2016-06-01

A Hybrid Approach to Cross-Linguistic Tokenization: Morphology with Statistics

Logan R. Kearsley
Brigham Young University

A Hybrid Approach to Cross-Linguistic Tokenization:

Morphology with Statistics

Logan R. Kearsley

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Arts

Deryle Lonsdale, Chair
Cynthia Hallen
Alan Melby
Michael Bush

Department of Linguistics and English Language

Brigham Young University

June 2016

ABSTRACT

A Hybrid Approach to Cross-Linguistic Tokenization:
Morphology with Statistics

Logan R. Kearsley
Department of Linguistics and English Language, BYU
Master of Arts

Tokenization, or word boundary detection, is a critical first step for most NLP applications. This is often given little attention in English and other languages which use explicit spaces between written words, but standard orthographies for many languages lack explicit markers. Tokenization systems for such languages are usually engineered on an individual basis, with little re-use. The human ability to decode any written language, however, suggests that a general algorithm exists.

This thesis presents simple morphologically-based and statistical methods for identifying word boundaries in multiple languages. Statistical methods tend to over-predict, while lexical and morphological methods fail when encountering unknown words. I demonstrate that a generic hybrid approach to tokenization using both morphological and statistical information generalizes well across multiple languages and improves performance over morphological or statistical methods alone, and show that it can be used for efficient tokenization of English, Korean, and Arabic.

Keywords: tokenization, lexing, morphological analysis

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Deryle Lonsdale, for putting up with me and helping me acquire the resources I needed to finish this research project, and for his guidance in polishing up this paper. I would also like to thank Dr. Michael Bush, my committee member and employer, for his encouragement and willingness to provide assistance even though finishing means I won't be a student employee anymore. It's his good luck that he's managing to retire just as I'm finally graduating. I am grateful also to Dr. Alan Melby for his mentorship on undergraduate projects which led up to this thesis, and to Dr. Cynthia Hallen for her good attitude, encouragement, and editing skills.

Of course, I also must thank my wonderful wife, Erin, who promised to love me whether I finished or not, and supported me throughout.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the most basic and yet often overlooked problems in natural language processing (NLP) is word boundary detection, or tokenization. Identifying words is a critical preprocessing step before almost any other work can be done with text: frequency counting, part-of-speech tagging, parsing, etc. In many languages, this is far from trivial. The standard orthographies for languages as diverse as Ancient Greek and Latin, and modern languages like Thai, Japanese, and Chinese, lack spaces or any other explicit indicator of word boundaries. (Examples of the standard orthographies for Thai and Japanese are shown in Figures 1.1 and 1.2.) This often results in ambiguity about the proper grouping of characters into words, and difficulty in identifying the boundaries of unknown words.

Substantial research has been done in text segmentation for languages like Chinese and Japanese. The highest-performing systems all use a hybrid approach, incorporating both statistical knowledge to predict probable word boundaries, and access to lexicons to identify known words. Unfortunately, the specific algorithms used are often drastically different, and bespoke tokenizers are typically built on a language-by-language basis. There is no standard, generic framework for handling tokenization across multiple languages. Even in English and other languages that use whitespace in writing, there are numerous edge-cases,

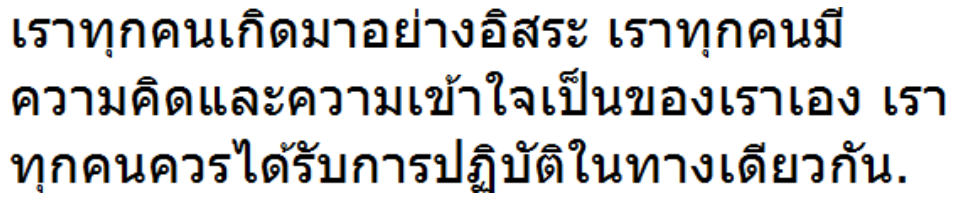

Figure 1.1: Example of Thai text. Thai uses spaces to separate clauses, but not individual words [1].

すべての人間は、生まれながらにして自由であり、かつ、尊厳と
権利とについて平等である。人間は、理性と良心とを授けられて
おり、互いに同 胞の精神をもって行動しなければならない。

Figure 1.2: Example of Japanese text. While punctuation and changes in writing system provide some clues, Japanese does not use spaces to separate words or clauses [2].

such as clitic and punctuation separation, which must be handled with special rules on a per-language basis. In developing software for searching, analyzing, or teaching multiple languages, or supporting natural-language based user interfaces, the complexity of developing separate systems to support equal levels of computer-enhanced interactivity for every desired language quickly becomes impractical. This severely limits both the number of languages that can be supported in any given application and the level of functionality available for each language. Given that the human ability to successfully read any natural language provides an existence proof that a generalized segmentation system (as implemented in the human mind) is possible, it is reasonable to investigate the feasibility of a language-agnostic segmentation system that could be easily integrated into larger natural language processing systems.

In order to make NLP applications more accessible in a wider variety of languages, I am developing a generalizable framework for tokenization which can be easily adapted to any given language. The general tokenization problem can be broken down into a few generic parts, regardless of the language involved: a hypothesis generator, which makes use of both a morphological recognizer encoding lexical knowledge and a statistical model or models, and a selector, which identifies the best hypothesis. Even if a single master system cannot be used for every language, decomposing the problem in this way should allow for replacing only certain parts and reusing others, as long as the replacement modules conform to a standard interface.

For the purposes of this thesis, I have focused only on the hypothesis generation stage. I identify promising lexical (morphological) and statistical models and several methods of integrating them into a single framework for hypothesis generation across multiple languages. I then test these systems on corpora of native-speaker texts from three typologically dissimilar

languages. The success of each system is determined by two major factors. The first is generalizability, or the consistency of results across multiple languages. Generalizability, however, is a necessary but not sufficient condition for a generic tokenization framework. The second factor is token-recognition performance. Additionally, hybrid systems can be evaluated in terms of their improvement over morphological or statistical methods used alone. I show that a hybrid system based on this framework:

1. does generalize well across a variety of languages,
2. produces results comparable to those of state-of-the-art tokenization systems developed for specific languages, and
3. outperforms both morphological and statistical methods used individually.

Chapter 2

Relevant Literature

Prior to almost any other natural language processing task, the first task that must be done with any text is tokenization (also called segmentation, lexing, or word breaking)—figuring out where the words (or other logical units) are.

This problem is most obvious in the context of automatic speech recognition, where it is further necessary to convert continuous phonetic data into discrete phonemes. The standard solution in this context is to use Hidden Markov Models (HMMs) encoding the possible words of the language to transduce phonetic data into sequences of potentially-overlapping possible phonemes and words [3]. The correct tokenization is obtained by applying the Viterbi algorithm over a pre-generated lattice[1] of "word hypotheses" [4], or by performing an on-line beam search over possible tokenizations as new possibilities are generated [5].

If we set aside the issue of transducing phonetic data into discrete phonemes, however, the problem of identifying logical groups in a stream of discrete lower-level symbols still remains, and we can isolate this aspect of the problem by working with written text,

[1] a data structure which stores only one copy of common prefixes and suffixes, showing which unique states (in this case, unique tokens) can precede or follow each other

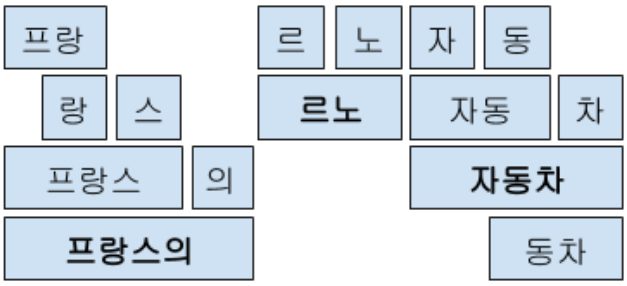

Figure 2.1: A sample lattice showing possible tokenizations of three words of Korean, excluding spaces. The correct selection is shown in bold.

assembling morphemes and words from a stream of graphemes. In general, tokenization of written text[2] may be a harder problem than tokenization of speech since writing generally does not reflect spoken language losslessly[3], and fewer boundary clues (such as prosodic information) will be available. Nevertheless, the fact that reading is possible is itself proof that text-only tokenization can be done at useful levels of accuracy.

2.1 Tokenization of English

In English and many other languages, the problem of text tokenization is often considered trivial: simply split words on whitespace, which substitutes for many of the prosodic clues available in speech. Van Aken [6], for example, studied the problem of tokenizing continuous English text with spaces removed, but only as a proxy for understanding human tokenization strategies more generally. While often good enough, however, the space-splitting approach is far from perfectly accurate. Even with perfectly clean data, this approach fails to account for:

1. Word boundaries that correctly occur in the absence of whitespace (e.g., before punctuation, with specialized items like URLs, or simply due to typographical errors)

2. Whitespace that may not indicate a relevant boundary in certain applications (e.g., spaces in compounds, idioms, and borrowed foreign-language phrases used as single logical lexical units).

These complications can be dealt with in various ways, but only on an individual language-by-language basis. Even among languages that all use the Roman alphabet, significant differences in whitespace and punctuation conventions exist. The tokenization conventions for the Penn Treebank, for example, separate "most punctuation" [7], but depend on prior knowledge of sentence boundaries to help disambiguate final periods from periods indicating abbreviations and on English-specific knowledge about the structure of contractions and cliticized forms.

