervised, there is a large drop in word F_1 score for all but the phoneme and syllable unigram segmenters. The phoneme unigram model has exactly the same performance as in the supervised trials because it never inserts any word boundaries, so it cannot get any worse. On the other hand, the syllable unigram model only sees a modest drop in F_1 score because, as was described in the previous experiment's results, it only inserts boundaries to make words out of syllables it has already seen, and it can still acquire syllables from unsegmented utterances, albeit less accurately. The results for the other models are not quite as straightforward, and are discussed below.

The syllable bigram and trigram models see the largest drop in word F_1 for two reasons. First, as they have no reason to insert boundaries early on, they learn that illegal word-internal syllable n-grams (e.g., "put.that" [pʊt.ðæt]) are actually good, and these initial mistakes compound on each other. Second, the syllabification algorithm does not work well on unsegmented corpora—it is meant for use on words, not utterances—and once the models have started to learn incorrect syllables that can start and end words, the errors compound on one another.

[13]In this approach models are being tested on progressively larger sets. In the future we may rerun this experiment with the segmenters trained on the first k sets, then run unsupervised with count updating enabled on all but the final set, and finally tested on the final set. Thus, the test sets would all be the same size while still allowing each model to get extra unsupervised training.

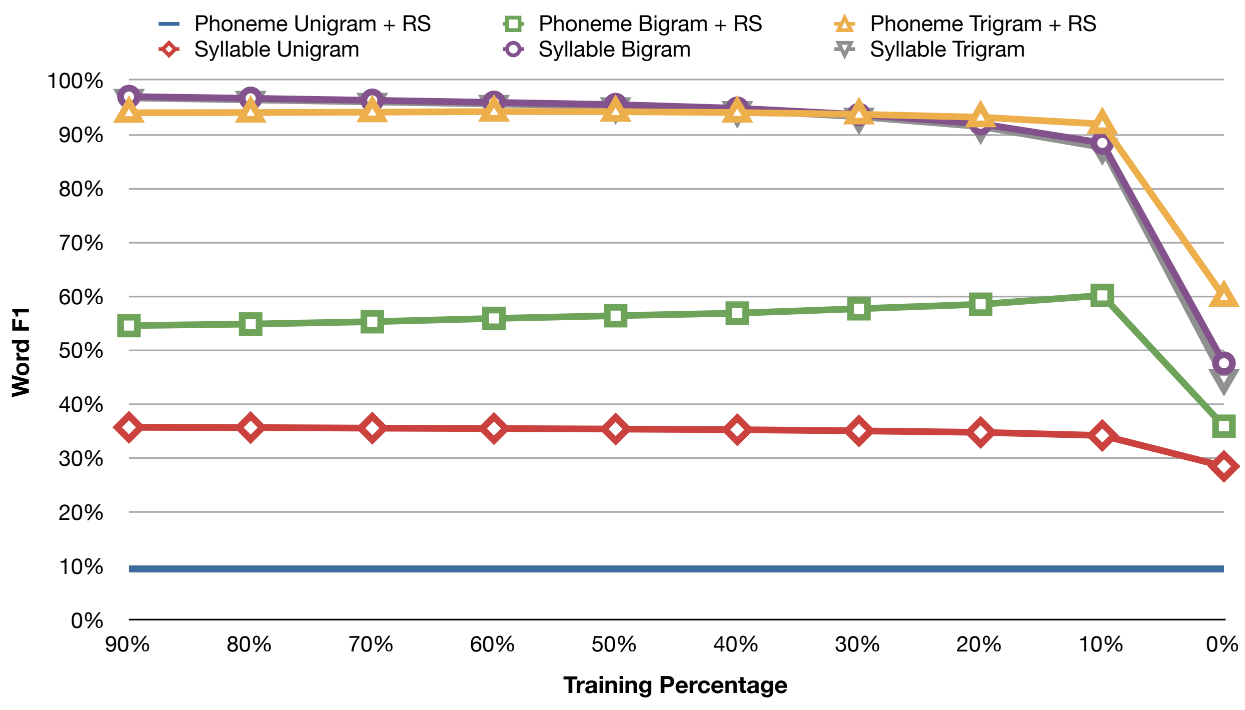

Figure 3.11: Word F_1 for segmenters on modified BR corpus.