[2]In corpus studies, "text" is often used to refer to any reification of language, be it spoken or written; here however, I will assume that "text" refers only to written communication.

[3]i.e., without loss of information, such that all details of speech could be re-constituted exactly from its written form

Van Aken's system relies purely on statistical information available within a single text to infer word segmentation. In particular, it keeps track of recurring sequences that occur with frequency greater than a certain threshold as possible word types. It further keeps track of the transition probabilities between known sequences to determine the optimal path through a lattice of tokenization hypotheses [6]. This algorithm was tested against both randomly generated strings of words from a dictionary, and a small sample of natural English text, which revealed a weakness in identifying boundaries in sequences of multiple short words, such as "in a"; this is a particular weakness shared by other statistical algorithms as well. This approach taken in isolation would likely perform poorly on a text containing a large number of hapax legomena[4], which would pass the frequency threshold for recognition, making it potentially less suitable for languages with extensive productive morphology.

Van Aken's algorithm also depends on an implicit assumption about what the definition of a "word" is: a fixed sequence of symbols that can occur multiple times. While this assumption may indeed be a useful one for vocabulary induction from unlabeled data, it is not necessarily the best definition of "word" for all circumstances, especially across multiple languages. Additionally, there is ambiguity in the usage of "word" to refer to a particular sequence of symbols, versus a particular instance of that sequence in a text. Unfortunately, per Islam et al. [8], no widely accepted definitions for what constitutes a "word" exist, even within a single language, let alone cross-linguistically. Different native speakers often segment text in different ways, and the rate of agreement between human judges can be less than 80%, making it impossible to construct a single "gold standard" to evaluate results between systems that employ different conventions, whether implicit or explicit. For these reasons, it is often useful to dispense with the problem of defining a "word" at all and, when exactness is required, to instead refer to *types*, meaning specific abstract sequences of symbols, and *tokens*, meaning specific instances of a type at specific locations in a text. Tokenization is thus the process of identifying meaningful tokens, however "meaningful" is defined for a particular application in a particular language, and may cover items traditionally identified as "words" as well as clitics, punctuation, and other errata. Note, however, that differences in conventions regarding what constitutes a meaningful token at a given level of analysis

[4]types which occur exactly once in a corpus

still require consideration during evaluation. The performance of tokenization algorithms is usually evaluated in terms of *precision* and *recall*; precision (P) is defined as the proportion of items in the output that are correct (i.e., which match items in the answer key), while recall (R) is defined as the proportion of correct items (i.e., entries in the answer key) which are in the output. Mathematically, these are given by

$$P = \frac{TP}{TP+FP}$$
$$R = \frac{TP}{TP+FN}$$

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. Since it is often easy to obtain high performance on either measure individually (e.g., perfect recall can be achieved simply by guessing that every character boundary might be a token boundary, such that there are many false positives but no false negatives), but difficult to achieve high performance on both, an additional measure, the *F-score* is used to combine precision and recall and provide an estimate of overall performance. In this thesis I use the usual definition:

$$F = \frac{2PR}{(P+R)}$$

For all three measures, scores range from 0.0 to 1.0, where a score of 1.0 represents perfect performance.

While his algorithm requires no prior knowledge of linguistic structure, van Aken [6] does note that "A shortcut [...] is to incorporate a ready-made dictionary that contains some or all of the words that the algorithm will encounter." This lexical-access approach is most clearly demonstrated by Norvig [9], who describes the use of a simple lexicon to recursively split a string into a known word and a suffix of remaining characters, using dynamic programming techniques to construct a lattice to avoid the inefficiency of recomputing overlapping segmentations for the same suffixes multiple times. Word-level n-gram frequency data is then used to extract the highest probability tokenization from the lattice. Norvig [9] cites applications to Chinese, as well as to specialized genres of English, such as URLs, which are written without spaces as previously described. Norvig's algorithm, however, is only as good as the dictionary—it will fail on encountering out-of-vocabulary (OOV) words. This makes it less suitable for application to languages that have extensive productive morphology, such

as Turkish, in which it may be literally impossible to encode the complete set of possible words in a finite dictionary, even before we account for the fact that new words appear constantly in living languages [8]. Although this overlaps somewhat with the weaknesses of van Aken's algorithm, the potential benefits of a hybrid approach are already evident, as a strong statistical algorithm may make up for gaps in vocabulary.

2.2 Orthographies Traditionally Lacking Whitespace

Most practical work in text tokenization involves languages whose writing systems traditionally have no spaces or other written word breaks, such as Chinese, Japanese, or Thai, as previously mentioned. In these cases, as with the special rules for Treebank annotation, researchers develop separate tokenization systems on a per-language basis, often using substantially different approaches. This has produced some impressive state-of-the-art results in each language, but unfortunately these systems all differ significantly in their details, and none are easily extensible for use on additional languages without effectively re-building them from the ground up.

Peng, Feng, & McCallum [10] developed a system for Chinese segmentation which recasts segmentation as a tagging problem using linear-chain conditional random fields (CRFs); every character is tagged as either a start-of-word character, or not. This is an inherently hybrid system, as the CRF models incorporate lexical knowledge in addition to lower-level statistics, which are used for probabilistic new-word detection. This provides the best of both worlds: the efficiency and accuracy improvements that lexical access entails, without the need to store the entire vocabulary, and allowing for the proper identification of unknown words as long as they do not exhibit structural characteristics too far removed from those used to create the statistical model (i.e., the normal morphographological[5] structure of the language). Their model produced results ranging from ($P = 0.828$, $R = 0.870$, $F = 0.849$) to ($P = 0.953$, $R = 0.961$, $F = 0.957$) over eight tests on various corpora.

The state-of-the-art Japanese segmentation system by Kudo, Yamamoto, & Matsumoto [11] also makes use of CRFs to simultaneously achieve tokenization and morphological analysis by selecting the best path through a lattice of possible *morpheme*-level to-

[5] the written parallel to morphophonological

kenizations generated from a morphologically-aware dictionary. In fact, Kudo et al. were primarily motivated by the problem of morphological analysis of unlabelled text, for which automatic tokenization was merely an unavoidable first step when working with Japanese orthography. This builds on prior work by Asahara et al. [12] and Uchimoto et al. [13] on Japanese tokenization and morphanalysis using two different statistical modelling techniques: HMMs and Maximum Entropy Markov Models (MEMMs), respectively. Kudo et al.'s CRF model improved on both HMM and MEMM models due to the ability of CRFs to make use of a wider range of tagging features, including lexical information, providing additional evidence for the utility of hybrid segmentation approaches. They report results of ($P = 0.9904$, $R = 0.9888$, $F = 0.9896$) and ($P = 0.9903$, $R = 0.9920$, $F = 0.9911$) on two different corpora.

Choosing to work at the level of morphemes rather than words additionally removes many of the problems present in other systems with hapax legomena or OOV words. This requires, however, either the ability to re-synthesize complete words from morphemes, or an agreement with later stages of the NLP pipeline to define "tokens" at the appropriate sub-word level. Where possible, however, the ability to achieve simultaneous tokenization and morphanalysis substantially improves efficiency and provides more contextual information to later stages of an NLP pipeline, such as syntactic parsing.

Suzuki, Brockett, and Kacmarcik [14], in fact, took this one step further, by using a syntactic model to simultaneously select an optimal tokenization while producing a syntactic parse. Like Kudo et al. [11], Asahara et al. [12] and Uchimoto et al. [13], they also use a "word breaker" that performs simultaneous morphanalysis while producing a lattice of possible tokenizations. However, they eliminate the complication of collapsing the information in the lattice into a single unambiguous best token sequence. Instead, the lattice is used as the bottom level of a parsing chart, with the selection of the correct token sequence being a natural consequence of identifying the best syntax tree. They report results ranging from ($P = 0.974$, $R = 0.980$, $F = 0.977$) to ($P = 0.981$, $R = 0.985$, $F = 0.982$) on three corpora.

In my own previous work [15], I addressed the problem of efficient lattice generation with simultaneous morphanalysis using a finite-state transducer (FST) based on the KIMMO two-level morphology system [16]. This system runs in amortized constant time

per-character. Hence, as each input character needs to be examined exactly once and in document order to determine which morphemes it could participate in, it generates a lattice of all possible sequences of known morphemes that could be extracted from an input stream in amortized linear time over arbitrary sized inputs [15]. Storing the lexicon as a finite-state transducer both saves a great deal of space and also allows recognition of a potentially infinite number of regularly derived or inflected words, though with some reduction in flexibility versus the CRF models. Compared to a fixed lexicon as used by Norvig [9], a morphological transducer suffers less from from the problem of encountering OOV tokens, but does not solve the problem completely; it will still encounter difficulty with unknown roots and other morphemes. This system was tested on English and Turkish, using synthetic corpora consisting of randomly-generated sequences of concatenated words, similar to the method used by van Aken [6]. On corpora of known words, the system achieved ($P = 0.209435$, $R = 1$, $F = 0.346335$) for English and ($P = 0.164989$, $R = 1$, $F = 0.283246$) on Turkish. The low precision scores are attributable to the fact that I focused only on lattice generation, as a clearly separate issue from path selection—an approach I continue in this thesis. Prior to optimal path selection, the precision measure of a lattice generation system reflects the level of genuine morphological ambiguity present in the input.

The great advantage of this approach is that nearly any morphological model can be adapted to use in a simultaneous lexical-access based tokenizer (although it is optimized for FST-based models). In other words, the language model does not need to be developed with tokenization specifically in mind, and existing analyzers are reusable, potentially saving a great deal of effort that would otherwise go into another bespoke, language-specific tokenization system. Additionally, this approach specifically supports on-line, real-time usage. While that feature is common (nearly obligatory, in fact) for speech processing, it is relatively rare in text tokenization [6]. On-line operation significantly restricts the types of algorithms that can employed in tokenization, because it is necessary for the speed of processing to keep up with the speed of input. This means that any system intended for on-line use must take, on average, constant time to process each character in the text, resulting in amortized linear ($O(n)$) time to process a complete text. Real-time applications introduce slightly stricter constraints on the maximum allowable constant factor per character.

2.3 Psychological Modeling

In addition to the NLP-oriented approach, the tokenization problem also arises in language acquisition: When hearing a new language that one does not know, how does a learner begin to identify new words to add to their mental lexicon? This clearly requires some generic mechanism (although of unknown and possibly great computational complexity) for inducing probable words boundaries (where contiguous sequences between two boundaries then constitute words) from unlabeled input. That humans are capable of this provides an existence proof of the technical possibility of language-agnostic tokenization.

Studying the ways that human minds may tackle tokenization is not guaranteed to produce the best possible artificial tokenization system, since they may involve unnecessary complexity or require quantities of data infeasible to obtain for many NLP applications, in the same way that the best way to build an airplane is not to exactly copy a flapping bird's wing. Studies of human psychological segmentation mechanisms do, however, serve two useful purposes: They set a baseline against which errors can be measured, and they may provide an initial example implementation from which further engineering can draw [17].

To establish a baseline of performance, it is sufficient to use manually-tagged corpora as a gold standard to test against; to provide an example implementation for re-engineering, it is important to have a reasonable psychological model of human speech segmentation. Daland [17] argues that both statistical predictive mechanisms and lexical access are necessarily involved in human segmentation, with contributions from each shifting over the language acquisition process: statistical mechanisms in order to guess at the boundaries around new unknown words and to determine the probability that a potential bounded sequence is in fact a new word, and lexical access to filter errors. Empirical evidence for the validity of this claim is provided by Islam et al. [8], who report that many state-of-the-art NLP systems do in fact use hybrid approaches. Daland [17] further proposes that there are only two feasible error-correction conditions for the human tokenization system. The first is statistical over-recognition, where lexical access must filter out erroneously predicted word boundaries; the second is statistical under-recognition, where lexical access must be prepared to add statistically unlikely word boundaries. Logically, however, both may occur in any newly-developed system.

In either case, we can expect an improvement from merging statistical and lexicon-based methods. The set of correct word boundaries found by either method would not *a priori* be expected to form a subset of those found by the other; thus, their unions would be a larger set of correct boundaries. However, in the case of statistical under-recognition, we would expect a smaller contribution from statistical boundary prediction to the recognition of unknown words by lexicon-based methods. We would thus expect that statistical over-recognition would be the norm, and indeed, this is exactly the result obtained by Rytting [18].

2.4 Simplified Statistical Models

Revisiting the analogy of the airplane versus flapping bird wing, one is led to wonder whether statistical models might exist which are much simpler than CRFs or HMMs yet still in some sense "good enough". Brent [19], motivated, like Daland, by the study of first language acquisition, identified distribution regularities at the phoneme (or grapheme) level, such as segment-to-segment transition probabilities, to be useful indicators of word boundaries [20]. In later work, Brent [19] and Rytting [18] both investigated simple character-level statistical cues that could be used as effective predictors of word boundaries during language acquisition, and for tokenization of text. The identification of such useful "minimal information" predictors holds significant promise for developing simplifications of Daland's cognitive model [17] and further improving the performance of hybrid tokenizers.

While Brent [19] assumed that a prediction technique based on local comparisons (i.e., looking for significant differences between the values of a given statistical measurement between adjacent positions in the text), would be *most* appropriate for identifying word boundaries, Rytting [18] argued that this is guaranteed to under-predict due to the impossibility of correctly identifying common single-segment words, since the values of any two measurements in immediately adjacent positions cannot simultaneously be either significantly greater than or significantly less than each other. This echoes the problems with identification of small words encountered by van Aken [6] with a significantly more complex model. Global comparisons are therefore needed to ensure maximum recall of correct boundaries.

Rytting [18] tested the predictive power of two statistics based on Brent's work [19]: character-level transition probabilities, and character bigram mutual information scores, using both local and global comparisons for each. Rytting [18] showed that, on Greek text, a very simple mutual-information (MI) model outperformed the competing transition-probability models. He also showed that it was highly over-predictive on its own, and that using a global threshold for comparison was the most effective of the techniques that he tested, giving results of ($P = 0.439$, $R = 0.544$, $F = 0.486$) for identifying individual token boundaries. This is despite the fact that the threshold value of zero used in Rytting's study [18] was chosen "arbitrarily" with no justification attempted. Rytting's model, however, does also take into account the probability, based on a pre-annotated training corpus, that any particular bigram spans a known word boundary.

In my own previous work [21], I applied the MI approach to English, testing the sensitivity of the model to the quantity of training data available for calculating bigram MI scores. In contrast to Rytting's supervised (i.e., guided by human input) approach, however, I employed a further simplified, completely unsupervised algorithm which relied solely on MI scores calculated from the unannotated text of the Open American National Corpus (OANC) [22], consisting of samples from a variety of genres including transcribed speech, fiction, news, and technical writing. Additionally, in response to Rytting's claim that the zero threshold is "arbitrary", I compared its performance with an alternative "gap threshold" assignment method. In this approach, a model-specific threshold is dynamically selected by dividing bigrams into two clusters based on the largest gap between successive MI scores.

The unsupervised zero-threshold MI approach had better precision on English than Rytting's supervised approach [18] did on Greek, although it comes at a cost in recall—($P = 0.564$, $R = 0.386$, $F = 0.458$). The gap-threshold approach, on the other hand, showed high recall, but significantly lower precision—($P = 0.275$, $R = 0.846$, $F = 0.415$). I also found that, above a certain relatively small minimum size, results were not strongly dependent on the size of the training corpus. This, combined with the demonstrated efficacy of the unsupervised approach, means that the data required to build useful statistical word boundary prediction systems can be acquired cheaply, without the need for human annotation. Neither my nor Rytting's results, however, show recall scores close to 1.0, which, together with the

early performance plateau, supports Daland's thesis that human tokenization must transition from primarily statistics-based to more lexical-access based methods during the learning process. While these extremely simple systems based on a single statistical measure (in this case, mutual information) do not have sufficiently good performance to use entirely on their own, their improvement over a random baseline is sufficient to justify investigating their use as heuristic filters to improve the performance of more complex hybrid word segmentation systems [21].

2.5 Path Selection Methods

While my work sidesteps the problem of path selection, it is worth briefly reviewing what options are available. Suzuki et al.'s [14] parser-based approach, while highly effective, is also very heavyweight, as it requires the use of a syntactic model and full parser. Several simpler options exist for modular systems that separate lattice generation from path selection. Kudo et al. [11], as justification for the need to incorporate lexical knowledge, cite accuracy of over 0.9 using a simple longest-prefix matching algorithm, which simply assumes that the longest prefix of a string which is a known token is a correct token, and then restarts after that longest prefix to find the next token. It is unclear, however, whether the cited statistic applies only to Japanese, or cross-linguistically. Islam et al. [8] also describe a "maximal match" algorithm (with several variations for resolving ambiguities), which essentially consists of finding the longest substring that is a known token (regardless of whether or not it is a prefix of the text), and then proceeding to select the next longest known tokens in the remaining non-overlapping sections of text. Their particular modification, using type frequency and entropy measurements to resolve ambiguity between equal-length options, achieved ($P = 0.8992$, $R = 0.9469$, $F = 0.9224$) in tests on the Brown corpus. This method is particularly useful in restricted settings where it is necessary to identify correct instances of certain types in desegmented text, but where complete tokenization and identification of unknown tokens is unnecessary; for example, a "maximal match" algorithm is used in the TIARA[6] [23] and Ayamel [24] software developed by BYU's ARCLITE Lab in order to identify words and phrases that require annotations to be applied—an application in which

[6]The Interactive Annotated Reader Application

normal English tokenization conventions are completely unsuitable, as the relevant tokens are often larger than single words and may contain spaces. Norvig [9] provides an approach which incorporates language-specific knowledge but is still relatively simple, calculating the most likely lattice traversal using n-gram probabilities, as noted above. Accurate results from this approach, however, depend on access to an extremely large training corpus (over 3 billion words in this case) to ensure that accurate statistics are available for all possible n-grams; the algorithm's performance is compromised when presented not merely with OOV tokens, but with previously unseen sequences of known tokens. The OOV token problem is partially addressed by model smoothing—assigning a small default probability to any unknown token—but can be significantly ameliorated by replacing the simple word n-gram model with a part-of-speech (POS) tag model, which can produce accurate statistics with a much smaller training corpus, since one tag sequence equates to several concrete word sequences. The tag-based approach is particularly attractive for use with systems that employ simultaneous morphanalysis, since POS information comes essentially "for free".