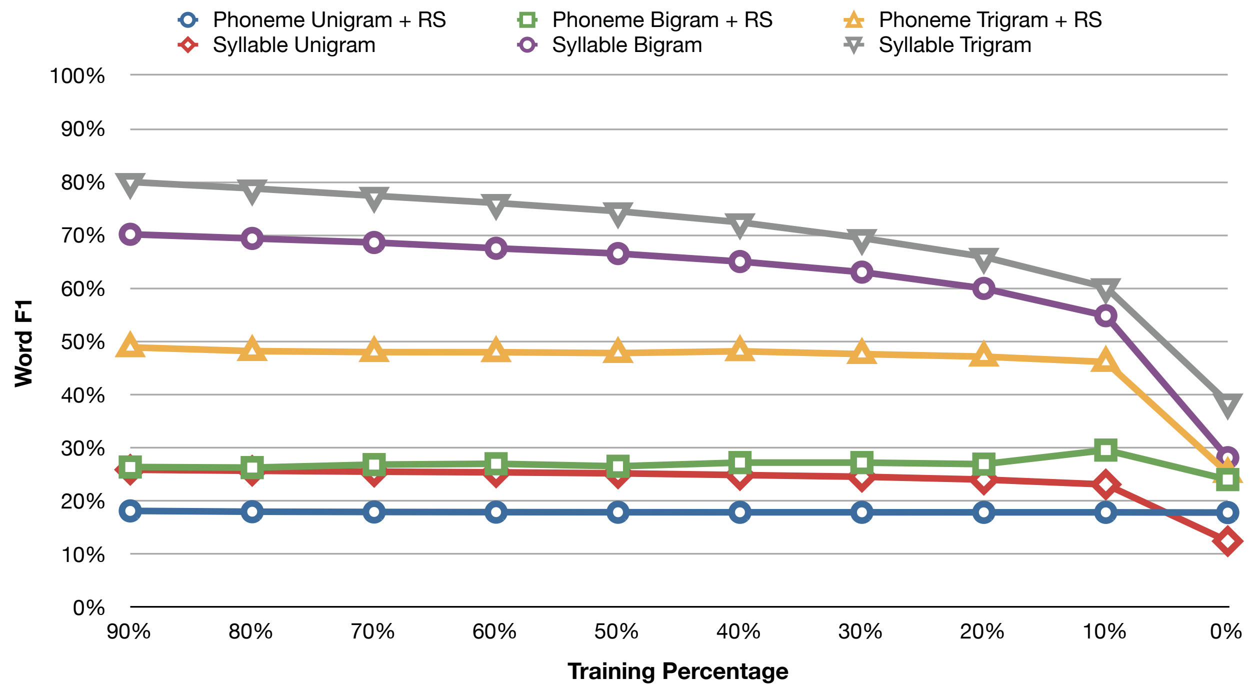

Figure 3.12: Word F_1 for segmenters on Sesotho corpus.

The way the syllable n-gram models are initialized becomes important for the unsupervised trial. As we cannot enumerate all logically possible syllables of a given language—there are an infinite number of them—the method we chose to create a uniform distribution over the syllable n-grams was to first find all of the syllables in the segmented corpus—there are 1178 in the BR corpus—and then calculate what the total number of n-grams at each level should be assuming all logically possible n-grams of the length we are concerned with are given a small constant count (0.0001). We could not directly enumerate all of the possible syllable n-grams as there are 1.6 billion possible trigrams consisting of the 1178 syllables in the BR corpus. Instead, any time the segmenter finds a word that consists of a syllable n-gram it has never seen before, the n-gram is assigned a probability equal to the small constant count for that length n-gram divided by the count of its $n-1$ length prefix, which may also be a larger constant number if the prefix has not been seen before. For example, if the first trigram encountered is "#cat#" [#.kat.#], it will be given a probability of roughly $\frac{0.0001}{0.2348}$. The problem lies in the fact that the model may also examine a segmentation that contains a bad syllable (i.e., one that did not occur in the segmented corpus), and that too will be given the same probability as a good unseen syllable. Hence, the number of syllables encountered will be greater than the number used to initialize this counts, which means the initial uniform distribution is not probabilistically sound, as it does not necessarily add up to one. However, the current method for initialization was the most reasonable approach at the time of the experiment.

In the semi-supervised trials, the phoneme bigram segmenter actually sees a slight improvement in word F_1 the less training data it is given. This is a counterintuitive result, but the cause is rather straightforward. Some phoneme bigrams in the corpus occur very rarely within words, but frequently across word boundaries, and as the segmenter only inserts boundaries between unseen bigrams, when these rare sequences are encountered in a word in the training data, the segmenter is prevented from inserting boundaries between the bigram later. For example, the sequence [əp] only occurs in the BR corpus in the words "apart" [əpart] and "supposed" [səpozd], but it occurs 146 times across word boundaries (e.g., "the pants" [ðə#pænts]), so when there is more training data, the segmenter is more likely to have encountered "apart" or "supposed." Thus, less training data is actually helpful for the phoneme bigram model for corpora that do not have a strict bimodal distribution of bigrams where all bigrams occur either within or across word boundaries.

As shown in Figure 3.11, all of the models except the phoneme unigram model (whose performance cannot get worse) see substantial drops in performance in the unsupervised trial. This is because the initial probability estimates the models make are unreliable. Since all n-grams (including unseen ones) are given an initial small constant count (0.0001) to ensure the models are probabilistically sound, any n-gram that has been seen even once is given a disproportionately high score in the beginning. If these initial n-grams are actually mistakes, then these early errors will beget many more later errors. For example, the first utterance that has a boundary inserted into by the phoneme bigram and trigram models is "You want to look at this?" [ju#want#tu#lʌk#æt#ðɪs#], which is broken up as [juwanttulʌk#ætðɪs#]. It is broken up this way because the first utterance in the corpus is "You want to see the book?" [ju#want#tu#lʌk#æt#ðə#bʌk#], so all of the n-grams that make up "Youwantto" [juwanttu] have high probabilities, and so do those that make up word-final "ook" [ʌk#]. While this may seem like a good thing, this is only the fourth utterance in the corpus, thus the segmenters are overconfident in the evidence they have gathered over just a few utterances. Considering that most of the utterances are completely unsegmented initially because a majority of the n-grams are unseen, the segmenters incorrectly learn that phoneme combinations that only occur across word boundaries are valid. For instance, “You want to” [ju#want#tu] is under-segmented throughout the entire corpus. One of the only reasons this does not quickly spiral out of control leading to the segmenters under-segmenting nearly every utterance is that infant-directed speech contains a relatively high frequency of one-word utterances (21% of the utterances in the BR corpus). This aids the segmenters in learning which phoneme sequences are acceptable at word boundaries without learning too many invalid sequences (as they would if there were only multi-word utterances in the corpus). Nevertheless, the segmenters do learn a variety of incorrect sequences initially (e.g., [tt] from under-segmenting “You want to” [ju#want#tu] as [juwanttu]) which adversely affects their performance throughout the rest of the corpus, causing the large drop in performance from the semi-supervised trials.