2.6 Lattice Generation with Morphology & Statistics

Regardless of the path selection algorithm used, the final results are only as good as their input. If a lattice generation step cannot produce all of the correct token boundaries as hypotheses, then no selection algorithm can produce the complete correct token sequence. On the other hand, a naïve conservative lattice generation algorithm could simply propose *every* character boundary as a possible token boundary, in which case the correct sequence is guaranteed to exist somewhere, and could be found—but only in a highly inefficient manner, as the lattice generator has provided no useful information to guide the search. A separate lattice generation algorithm can thus be rated both on how many correct hypotheses its output contains, and on how much work it requires of the succeeding path selection algorithm. In this context, lexicon-based tokenizers have an additional advantage over many pure statistical algorithms in that they inherently produce pairs of starting and ending boundaries for specific tokens; this significantly reduces redundancy compared to single-boundary tagging systems [11] like those of Peng et al. [10], Brent [19], or Rytting [18]. In the worst case, these systems can require considering a number of token hypotheses

proportional to the square of the number of individual boundary hypotheses, significantly slowing down the tokenization process.

Based on my previous work, I have built a series of hybrid lattice-generation systems implementing fundamentally language-agnostic lexical-access based and simple statistical algorithms, employing pre-existing morphological analyzers to provide language-specific lexical knowledge. I will show that a relatively simple hybrid model produces results which improve on the performance of either lexical-access based or simple statistical methods alone. I will also demonstrate the generalizability of these algorithms across multiple languages that are orthographically and typologically (specifically, morphologically) dissimilar.

Chapter 3

Methodology

3.1 Data Collection & Formatting

I ran tokenization experiments on each of three languages—Arabic, English, and Korean, covering the range of synthetic, isolating, and agglutinative typologies. This first required acquiring and pre-processing appropriate corpus data to use for testing, including annotations indicating the correct token boundaries to use as a gold standard.

Corpus data for Arabic came from the Arabic Treebank: Part 1 v 3.0 [27], consisting of 165,419 tokens of Modern Standard Arabic, where tokens include full words, clitics, and punctuation. Corpus data for English came from a subset of the Open American National Corpus (OANC) [22], consisting of 100,165 tokens, again including words, clitics, and punctuation. Corpus data for Korean came from data packaged with the KLEX morphological analyzer [28], itself a subset of the Korean Treebank, consisting of 41,024 tokens, including independent words, punctuation, and dependent suffixes. In each case, I converted all data into plain text in UTF-8 encoding.

Answer keys and tokenizer outputs were stored as lists of pairs of starting and ending indices for each token. I measured token boundary indices in terms of Unicode codepoints in order to abstract away the problem of defining a single "character" in the presence of

يولد جميع الناس أحراراً متساوين في الكرامة والحقوق. وقد وهبوا
عقلاً وضميراً وعليهم ان يعامل بعضهم بعضاً بروح الإخاء.

Figure 3.1: Example of Arabic text [25]. While Arabic does use spaces, not *all* tokens are separated by spaces. Additionally, the lack of written vowels introduces an unusually high level of ambiguity into the recognition of tokens and token boundaries.

모든 인간은 태어날 때부터 자유로우며 그 존엄과 권리에 있어
동등하다. 인간은 천부적으로 이성과 양심을 부여받았으며 서로
형제애의 정신으로 행동하여야 한다.

Figure 3.2: Example of Korean text [26]. Like Arabic, Korean does use spaces, but with different conventions for spacing, punctuation, and clitic attachment than English uses.

Arabic vowel diacritics and Hangul syllabary combining characters. English token boundary locations as measured in codepoints were already available in the OANC annotations, and only had to be extracted and put into the list format. I calculated answer keys for Arabic and Korean by matching up sequential tokens in part-of-speech tagging files with their locations in the unannotated plain text.

Acquiring suitable annotated corpora and morphological analyzers for all three languages was the most time-intensive portion of the project. Once those were in hand, it was a matter of a day or two per language to write and run Python scripts which could extract answer key data from each of the differing annotation formats used by the three corpora, convert the data into UTF-8 plain text, and write adapters for each morphological analyzer's interface. Running experiments took approximately two weeks, including debugging time.

3.2 Morphological Tokenization

The simplest dictionary-based tokenization algorithm calculates all possible substrings of the input. Each substring is then checked against the dictionary and, if found, is added to the lattice of possible tokens. In order to reduce the incidence of unknown tokens and reduce the size of the lexicon that needs to be stored, the simple dictionary can be replaced by a morphological analyzer to determine whether a given string is or is not a token. This is particularly important in languages, such as Turkish, with extensive productive morphology, in which most tokens will not appear in a standard dictionary.

This naïve approach to lattice generation is, however, highly inefficient. Given that there are $n-m+1$ substrings of length m in any string of length n, we must check for tokens of all possible lengths (of which there are n). The algorithm thus requires $\sum_{m=1}^{n}(n-m+1) = \frac{1}{2}n(n+1)$ or $\mathrm{O}(n^2)$ time to produce all possible substrings, times a language-specific function

$w(m)$ representing the time required to accept or reject a string of length m to produce the lattice of possible tokens. If $w(m)$ is approximately linear (i.e., the amount of time it takes to accept or reject a possible token is, on average, proportional to the length of the token), this results in $\sum_{m=1}^{n} m(n-m+1) = \frac{1}{6}n(n+1)(n+2)$ or $\mathrm{O}(n^3)$ time required to produce all possible tokenizations of a given input—entirely unacceptable for on-line, real-time use. In practice, however, while specific tokens may be unbounded in length, the highly non-random morphological structure of human languages means that most non-tokens will be rejected early since the probability of any randomly selected substring being a valid prefix of the language is inversely related to length. As a result, for long inputs, $w(m)$ is well approximated by a language-specific constant factor k_w representing the average time (in terms of codepoints read) required to accept or reject a string as a valid token, resulting in $\mathrm{O}(n^2)$ asymptotic complexity. This is still too slow for on-line applications, but some additional savings are possible if we know that there is an upper limit on the largest possible token in the dictionary; that reduces m to an effective constant factor as well, rather than ranging to the arbitrarily large value of n (the total size of the input). A similar, adaptive, version of that optimization was used by Norvig in his lexicon-based algorithm [9].

There is one fundamental problem with the naïve algorithm which causes repeated work: the rejection of a particular substring as a token says nothing about the status of any other substrings containing the first as a prefix. This results in the need to check each substring, which means that each codepoint is examined $n/2$ times—once for each substring of which it is a part—thus generating the aforementioned $O(n^2)$ time complexity under the assumption that $w(m)$ has some constant upper bound. However, if it is possible to accept or reject substrings as possible *prefixes*[1] of valid tokens, then, after a substring is rejected as a possible prefix of any valid token, there is no need to check any larger substrings containing the first as a prefix. This reduces the number of operations that must be performed on each codepoint to be proportional to the level of morphological ambiguity at any point in the input, which, under the same assumption that let us treat $w(m)$ as an effective constant, approaches a constant average value on large inputs, rather than growing linearly with the

[1] where "prefix" in this case refers to any string of codepoints matching the beginning of a token, not to a morphological prefix

size of the input. This gives us access to amortized linear-time performance without the need to artificially limit the maximum length of substrings that we will consider for tokenhood.

Additional redundancy can be eliminated by remembering that a particular substring has already been accepted as a possible token prefix when considering the text that follows it. Given an analyzer that can only check complete strings all at once (which is the interface provided by most morphological analyzers), each codepoint must be examined at least once for every possible token (or not-yet-rejected prefix of a possible token) that overlaps that codepoint's position. If, however, the internal state of an analyzer that has already accepted a given prefix can be saved and incrementally updated with following codepoints, rather than starting over from the beginning of that prefix to check the validity of every new substring starting at the same position, the number of operations that must be performed per codepoint is reduced to become proportional to the number of *groups* of valid token prefixes having different starting points that overlap that position. This does not provide any additional asymptotic improvements, but yields significant constant-factor savings.

Both of these optimizations can be naturally realized by encoding the morphological analyzer as a finite state machine (FSM), which can consume input one codepoint at a time, examining each codepoint exactly once. Transitioning to an accepting state of the automaton triggers the output of the current accepted token, but does not cause the automaton to terminate. Instead, it continues consuming more codepoints and emitting tokens at every accepting state until entering a failure state, indicating that the string consumed up to that point is no longer a valid prefix of any token, and thus no more tokens beginning in the same position are recognizable. Since the automaton consumes no more input after reaching a failing state, substrings containing rejected prefixes are never considered. Additionally, common prefixes are identified only once; the complete set of a maximal token and all possible prefix tokens (e.g., base forms missing suffixes, or components of compounds) are recognized in the same time as a single maximal token. Encoding a morphological analyzer as a pure FSM is not always convenient, but the same optimizations are available to any model that relies only on features that can be updated one codepoint at a time, and can thus be analyzed as a (potentially infinite) state machine with each codepoint consumed resulting in a transition to a discrete, recallable state. Even without the optimization that

eliminates duplicate work on all prefixes of a maximal token, however, we can at least achieve the necessary linear-time performance with any model that can reject invalid prefixes.