The results of running the segmenters with progressively smaller amounts of training data on the Sesotho corpus are shown in Figure 3.12. The rates of the performance drops on Sesotho seem very similar to those of the ones on the BR corpus, with the exception of the syllable bigram and trigram segmenters. Their F_1 plummets compared to the other models, and this is again due to the prevalence of single-vowel words such as [e] in Sesotho combined with the syllable trigrams ability to act as a lexicon for monosyllabic words. The less training data the models have, the more of an impact this problem makes, because they will know fewer correct syllable trigrams, but still learn [e] is a very likely word.

In this experiment we have demonstrated what the effects of reducing the amount of training data for the previously examined segmenters are. In general, there were relatively drops in word F_1 when moving from 90% to 10% training data were relatively small, with the exception of the syllable bigram and trigram models on the Sesotho corpus. Their F_1 declined steadily with each reduction in training data as a result of increasing over-segmentation of single-vowel words like [#e#]; when there are fewer training examples with syllable trigrams containing [e], more words containing [e] are over-segmented. The phoneme bigram model on the BR corpus anomalously saw an increase in performance as the training data was reduced, but this was due to the BR corpus having some phoneme pairs that occur rarely within words but frequently between words. As for the unsupervised trial’s results, performance dropped substantially on both corpora for all but the phoneme unigram model, and this exception was simply due to the phoneme unigram model not inserting any boundaries even with training data. On the Sesotho corpus, none of the models got higher than 38% word F_1 during the unsupervised trial, hence none of them can be considered adequate for segmenting sesotho. Conversely, the phoneme trigram model on the BR corpus had a moderately high word F_1 (60%) during the unsupervised trial, which is impressive since trigram models usually suffer from data sparsity issues when given such little training data. In summary, only the phoneme trigram model maintained a word F_1 scores of at least 60% in the unsupervised trial on the BR corpus, whereas none of the models exceeded 38% F_1 on the Sesotho corpus.

3.2.6 Overall Results

As is shown in Table 3.2.6, it seems that out of the features I have examined so far, a phoneme trigram model with the “require syllabic” constraint has the best combination of overall performance (see Experiment 1) and resilience to less training data (see Experiment 2).[14] The syllable bigram

[14]Our results do not completely line up with the results of Blanchard et al. (2010) (where the phoneme unigram and bigram models were not far behind the trigram model), but this is simply because in our current experiments

Sub-word Feature	**Advantages**	**Disadvantages**
Phoneme Unigram	No over-segmentation errors.	Inserts no word boundaries in either experiment.
Phoneme Bigram	Very few over-segmentation errors.	Word F_1 drops to 37% in unsupervised trial. 55% word F_1 score in supervised experiment.
Phoneme Trigram	94% word F_1 scores in supervised experiment. 60% word F_1 in unsupervised trial.	Makes some over-segmentation errors.
"Require Syllabic" Constraint	Prevents consonant over-segmentation.	*A priori* knowledge of language structure.
Syllable Unigram	Very few over-segmentation errors.	Inserts very few word boundaries in either experiment.
Syllable Bigram	97% word F_1 when supervised. Very few over-segmentation errors.	Word F_1 drops to 48% in unsupervised trial.
Syllable Trigram	96% word F_1 score in supervised experiment. Very few over-segmentation errors.	Word F_1 drops to 45% in unsupervised trial.

Table 3.2: Advantages and Disadvantages of Each Feature Examined

and trigram models perform slightly better in the supervised experiment, but have a 12% lower F_1 than the phoneme model in the unsupervised trial. All of the bigram and trigram models are very sensitive to early errors as they are overconfident in their early decisions.

3.A Results with Lexicon Enabled

As shown in Figure 3.13, when the models are run unsupervised on the BR corpus with the lexicon enabled, the F_1 is greatly influenced by the value chosen to smooth the unseen n-grams. When using 0.0001, as we did in Experiments 1 & 2, the performance for each model seems to decrease as the value for n is increased. This runs counter to the previous experiments, so I also ran it with the initial count set to 1, which allocates much more of the probability space to unseen n-grams. In this case, instead of seeing a drop in performance for larger window sizes, we see relatively stable performance from all phoneme and syllable n-gram models. These results underscore the importance of looking at these features in isolation in Experiments 1 & 2, as the fundamental advantages and disadvantages of the different features are completely distorted when using the lexicon. Figure 3.13 actually has the rankings of models almost completely reversed from those in Experiments 1 & 2 where the bigram and trigram models definitively outperformed the unigram models. This is because currently the models only utilize the sub-word feature scores when no lexical score is available for a word (i.e., the word is novel). These results really only show how

there was no lexicon, unlike in their experiments.

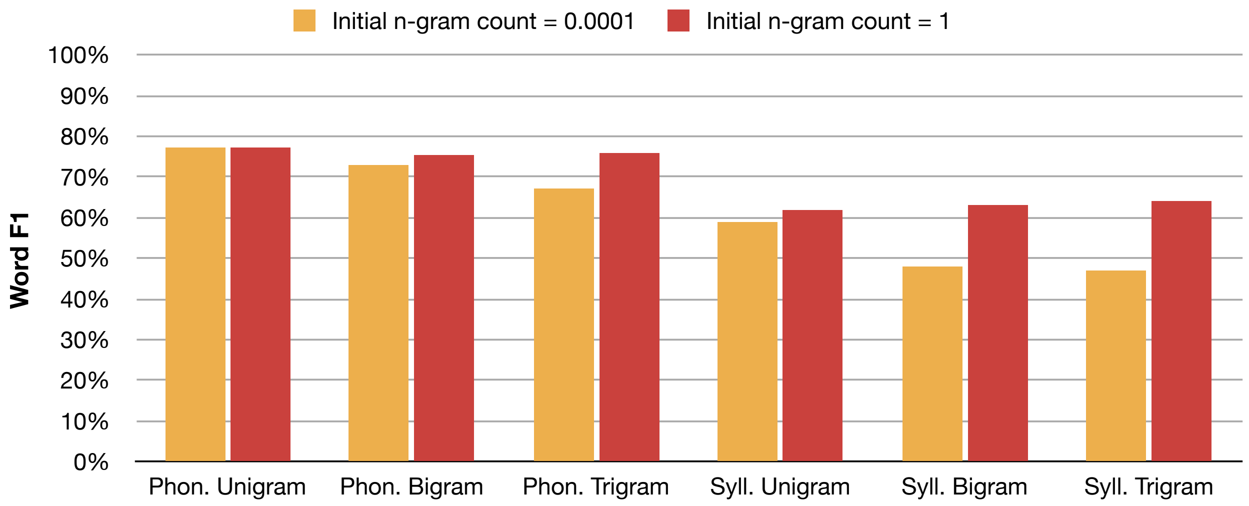

Figure 3.13: Word F_1 for segmenters on modified BR corpus.

well the lexical feature does on its own, since a lexical score is almost always available after the first few hundred utterances because the BR corpus consists of 21% one-word utterances. In my dissertation, I will develop a more principled method for combining the sub-word and lexical scores, which should alleviate some of these issues.

4 Moving Forward

In this chapter I enumerate the problems that I will attempt to solve in my dissertation, discuss some potential solutions to them, and explain how addressing these issues will make the contributions to the field outlined in Section 1.1.

4.1 Fundamental Issues

There are two fundamental issues with the current PHOCUS models that I will address in my dissertation: (1) how to handle early errors and (2) how to combine feature scores. Each issue is presented in detail below along with potential solutions.

4.1.1 Handling Early Errors

As exemplified by the errors discussed in Section 3.2, an inherent issue with taking an incremental approach to word segmentation is that any errors that a segmenter makes initially quickly compound on one another. While these compounding errors are present to some extent when the segmenters are run in a supervised fashion, when the segmenters are not given any training data these errors can have an extremely detrimental effect on segmentation accuracy when (see Section 3.2.5). Therefore, examining how to eliminate or recover from early errors will be a major part of my dissertation research.

I have a few ideas for addressing the question of how to prevent early errors in an incremental segmenter. First, if we cannot prevent the errors, it seems reasonable to try to make the model more resilient to these errors. One way to do that would be to add some sort of decay factor to the counts stored by the model, so that n-grams seen frequently earlier in the corpus but very infrequently later have their scores decremented. This would allow the model to effectively remove the invalid n-grams (e.g., [tj]) that are seen early on because of the drastic under-segmentation most of the models due in the beginning of the learning process.

Second, we could try to add a feature that is more reliable initially and use that feature exclusively until the model is confident that the statistics the other features use have stabilized. For example, the incremental version of the Bootstrap Voting Experts algorithm (Hewlett and Cohen, 2009) mentioned in Section 2.4 uses conditional entropy to determine where to insert boundaries, and Hewlett and Cohen (2009) show that this approach segments fairly accurately even in the very beginning of the process. A strategy such as theirs could be used initially to find some words, and then update the evidence for the other sub-word features on the basis of those words.

The final approach to handling early errors in an incremental segmenter I propose is to modify the Top Candidate Selector (see Figure 3.2) and the Evidence Updater (see Figure 3.1) to return and store the n best segmentations (instead of just the top one) where alternative segmentations are stored with lower weights based on their scores. This would be especially useful in the early stages of learning, because these alternative segmentations would make the model less likely to lock in to one particular way of segmenting things, thereby making the model less over-confident initially.