In order to recognize possible tokens occurring as suffixes of maximal tokens (base forms missing prefixes, components of compounds, etc.) and to recognize tokens beginning at later points in the input stream, it is necessary to initialize a new instance of a morphological model, whether FSM-based or otherwise, (henceforth a "recognizer") in its starting state at each point in the input stream. Every time a codepoint is consumed, it is thus fed into each of a cohort of recognizers that have not yet reached failure states. When processing first starts, the number of active FSMs grows linearly with the number of input codepoints consumed, as one is created for each codepoint. Recognizers will, however, be eliminated at a rate proportional to k_w (the aforementioned average number of codepoints required to reject a string as a possible token) times the size of the cohort. This algorithm is illustrated in 3.3. Eventually, the collection of active recognizers reaches an approximate steady state, with a uniform average cohort size over long stretches of input. The expected number of recognizers active at any given point in the input stream, and thus the number of operations that must be performed per codepoint, is thus proportional to k_w. Hence, we achieve $O(k_w n) = O(n)$ (linear) performance, allowing the production of all possible tokenizations of any input on-line and (if the constant factors are small enough) in real time. Additionally, as each recognizer in a cohort can be advanced independently, the algorithm is trivially parallelizable.

For the first set of experiments, I implemented a generic morphological tokenizer in Python which reads input, builds a lattice, and manages cohorts of recognizers, and and which allows any concrete morphological analyzer to be plugged in to the system. Where necessary, I then wrote simple wrappers around the morphological analyzers for each language to ensure that they would present the proper FSM-like interface. The output of each morphological experiment was saved in the form of a list of pairs of codepoint indices indicating the beginnings and ends of identified tokens—the same format used for the answer keys. This allowed the output stream to be replayed for use in hybrid experiments without needing to actually re-run the morphological tokenizer again, saving significant amounts of time and computational resources.

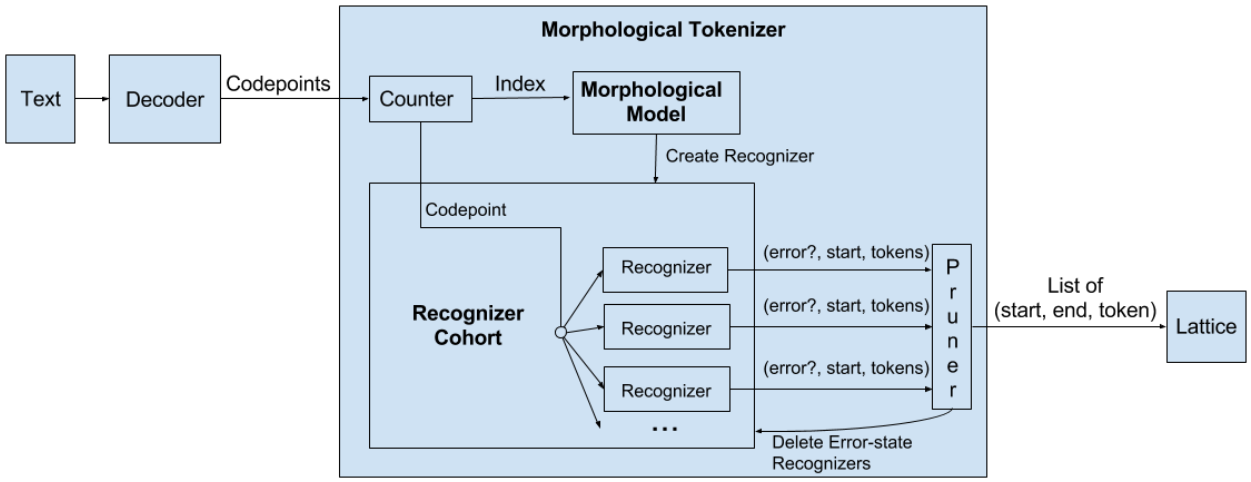

Figure 3.3: Morphological Tokenization

3.2.1 Arabic

I performed morphological tokenization of Arabic using the MADAMIRA morphological analyzer [29]. MADAMIRA, unfortunately, does not expose an FSM network API, so it was necessary to collect complete candidate-token substrings to check all at once. Additionally, it does not provide any means of querying valid prefixes. In order to avoid quadratic or cubic runtime, which would make the experiment infeasible even on relatively short texts, extra rules were added to the interface connecting MADAMIRA to the generic tokenizer to reject strings satisfying certain conditions which I knew a priori could never result in valid prefixes. First, the interface would reject any string longer than 16 codepoints, as there were no tokens in the Arabic corpus greater than that length. Second, the interface would reject any string containing a whitespace codepoint, since I knew ahead of time that the MADAMIRA lexicon did not contain any tokens containing whitespace codepoints. In other respects, however, MADAMIRA is almost *too* sophisticated to effectively integrate with a tokenizer. In particular, MADAMIRA does not assume that the input will necessarily be a single word, and can analyze sentences with words in context; as a result, it does its own word breaking (by some apparently undocumented method), and can return positive results for input containing more than one token. In order to account for this, I filtered MADAMIRA's output to accept only those strings which MADAMIRA identified as containing exactly one "word". Additionally, MADAMIRA will "helpfully" ignore extraneous punctuation, leading

it to produce valid analyses for words with, e.g., parentheses attached, as well as for the actual words themselves. I could find no principled way to avoid these spurious analyses.

MADAMIRA is accessed via an HTTP interface, which introduces significant input/output overhead. In order to maximize CPU usage, I split the corpus into three parts processed by three simultaneous instances of the tokenizer, each accessing a single MADAMIRA server process.

3.2.2 English

I performed morphological tokenization of English with the Englex morphological description of English [30] developed for PC-KIMMO [16]. This model was run on a modified version of PyKIMMO[2] known as Stream-KIMMO [15], which already has the appropriate FSM-like interface and required no adaptation. While the system runs in linear time, however, the original PyKIMMO implementation was highly inefficient, and introduced an enormous constant-factor slowdown, proportional to the size of the lexicon, due to simulating an FSM by dynamically calculating transitions and new states by iterating over the entire plain-text lexicon on every codepoint. Re-implementing the core of PyKIMMO to construct a real FSM was not a viable option, but I was able to introduce several efficiency improvements which sped up the algorithm by approximately a factor of five; still, the relative slowness of this analyzer limited the quantity of text that could be processed from the OANC within a reasonable timeframe. The English morphological experiment thus terminated after approximately four days, once sufficient data was obtained. The 100,165 tokens processed constitute a random sample of approximately 0.7% of the 15-million-word OANC covering fiction, magazine text, and voice transcription.

3.2.3 Korean

I performed morphological tokenization of Korean using the KLEX morphological analyzer developed for the Xerox Finite-State Tools (XFST). KLEX was originally designed to operate with input in either KSC-5601 or Unicode 1.0 encoding. In order to comply with modern versions of XFST, which only accept UTF-8 or Latin-1 encodings, and to enable

[2] a Python implementation of the KIMMO two-level morphology algorithm

it to work with the corpus which had been converted into UTF-8, I modified the KLEX source code to handle modern Unicode input. Fortunately, KLEX had been designed to transliterate Hangul into a modified Yale romanization for internal processing and back again; thus, updating it for modern encodings was a simple matter of running a search-and-replace on the file containing Hangul-Yale equivalencies to replace the old Korean codepoints with modern Unicode 8.0 UTF-8 codepoints.

Python bindings for the XFST library do exist which are supposed to directly expose the underlying FSM, allowing linear-time traversal. However, these tools have not been maintained, and I was unable to successfully install the software. Instead, I made use of a simpler Python API which exposes only the "apply up" and "apply down" functions of the morphological analyzer. As with MADAMIRA, I thus introduced special-case rules which rejected any string longer than 22 codepoints (again based on the maximum-length token known to exist in the corpus), and to reject any string containing whitespace, again based on prior knowledge that the KLEX lexicon contained no tokens containing whitespace.

KLEX has the ability to recognize certain suffixes and clitics in isolation, if they are prefixed with the tag "^DEP+" to indicate their status as bound (dependent) morphemes. Thus, in order to account for at least some possible variation in tokenization conventions, I ran two experiments with the morphological tokenizer on Korean: one which ignored dependents, and one which checked every possible token alone and with "^DEP+" prefixed.

3.3 Statistical Tokenization

In order to create a predictive model for statistical tokenization, I computed mutual information (MI) scores for every pair of codepoints present in each corpus. For this purpose, MI is given by the formula $\log_2(\frac{P(a,b)}{P(a)P(b)})$, where a and b are adjacent codepoints, $P(x)$ is the probability of encountering a given codepoint x at any position, and $P(a, b)$ is the joint probability of encountering the pair of codepoints a followed by b at any given position. Word boundaries were then predicted by testing the MI score of each codepoint pair in the corpora against a threshold value, as shown in 3.4. This trivially requires constant time per pair, and thus achieves the necessary $O(n)$ complexity for on-line usage, like the morphological algorithm.