4.1.2 Combining Feature Scores

In my dissertation, I will conduct experiments similar to those in Section 3.2 that include additional sub-word features. One problem that arises from adding more features to the model is that it is not clear how these usually disparate probabilities should be combined. The current instantiation of the PHOCUS Word Score Combiner in the models evaluated in Section 3.2 simply uses the lexical score if it is non-zero and that component is enabled, and uses the phoneme or syllable n-gram score otherwise, but when we want to evaluate combinations of sub-word features this method will not suffice.

There are many issues to consider when choosing an appropriate method for combining the sub-word feature scores. For example, if the range or variance of the score distributions are different, then the Word Score Combiner would have to ensure that the differences in the distributions do not cause some feature scores to inherently have more of an impact on the final score than others. Furthermore, if we want the final score to be a probability, then the most common methods for combining multiple feature scores (e.g., taking a weighted sum of the scores) would not be appropriate. Determining how exactly to combine these feature scores will require significant work and will be an important part of the research for my dissertation.

4.2 Other Issues

In the course of my dissertation research, there are also some lesser problems that I will address. These include verifying more PHOCUS instantiations, creating a phonological word corpus, and correcting feature implementation issues.

4.2.1 PHOCUS Instantiations

I have established that the PHOCUS framework unifies the segmenters of Brent (1999), Batchelder (2002), Venkataraman (2001), and Blanchard et al. (2010); however, there are other segmenters (e.g., Hewlett and Cohen's (2009) incremental implementation of Voting Experts (Cohen and Adams, 2001; Cohen et al., 2007)) that should fit into the framework whose exact instantiations have yet to be determined. In my dissertation I will examine which state-of-the-art segmenters can be instantiated within the PHOCUS framework to allow for easier comparison between models. As it is unclear how one could mathematically prove that the PHOCUS instantiations of these models are correct, I will have to verify the accuracy of them by comparing the outputs from running the PHOCUS and original versions of each segmenter on the same corpus. Those with identical segmentation can be considered to be correctly instantiated. This will concretely establish that the PHOCUS framework unifies many seemingly disparate models.

4.2.2 Phonological Word Corpus

To evaluate my hypothesis that many errors made by segmenters that are searching for phonological words are simply due to them being evaluated against corpora of orthographic words, in the course of my dissertation research I will create a version of the modified BR corpus that is split into phonological words. The main change I will make is removing the breaks between unstressed function words the words they proceed. I will not add any word boundaries to the corpus, as English orthography typically does not have two phonological words without a space between them.

4.2.3 Feature-Specific Issues

As was shown in Table 3.2.6, each sub-word feature we have examined has its shortcomings, and we need to address them. Below we consider possible solutions to the issues facing the syllable and phoneme n-gram features, and I propose a new feature for handling word length.

Syllable N-gram

The feature that seems to have the most glaring issue with its implementation is the syllable n-gram one, as the counts are not initialized in a probabilistically sound manner. While one cannot have a uniform distribution over an infinite set, there are alternative initialization methods that may bring the model closer to being probabilistically sound (e.g., use the true syllable boundaries) Regardless of the specific solution we use in the end, addressing the issue of syllable n-gram count initialization will be important for being able to conclusively say that syllable n-grams work better or worse than some other sub-word feature.

Phoneme N-gram

Although it is probabilistically sound, the phoneme n-gram feature is not without its problems as well. One such issue was raised in Section 3.2.2: it would more directly measure a words well-formedness if we calculated the likelihood of the phoneme n-grams occurring within words, instead of calculating how likely one phoneme is given another.

Another shortcoming of the way our model currently approaches phoneme combinations is that there is no feature that can support long-distance dependencies (e.g., vowel harmony in languages like Finnish). Many of the worlds' languages have vowel or consonantal harmony, and words in these languages must obey these long-distance patterns to be well-formed. A straightforward way to do this is to add a probabilistic strictly-piecewise learner as proposed by Heinz (2007) (essentially, a bigram model that stores non-adjacent pairs in addition to adjacent ones).

Word Length

As both under-segmentations and over-segmentations are errors in which the lengths of the words and utterances found do not match those present in the correct segmentation, one way to try to prevent both of these types of errors would be to add or modify features to penalize words and utterances of incorrect lengths. First, we would need to factor out the inherent length penalties from the phoneme and syllable n-gram models from the multiplicative nature in which scores are