Two different methods of calculating thresholds were used in two different experiments:

1. In the gap-threshold method, the largest gap between MI scores for a given languages was identified, and a model was created which predicts a token boundary between every pair of codepoints whose mutual information score falls below that gap. This encodes the intuition that there may be a significant clustering of codepoint pairs that can occur across token boundaries versus codepoint pairs that tend to occur within tokens, and was previously found to produce extremely high recall with reasonable accuracy on English data [21].

2. In the zero-threshold method, boundaries are predicted between every pair of codepoints whose MI score is less than or equal to 0, for all languages. This encodes the intuition that there may be a token boundary wherever the probability of encountering a codepoint in that position is less than the overall probability of encountering that codepoint in any position.

Contrary to Rytting's method [18], MI scores are used to predict token boundaries directly, rather than being weighted against a known prior probability of a particular codepoint pair marking a token boundary as determined from training data. This simplification results in a completely unsupervised system, which makes it attractive for use in a hybrid system as it avoids increasing the amount of language-specific configuration needed beyond the dictionary or morphological model. With either threshold-assignment method the model could also be updated on-line, as in Brent's prior work [19], during real-time processing of a large input stream.

As in the morphological experiments, the output of each statistical experiment was saved to allow replaying the output for hybrid experiments. Since the statistical methods only identify single boundaries between tokens, rather than pairs bounding the start and end of a token, the saved output in this case consists of only a single list of possible boundary indices.

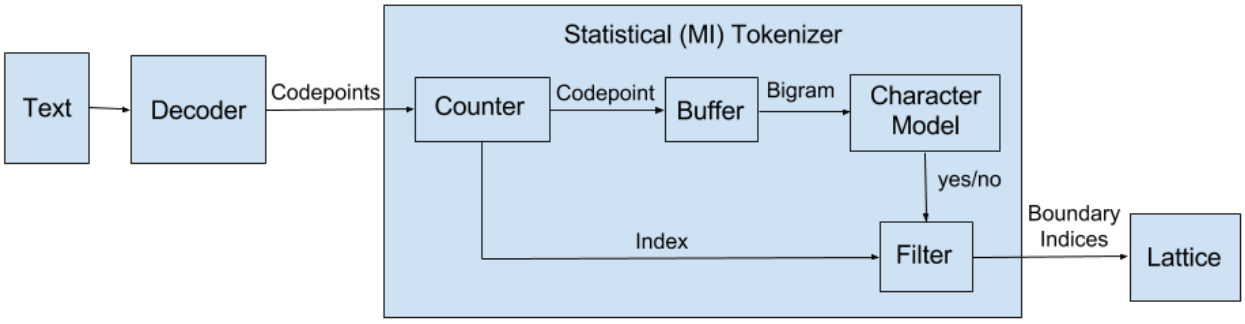

Figure 3.4: Statistical Tokenization

3.4 Hybrid Tokenization

I tested two methods of combining statistical and morphological methods to produce an improved hybrid tokenization system: "filtering" and "filling". Each of these was run with the output of both zero-threshold and gap-threshold statistical experiments, for a total of four hybrid experiments.

The filtering method is an attempt to improve system performance on precision by only initiating a new recognizer to add to the active cohort in the morphological tokenizer at indices where the statistical model indicates a token boundary is likely. This should eliminate spurious tokens that appear as suffixes of other tokens. This approach was simulated by constructing a new stream of (start, end) index pairs from stored morphological output including only those pairs in which the start index exists in the stored statistical data.

The filling method is an attempt to improve performance on recall and address the limitations of lexical-access based tokenization methods (including morphological tokenizers). This approach uses statistical predictions to fill in gaps of unanalyzed codepoints between tokens identified by the morphological model. This was simulated by identifying spans of codepoints not covered by any (start, end) pair in the morphological output, and filling them in with all (start, end) pairs that could be constructed from the predictions of the statistical models over those spans. One downside of this method is that, while it should run in amortized linear ($O(n)$) time in most cases, it does have worst-case quadratic ($O(n^2)$) performance on pathological inputs where the morphological model recognizes no possible tokens. As previously discussed, this would make it unsuitable for on-line usage. As long as

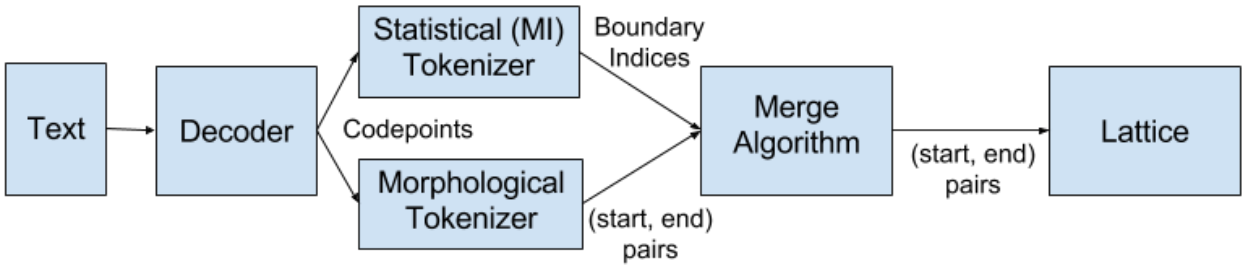

Figure 3.5: Hybrid Tokenization

the morphological model has reasonably good coverage, however, this should not become an issue in practice.

Combined with the individual morphological and statistical experiments, this resulted in a total of seven tokenization experiments run on each language.

3.5 Evaluation

I evaluated all seven experiments on recognition of starting boundaries, ending boundaries, total boundaries (with no distinction between starting and ending), and matched pairs, representing specific tokens.

Since statistical predictions do not distinguish starting boundaries from ending boundaries, the precision measures for starting and ending boundaries considered individually are skewed, as they were determined by the total number of correctly-predicted boundaries that are starting (or ending) boundaries over the total number of undifferentiated boundaries predicted. This makes comparisons with the precision and F-scores for other methods slightly suspect. Recall, however, could be calculated normally for start and ending boundary predictions, and all three measures were calculated normally for the total set of individual boundaries. Token recall was based on the number of (start, end) pairs in the answer key for which both starting and ending indices appeared in the statistical output. Because the statistical tokenizers output only individual boundaries at a time—rather than (start, end) pairs bounding a single token—I omitted precision calculations for complete tokens for these experiments. The lack of pairing information makes precision measures on an un-pruned lattice (containing token hypotheses derived from all possible pairs of undistinguished boundaries)

almost useless—the number of possible hypotheses for all possible pairs grows as the square of the input length, making precision measurements dependent almost entirely on the size of the input, rather than any inherent feature of the algorithm. In the absence of a precision value, F-score is also meaningless. In place of precision for complete tokens, I also evaluated statistical results on what fraction of all codepoint boundaries were identified as possible token boundaries; a lower score on this measure corresponds to a higher effective precision, in terms of reducing the amount of work required by later stages of an NLP pipeline.

For the morphological experiments, I calculated precision, recall, and F-score for all four types of output. Starting and ending boundary sets were merged into a single undifferentiated set to allow direct comparison on that measure with the purely statistical methods. With no generic method of controlling for the effects of differing effective token definitions between corpora and morphological analyzers, I obtained qualitative results by manual inspection of the lists of missed types and tokens for each language.

Since morphological tokenization depends on a morphological analyzer to identify lexical items and other valid tokens, my evaluation results for any particular language depend on the qualities of the analyzer available for that language. In particular, we can expect maximum accuracy and recall measurements when the definition of a "token" used by the creator of a morphological analyzer is the same as the definition of a token used by the corpus annotator. Additionally, we can expect recall to be approximately bounded by the percentage of actual tokens in the corpus that are recognizable by the analyzer in isolation, with any missed tokens being restricted to those that are out-of-vocabulary for the analyzer. In order to control for the quality of the analyzers used, I ran coverage tests to measure the fraction of tokens and types in the answer key that were recognizable by the analyzer for each language.

As in the morphological experiments, I calculated precision, recall, and F-score for starting boundaries, ending boundaries, total individual boundaries, and matched pairs for all hybrid experiments. I also calculated a reduction factor for the filter experiments, consisting of the total number of token boundaries output by the filtering algorithm divided by the number of boundaries present in the raw morphological output.

Chapter 4

Results & Discussion

Table 4.1 summarizes overall system performance on boundary identification for all seven experiments on all three languages. Table 4.2 summarizes performance on token identification. Since F-scores were not calculated for token identification on the pure statistical experiments, results from only five experiments are included here. Note that only one set of results is provided for the filling experiments in either case. Results for both zero-threshold and gap-threshold filling experiments were identical, so the common results for the filling hybridization method are presented only once.

These summary views show significant variability in F-score across languages on almost every experiment, with the notable exception of gap-threshold boundary recognition. These summary numbers, while an important indicator of the range of performance that can be expected from this kind of system using off-the-shelf data sources, do not, however, account for variability in the performance of individual morphological models or mismatches between the effective token definitions used by the answer keys versus the tokenizer. Deeper insights are gained by examining the raw precision and recall scores for each individual experiment, as presented below.