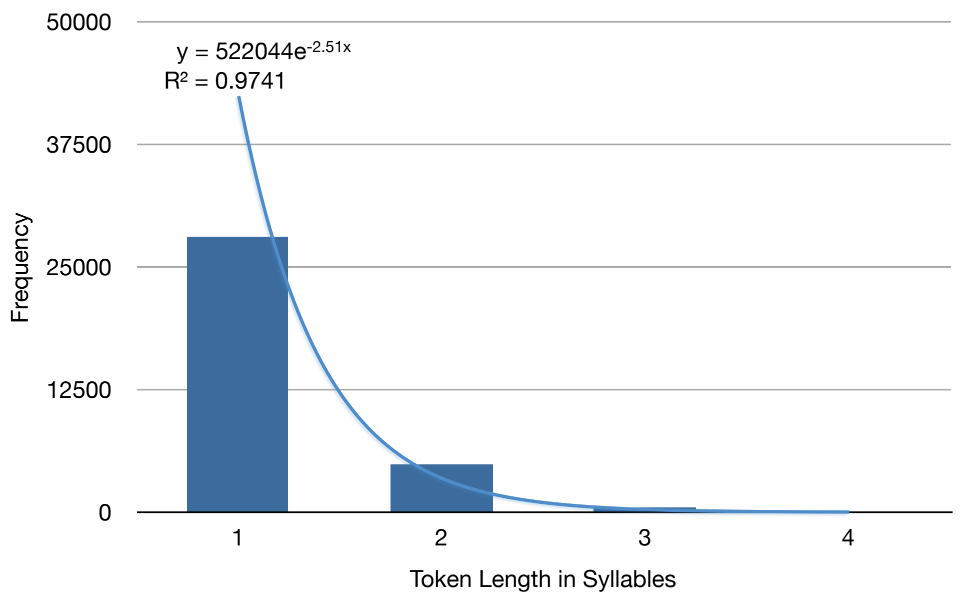

Figure 4.1: Word token length distribution for modified BR corpus.

currently combined: as more probabilities are multiplied together, the product gets smaller. One way to factor out the length penalties present in these scores would be to use the geometric mean of the probabilities—instead of just the product—by taking the nth root of the product (e.g., take the square root of the product of two probabilities).

After removing the inherent length penalties, we should add a feature that scores a word based on how likely it is to be a particular length so that one-word segmentations do not always have the highest score. The score from this feature could then be combined with the other sub-word feature scores by the Word Score Combiner (see Figure 3.5) to yield a score for each word in a potential segmentation. For the BR corpus, the distribution of word token phoneme lengths is not something that seems to be a necessary consequence of any other properties of the corpus that we can find;[1] however, the distribution of word token **syllable** lengths seems to be a nearly exponentially decreasing function peaking at one (see Figure 4.1), which could be approximated by simply multiplying the syllable unigram probabilities in each word together. Therefore, while the syllable unigram feature was not very successful in isolation, it may be useful for scoring word length in future models. With the constant inclusion of the currently implemented syllable unigram feature, and the removal of the inherent length penalties from the phoneme and syllable n-gram models, we can expect the length distribution of the segmented corpus to more closely match that of the true corpus, which should prevent both major classes of errors.

[1]It peaks at three and drops slightly on either side, and then rapidly plummets.

4.3 Contributions

My dissertation will make five major contributions to the field: a unified framework for segmentation, a thorough investigation of sub-word features, evidence that evaluating segmenters with respect to orthographic words is incorrect, methods for preventing and recovering from early errors, and a highly accurate unsupervised incremental word segmenter. Each of these contributions and the progress I have made toward making them are discussed in more detail below.

4.3.1 PHOCUS Framework

The PHOCUS framework presented in Section 3.1 is a significant contribution for the following reasons. First, as was shown in Section 2.4, there are many existing incremental and "repeated incremental" batch segmentation models that all employ different features, which makes them seem disparate; however, many can be described within the PHOCUS framework (see Section 3.1), thereby allowing the systematic investigation of the effects of changing the instantiations of the framework's components, and simplifying the process of comparing competing segmentation models. Second, by unifying seemingly disparate segmentation algorithms, the framework makes the similarities that many segmenters share more apparent. Finally, those interested in studying the acquisition of phonotactic patterns may also be interested in the framework, because the models in it can acquire phonotactic patterns from unsegmented text, instead of from words (as is the norm). I have already established that the PHOCUS framework unifies the segmenters of Brent (1999), Batchelder (2002), Venkataraman (2001), and Blanchard et al. (2010), and will verify more instantiations of the framework as described in Section 4.2.1.

4.3.2 Investigation of Sub-word Features

The main contribution of my dissertation will be establishing in principle what sub-word features are more useful than others, and the types of errors each prevents or causes. This will be particularly useful for anyone who is building an unsupervised word segmentation system, as they will know which features they should use to address particular issues with a given corpus or model. Furthermore, my work will also be the first to examine some sub-word features like long-distance phonotactic patterns, which may turn out to be beneficial for segmenting languages that have vowel harmony (e.g., Finnish). Finally, if syllable-based sub-word features (e.g., syllable n-grams) are especially useful, then my work could also be considered evidence for the necessity of the syllable, a unit whose status is often debated by phonologists.

The experiments conducted in Section 3.2 were the first pieces of my thorough investigation of sub-word features. The first experiment was conducted to get an idea of how the features work in an ideal situation and used ten-fold cross validation, and the second tested the features in progressively more realistic scenarios by training on smaller and smaller portions of the corpus before testing. In my dissertation, I will conduct similar experiments that include the additional sub-word features mentioned in the previous section: a strictly-piecewise learner, and word length evaluator. These experiments will also evaluate the combined effectiveness of different features using one of the implementations of the Word Score Combiner (see Figure 3.5) described in Section 4.1.2.

4.3.3 Phonological Word Evaluation

All corpora we are aware of that have been used for evaluating segmenters are divided into orthographic words, but my research calls these evaluations into question, which should have an impact on the methodology others use to evaluate their segmenters. As mentioned previously, models that make use of phonotactic features (e.g., Fleck, 2008; Blanchard et al., 2010; Daland, 2009) or innate knowledge of syllable structure (e.g., Johnson, 2008b) are searching for phonological words; however, these models have only been evaluated against corpora that were segmented into orthographic words, although they are phonetically transcribed ones. Therefore, my hypothesis is that many of the errors in the segmentations these models output may not be errors, but rather a result of comparing them against the wrong gold standard. For example, many of the under-segmentation errors unsupervised models make contain function words that are not actually distinct phonological words (e.g., "the" [ðə]).

To alleviate the above problems, in the course of my dissertation research I will create a version of the modified BR corpus that is split into phonological words. This corpus would be useful not only to myself but also to other researchers who study unsupervised word segmentation, as the word boundaries will no longer be arbitrarily based on the English writing system. When this new corpus is developed, I will re-run the experiments I have already conducted to see if changing the gold standard to be phonological words makes the predicted difference.

4.3.4 Early Error Recovery & Prevention

In Section 4.1.1 I outlined some of the approaches I will try in my dissertation to help prevent and recover from early errors in the unsupervised learning process. The ideas that should be applicable to other unsupervised learning scenarios are (1) adding decay to the counts of stored by the model to decrement the scores of n-grams that only occur initially, and (2) using one feature that is more reliable initially and then switching to using others once enough words have been learned. I will investigate these methods in detail in my dissertation.

4.3.5 Accurate Incremental Segmenter

The final major contribution that my work will make is a highly accurate unsupervised incremental word segmenter. The instantiation of my segmentation framework presented by Blanchard et al. (2010) is currently the most accurate unsupervised incremental segmenter on the *de facto* standard corpus for evaluation in the unsupervised segmentation community, the Bernstein-Ratner (1987) corpus. Although this segmenter's results are promising, there is still room for improvement, and in my dissertation I will use the error prevention methods mentioned previously to develop more accurate unsupervised incremental segmenters.

4.4 Conclusion

In this proposal I have outlined what the current state of unsupervised word segmentation research is, and described how determining what features are most useful for the process would be beneficial to many types of segmenters. I also examined the performance of a few sub-word features and discussed the work that I will do for my dissertation to alleviate any problems inherent to those features. Finally, I have outlined how I will make the contributions to the field proposed in the first chapter.

Bibliography

Aslin, R, Jenny R Saffran, and Elissa Newport. 1998. Computation of conditional probability statistics by 8-month-old infants. *Psychological Science.*

Batchelder, Eleanor Olds. 2002. Bootstrapping the lexicon: a computational model of infant speech segmentation. *Cognition*, 83(2):167–206.

Bernstein-Ratner, Nan. 1987. *The phonology of parent child speech*, volume 6, pages 159–174. Erlbaum, Hillsdale, NJ.

Blanchard, Daniel and Jeffrey Heinz. 2008. Improving word segmentation by simultaneously learning phonotactics. In *12th Conference on Computational Natural Language Learning*, pages 65–72. Association for Computational Linguistics, Morristown, NJ.

Blanchard, Daniel, Jeffrey Heinz, and Roberta Michnick Golinkoff. 2010. Modeling the contribution of phonotactic cues to the problem of word segmentation. *Journal of Child Language*, 37.

Bortfeld, Heather, James Morgan, Roberta Golinkoff, and Karen Rathbun. 2005. Mommy and me: Familiar names help launch babies into speech-stream segmentation. *Psychological Science*, 16(4):298–304.

Brandl, Holger, Britta Wrede, Frank Joublin, and Christian Goerick. 2008. A self-referential child-like model to acquire phones, syllables and words from acoustic speech. *Proceedings of the 7th IEEE International Conference on Development and Learning*, pages 31–36.

Brent, Michael. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Brent, Michael and Timothy Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61(1-2):93–125.

Brent, Michael and Jeffrey Siskind. 2001. The role of exposure to isolated words in early vocabulary development. *Cognition*, 81:B33–B44.

Cairns, Paul, Richard Shillcock, Nick Chater, and Joe Levy. 1997. Bootstrapping word boundaries: A bottom-up corpus-based approach to speech segmentation. *Cognitive Psychology*, 33:111–153.

Cheng, Jimming and Michael Mitzenmacher. 2005. The markov expert for finding episodes in time series. *Proceedings of the Data Compression Conference.*

Chomsky, Noam and Morris Halle. 1965. Some controversial questions in phonological theory. *Journal of Linguistics*, 1:97–138.

Christiansen, Morten H, Joseph Allen, and Mark S Seidenberg. 1998. Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes*, 13(2/3):221–268.

Christiansen, Morten H, Christopher Conway, and Suzanne Curtin. 2005. Multiple-cue integration in language acquisition: A connectionist model of speech segmentation and rule-like behavior. *Language Acquisition*, pages 1–39.

Cohen, Paul and Niall Adams. 2001. An algorithm for segmenting categorical time series into meaningful episodes. *Lecture notes in computer science*, pages 198–207.

Cohen, Paul, Niall Adams, and Brent Heeringa. 2007. Voting experts: An unsupervised algorithm for segmenting sequences. *Intelligent Data Analysis*, 11(6):607–625.