	Arabic	English	Korean
Morphological	0.257299	0.601582	0.511887
Zero Threshold	0.242313	0.280959	0.129573
Zero Filter	0.309639	0.256445	0.214498
Gap Threshold	0.606717	0.539698	0.668813
Gap Filter	0.257299	0.601582	0.511884
Fill	0.257201	0.153340	0.510694

Table 4.1: F-scores for Boundary Recognition

	Arabic	English	Korean
Morphological	0.137505	0.346242	0.284924
Zero Filter	0.087241	0.155838	0.057650
Gap Filter	0.137505	0.346242	0.142457
Fill	0.137448	0.074367	0.283966

Table 4.2: F-scores for Token Recognition

4.1 Morphological Tokenization

Table 4.3 presents the results of the morphological tokenization experiments, while Table 4.4 shows the morphological recognition rates for each analyzer. Surprisingly, the Korean morphological experiments produced identical output when explicitly checking for dependent tokens and when ignoring them. Thus, only one set of Korean results is shown. The token recall rates for each language closely track the token coverage rates of the respective analyzers, which indicates that, given a suitable morphological model, this approach to lattice generation does indeed generalize well across languages.

MADAMIRA, with the best coverage, failed to recognize only four types in the entire corpus, all of which were abbreviations in non-Arabic script. The unusually low coverage of ENGLEX is attributable to multiple factors. First, unlike the other two analyzers, ENGLEX did not recognize punctuation, which eliminates a large group of common types. A more significant contribution to ENGLEX's low performance, however, is due to arguable errors in the answer key, derived from errors in the annotations to the OANC. Specifically, a large number of the types which ENGLEX failed to recognize are sequences like "Alaska–are", "bed-", or "city–to"—agglomerations of one or more words and punctuation which reasonably should be interpreted as multiple tokens. Both ENGLEX and KLEX, however, failed to identify over a thousand genuine word types genuinely present in the corpora.

		Precision	Recall	F-score
Arabic	Start	0.073836	0.998735	0.137506
	End	0.073836	0.998735	0.137506
	Total	0.147672	0.998735	0.257299
	Tokens	0.073835	0.998729	0.137505
English	Start	0.238051	0.808606	0.367818
	End	0.239844	0.814696	0.370588
	Total	0.477895	0.811651	0.601582
	Tokens	0.224087	0.761174	0.346242
Korean	Start	0.173279	0.988841	0.294884
	End	0.172365	0.983624	0.293328
	Total	0.345644	0.986233	0.511887
	Tokens	0.167426	0.955441	0.284924

Table 4.3: Results for Morphological Tokenization

	MADAMIRA (Arabic)	ENGLEX (English)	KLEX (Korean)
Types	0.999764	0.747059	0.924060
Tokens	0.999976	0.761174	0.941961

Table 4.4: Morphological Analyzer Recognition Rates

Neither ENGLEX nor KLEX correctly identified numerical tokens, and ENGLEX failed to recognize punctuation.

Precision scores are more variable. This is attributable to the level of genuine ambiguity recognized by the model, which is in part a feature of specific morphological analyzers but is also largely dependent on the real features of a language. English, with the highest token precision score, has a relatively low rate of smaller tokens occurring as substrings of larger tokens, attributable to the relatively isolating nature of English morphology compared to Arabic (more synthetic) or Korean (more agglutinative). Additionally, the English precision scores are artificially lowered due to the English tokenizer *correctly* identifying individual words in the aforementioned agglomerations like "city–to", which are missing from the answer key.

The extremely low precision seen for Arabic is easily explained by the nature of Arabic orthography. Since short vowels are typically not written, this explodes the number of possible analyses that have to be considered for any given string of text, even compared to

the large number of prefix analyses necessitated by Korean's agglutinative morphology. The results for this particular experiment, however, are likely particularly bad due to the overly-helpful nature of MADAMIRA as previously described; these low precision scores reflect the identification of strings like "المتخهدء" as possible unique tokens, where MADAMIRA has "helpfully" ignored a piece of punctuation which should have simply invalidated that string, thus causing the tokenizer to output a spurious token hypothesis.

4.2 Statistical Tokenization

Having considered the purely morphological approach, we now turn to the two methods of mutual-information based statistical tokenization: zero-threshold and gap-threshold boundary prediction. Recall that the zero-threshold method predicts a token boundary between any two codepoints whose MI score is below zero, while the gap-threshold approach predicts a token boundary between any two codepoints whose MI score is below the largest gap in the set of all MI scores for all possible codepoint pairs in a given language.

4.2.1 Zero-Threshold

Table 4.5 shows the fraction of all codepoint boundaries which were identified as possible token boundaries in each language by the zero-threshold criterion. Surprisingly, the prediction rates closely correspond to recall performance for each language—Korean, with the lowest prediction rate, also shows the lowest recall and F-scores, while English and Arabic are comparatively very close together. This suggests that a large fraction of correct boundaries are being identified essentially at random, such that a larger number of guesses produces a larger number of correct guesses. Per my previous work on English, however, this method is known to show a statistically significant improvement in performance over purely random guessing [21].

Arabic	English	Korean
0.267130	0.209335	0.099916

Table 4.5: Prediction Rate for Zero-threshold Statistical Tokenization

		Precision	Recall	F-score
Arabic	Start	0.162632	0.198899	0.178947
	End	0.156655	0.191589	0.172370
	Total	0.319287	0.195244	0.242313
	Tokens	-	0.040725	-
English	Start	0.121669	0.137783	0.129225
	End	0.266911	0.302261	0.283488
	Total	0.388580	0.220022	0.280959
	Tokens	-	0.046703	-
Korean	Start	0.109952	0.042950	0.061771
	End	0.286540	0.111930	0.160978
	Total	0.396492	0.077440	0.129573
	Tokens	-	0.006073	-

Table 4.6: Results for Zero-Threshold Statistical Tokenization

Table 4.6 presents the remaining results of the zero-threshold experiments. There is remarkable similarity between all three languages on total boundary precision. Compared to my [21] and Rytting's [18] prior results, however, these look quite bad. Since my previous experiments were run on a different subset of the same larger corpus (OANC), this suggests that the similarity of results seen here may simply be coincidental. This approach, therefore, does not seem to generalize well either across languages or even across different corpora within the same language. The practical usefulness of this method is thus determined by the level of overlap between its predictions and those of morphological tokenization, which is addressed in the hybrid experiments.

4.2.2 Gap Threshold

Table 4.7 shows the fraction of all codepoint boundaries which were identified as possible token boundaries in each language by the gap-threshold criterion. In all three languages, a similar high fraction of boundaries were identified. Table 4.8 presents the

Arabic	English	Korean
0.999988	0.999945	0.999973

Table 4.7: Prediction Rate for Gap-Threshold Statistical Tokenization

		Precision	Recall	F-score
Arabic	Start	0.218421	0.999982	0.358530
	End	0.217459	0.995580	0.356952
	Total	0.435880	0.997781	0.606717
	Tokens	-	0.9955628	-
English	Start	0.184808	0.999700	0.311948
	End	0.184812	0.999720	0.311954
	Total	0.369619	0.999710	0.539698
	Tokens	-	0.999621	-
Korean	Start	0.253362	0.990502	0.403510
	End	0.252121	0.985649	0.401533
	Total	0.505483	0.988075	0.668813
	Tokens	-	0.999663	-

Table 4.8: Results for Gap-Threshold Statistical Tokenization

remaining results of the gap-threshold experiments. Precision scores for this method are higher than those for the zero-threshold experiments on Arabic and Korean, and comparable to those from the morphological experiments on all languages. Recall rates for this method are the highest out of all seven total methods tried.

In one sense, the uniformly high recall rates are clear evidence of generalization. Unfortunately, however, the extremely high prediction rate of this model makes it very nearly useless for practical purposes. It fails to capture the unique structural features of each language's lexicon, and is thus little better than the brute-force case of hypothesizing that *every* codepoint boundary is a possible token boundary. It is, in fact, even worse than brute-force in one significant way: although the recall scores shown in 4.8 are very high, they are not 1.0. This means that despite extreme levels of overprediction, the gap-threshold statistical approach still missed some real token boundaries.

Given the marginal difference from hypothesizing every codepoint boundary, the precision scores (and thus F-scores) for this method are almost entirely determined by the average token length in each language. None of the languages tested have sufficiently distinctive bigram-level token-boundary statistics to prevent prediction of spurious boundaries internal to actual tokens. Assuming a uniform distribution of predicted boundaries throughout the text, the number of spurious boundaries internal to an actual token is inversely related to

the number of actual tokens in the text, and thus directly related to average token length. As in the zero-threshold case, the practical usefulness of this method is thus determined by the level of overlap between its predictions and those of morphological tokenization.

4.3 Hybrid Tokenization

We now consider hybrid methods, which attempt to improve performance by combining the strengths of both morphological and statistical models. I tested two hybridization methods: filtering and filling. Filtering aims to improve the precision of morphological tokenization by eliminating token hypotheses which start at statistically unlikely locations. Filling aims to improve the recall of morphological tokenization by predicting additional boundaries in spans where the morphological model could not recognize any possible tokens.