Cole, Ronald and Jola Jakimik. 1980. *A model of speech perception*, pages 136–163. Lawrence Erlbaum Associates, Hillsdale, NJ.

Daland, Robert. 2009. *Word Segmentation, Word Recognition, and Word Learning: A Computational Model of First Language Acquisition*. Ph.D. thesis.

de Marcken, Carl. 1995. Acquiring a lexicon from unsegmented speech. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 311–313.

de Marcken, Carl. 1995. The unsupervised acquisition of a lexicon from continuous speech. *Massachusetts Institute of Technology*.

Demuth, Katherine. 1992. *Acquisition of Sesotho*, volume 3, pages 557–638. Lawrence Erlbaum Associates, Hillsdale, NJ.

Dixon, R. M. W. and Alexandra Y. Aikhenvald. 2002. *Word: a typological framework*, pages 1–41. Cambridge University Press, Cambridge, UK.

Elman, Jeffrey L. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Fleck, Margaret M. 2008. Lexicalized phonotactic word segmentation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 130–138. Association for Computational Linguistics, Morristown, NJ.

Gambell, Timothy and Charles Yang. 2004. Statistics learning and universal grammar: Modeling word segmentation. *First Workshop on Psycho-computational Models of Human Language Acquisition*, page 49.

Goldsmith, JA. 2009. Segmentation and morphology. *Blackwell Computational Linguistics and Natural Language Processing Handbook*, pages 1–41.

Goldwater, Sharon. 2007. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Brown University, Department of Cognitive and Linguistic Sciences.

Goldwater, Sharon, Thomas L Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112:21–54.

Halle, Morris. 1978. *Knowledge unlearned and untaught: What speakers know about the sounds of their language*, pages 294–303. MIT Press, Cambridge, MA.

Harris, Zellig. 1954. Distributional structure. *Word*, 10(2/3):146—162.

Heinz, Jeffrey. 2007. *Inductive Learning of Phonotactic Patterns.* Ph.D. thesis, University of California, Los Angeles, Department of Linguistics.

Hewlett, D and Paul Cohen. 2009. Bootstrap voting experts. *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI).*

Hockema, Stephen A. 2006. Finding words in speech: An investigation of american english. *Language Learning and Development*, 2(2):119–146.

Iwahashi, Naoto. 2006. Robots that learn language: Developmental approach to human-machine conversations. *Lecture Notes in Computer Science: Symbol Grounding and Beyond*, 4211/2006:143–167.

Johnson, Mark. 2008a. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of Association for Computational Linguistics SIGMORPHON*, pages 20–27. Association for Computational Linguistics, Morristown, NJ.

Johnson, Mark. 2008b. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 398–406. Association for Computational Linguistics, Morristown, NJ.

Jurafsky, Daniel and James Martin. 2008. *Speech and Language Processing.* Prentice-Hall, second edition.

Jusczyk, Peter, Elizabeth Hohne, and Angela Baumann. 1999a. Infants' sensitivity to allophonic cues for word segmentation. *Perception & psychophysics*, 61(8):1465–1476.

Jusczyk, Peter, Derek Houston, and Mary Newsome. 1999b. The beginnings of word segmentation in english-learning infants. *Cognitive Psychology*, 39:159–207.

Lu, Xiaofei. 2006. *Hybrid Models for Chinese Unknown Word Resolution.* Ph.D. thesis, Ohio State University, Department of Linguistics.

MacWhinney, B and C Snow. 1985. The child language data exchange system. *J Child Lang*, 12(2):271–295.

Matthews, Peter. 1991. *Morphology.* Cambridge University Press, second edition.

Mattys, Sven and Peter Jusczyk. 2001. Phonotactic cues for segmentation of fluent speech by infants. *Cognition*, 78:91–121.

Miller, Matthew and Alexander Stoytchev. 2008. Hierarchical voting experts: An unsupervised algorithm for segmenting hierarchically structured sequences. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1820–1821.

Miller, Matthew and Alexander Stoytchev. 2009. An unsupervised model of infant acoustic speech segmentation. *Proceedings of 9th International Conference on Epigenetic Robotics.*

Olivier, Donald. 1968. *Stochastic Grammars and Language Acquisition Mechanisms.* Ph.D. thesis, Harvard Univerity.

Saffran, Jenny R, Richard Aslin, and Elissa Newport. 1996. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928.

Swingley, Daniel. 2005. Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*, 50(1):86–132.

Toft, Zoë. 2002. The phonetics and phonology of some syllabic consonants in Southern British English. In Toft, Zoë, editor, *Papers on Phonetics and Phonology: The Articulation, Acoustics and Perception of Consonants*, volume 28 of *ZAS Papers in Linguistics*, pages 111–144.

Venkataraman, Anand. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):352–372.

Villing, R, J Timoney, T Ward, and J Costello. 2004. Automatic blind syllable segmentation for continuous speech. *Proceedings of ISSC.*

Wolff, JG. 1977. The discovery of segments in natural language. *British Journal of Psychology*, 68:97–106.

Xie, Zhimin and Partha Niyogi. 2006. Robust acoustic-based syllable detection. In *Proceedings of the Ninth International Conference on Spoken Language Processing.*