4.3.1 Filtering

Table 4.9 shows the proportion of hypotheses generated by zero-threshold filtering compared to pure morphological tokenization. Table 4.10 presents the remaining results from this experiment. As could be predicted from the poor recall scores obtained in the zero-threshold statistical experiment and shown in Table 4.6, zero-threshold filtering resulted in reduced scores for almost all measures compared to pure morphological tokenization. There are slight improvements to ending boundary precisions for Arabic and Korean, and a slight increase in start boundary precision and token precision on English, but these are far outweighed by the reduced recall rates. Additionally, while useful for comparison with the pure statistical experiments, single-boundary precision is one of the least consequential performance metrics. What matters most are token precision and token recall; i.e., not merely correctly identifying individual boundaries, but correctly pairing them. While the filtering method did achieve its goal of increasing precision by some amount, the fact that it only produced *marginal* improvement on token precision in one language, and had a severely

Arabic	English	Korean
0.262744	0.135199	0.075932

Table 4.9: Filter Reduction for Zero Threshold

		Precision	Recall	F-score
Arabic	Start	0.055894	0.198646	0.087241
	End	0.186050	0.661221	0.290392
	Total	0.241944	0.429934	0.309639
	Tokens	0.055894	0.198646	0.087241
English	Start	0.263174	0.120861	0.165648
	End	0.423457	0.194469	0.266534
	Total	0.686630	0.157665	0.256445
	Tokens	0.247587	0.113702	0.155838
Korean	Start	0.099060	0.042924	0.059895
	End	0.503204	0.218047	0.304255
	Total	0.602264	0.130486	0.214498
	Tokens	0.095346	0.041315	0.057650

Table 4.10: Results for Zero-Threshold Filtering

Arabic	English	Korean
1.000000	1.000000	0.999973

Table 4.11: Filter Reduction for Gap-Threshold

negative effect on recall, makes this an essentially useless combination for practical purposes. The reduction rates, indicating potential run-time efficiency improvements, while impressive, are thus unfortunately irrelevant.

Tables 4.11 and 4.12 present the reduction rates and prediction statistics, respectively, for the gap-threshold filtering experiments. The results for gap-threshold filtering show better performance than zero-threshold filtering, but no better utility. The level of overprediction produced by the gap-threshold algorithm resulted in almost no actual filtering of the morphological outputs. In consequence, the results are identical between these two methods for Arabic and English.

Only tiny (less than $\pm.01\%$) differences exist for Korean, but this was the only language for which any filtering occurred at all. Unfortunately, even with the high level of overprediction in the gap-threshold statistical results, the set of correct boundaries predicted by the gap-threshold statistical algorithm did not completely overlap with the set of correct start boundaries predicted by the morphological algorithm, resulting in a decrease

		Precision	Recall	F-score
Arabic	Start	0.073836	0.998735	0.137506
	End	0.073836	0.998735	0.137506
	Total	0.147672	0.998735	0.257299
	Tokens	0.073835	0.998729	0.137505
English	Start	0.238051	0.808606	0.367818
	End	0.239844	0.814696	0.370588
	Total	0.477895	0.811651	0.601582
	Tokens	0.224087	0.761174	0.346242
Korean	Start	0.173275	0.988789	0.294875
	End	0.172370	0.983624	0.293335
	Total	0.345644	0.986207	0.511884
	Tokens	0.167422	0.955389	0.284915

Table 4.12: Results for Gap-Threshold Filtering

in recall compared to either method alone. The extra time needed to calculate statistical information thus represent a loss rather than a gain in run-time efficiency, for, at best, no improvement. At worst, this hybrid approach results in both worse run-time efficiency and decreased recall.

4.3.2 Filling

Surprisingly, the results for the filling hybridization method were identical for both zero-threshold and gap-threshold statistics on all measures and across all three languages. Thus, only one set of unified results is shown, in Table 4.13. Recall scores are all higher than or equal to those for pure morphological tokenization, but precision scores are lower. On balance, this has resulted in slightly lower F-scores as well, but this does not take into account the relative importance of recall over precision during the lattice-generation step. The largest improvement is seen for English, which also had the lowest morphological recall. Korean, with higher morphological recall scores, shows a much smaller improvement over the morphological tokenization results, and the improvement for Arabic is negligible.

The congruence of results from zero-threshold and gap-threshold filling suggests that, despite the vast differences in results between the zero-threshold and gap-threshold statistical experiments on their own, both had identical outputs on spans of text for which the

		Precision	Recall	F-score
Arabic	Start	0.073804	0.998760	0.137451
	End	0.073802	0.998735	0.137448
	Total	0.147607	0.998747	0.257201
	Tokens	0.073802	0.998735	0.137448
English	Start	0.041307	0.970349	0.079241
	End	0.041891	0.984056	0.080360
	Total	0.083198	0.977203	0.153340
	Tokens	0.038767	0.910667	0.074367
Korean	Start	0.172582	0.989619	0.293908
	End	0.171826	0.985285	0.292621
	Total	0.344408	0.987452	0.510694
	Tokens	0.166744	0.956141	0.283966

Table 4.13: Results for Filling—Both Thresholds

morphological algorithm alone could not identify any possible tokens. This pattern would have to break down given a morphological model with sufficiently low coverage, since results for zero-threshold filling and gap-threshold filling must approach the results obtained from the pure statistical experiments in the limit where analyzer coverage goes to zero. But, even with token coverage as low as 0.761174, as demonstrated by ENGLEX in the morphological experiment on the English corpus, the pattern still holds.

The identical results of both filling experiments are fairly easy to explain for Arabic: with such high initial recall scores due to MADAMIRA's high coverage, there was very little left to fill in, and thus little room for potential variation in the output from different statistical algorithms. This also explains why Arabic sees almost no improvement in recall—there simply aren't that many more correct tokens left to find.

The situation is more complicated for English and Korean, but suggests some commonality in the kinds of types that are likely to be left out of a morphological model. There are two (non-exclusive) obvious possibilities: First, they may be relatively small but frequent. Small gaps occupied by only one or a few small tokens minimize the opportunities for different statistical algorithms to differ, just as is the case with Arabic. Second, they may have uniquely distinguishing MI scores for their bounding bigrams. Specifically, they are types whose sets of bounding bigrams tend to have MI scores that are both less than

or equal to zero, and less than the largest gap in MI scores for all bigrams in the language. Given that the largest gap is above zero for all three languages tested[1], this is equivalent to saying that the boundaries of common tokens unlikely to be recognized by morphological analyzers tend to have MI scores below zero, while the poor recall of the zero-threshold method indicates that many genuine tokens that are recognized by a morphological analyzer have boundary MI values above zero, but below the largest gap.

Both of these explanations appear to fail when confronted with examples like the aforementioned "city–to", which make up a significant fraction of the missed tokens in English, and which is neither particularly short nor possesses particularly uncommon boundary characters. In fact, however, a seven-codepoint missed token like "city–to" *does not* constitute a seven-codepoint gap; in other words, the fact that the specific boundary pair enclosing "city–to" is missing does not imply that *no* boundaries were identified in that span of codepoints. The morphological tokenizer *does* identify tokens in that span—just not the one in the answer key. Thus, the filling algorithm will not be expected to identify any additional boundaries in that span. If tokens like "city–to" were in fact legitimate, this would be a problem to be addressed by filtering, to remove the extra hypotheses inserted into that span, not filling.

In fact, the statistical algorithms are primarily identifying numbers and punctuation marks. Given that ENGLEX, as previously mentioned, misses both of these categories, while KLEX only fails on numbers, this explains why English is improved so much more than Korean, even though it does not quite reach parity: English simply has more of the relevant categories to make up, and results after statistical filling are limited by the number of remaining word tokens which the morphological analyzers cannot recognize.

Filling gaps of course comes at a cost in precision, as noted above; this is seen most strongly in the English data, which is to be expected since the low coverage of the English analyzer provides more gaps to fill in, and thus more opportunities for overprediction. This of course results in a severe drop in F-score relative to pure morphological tokenization, and as previously mentioned the small reduction in precision more than offsets the small

[1]Even without checking the specific numbers, this can be inferred from how much the gap-threshold method overpredicts compared to zero-threshold.

increase in recall for both remaining languages as well. The precision scores for all languages are, however, still much higher than would be obtained by the brute-force method. Given that low recall makes identifying the correct tokenization from a lattice impossible, however, while low precision merely makes it more difficult, this is therefore an acceptable tradeoff in most situations. Furthermore, the token recall and individual boundary recall rates for all three languages, which are all above 0.9, are comparable to the state-of-the-art results given by language-specific tokenizers for Japanese, Chinese, and desegmented English.

Chapter 5

Conclusions & Future Work

Precision scores for tokenization are highly sensitive to the details of a specific language's morphology and orthography. On the other hand, morphological tokenization generalizes well across the three languages I tested in terms of recall. In each case, recall scores are limited by the quality of the morphological analyzer used. Additionally, a simplified statistical model using character bigram mutual information scores with a zero threshold was effective at identifying unknown tokens like numbers and punctuation marks, which are likely to be left out of typical morphological analyses lexicons. This significantly improves recall performance in a hybrid tokenizer across multiple languages as well, even though the statistical approaches do not produce practical results in isolation. As a result, the filling hybrid system was capable of producing recall results comparable to state-of-the-art language-specific tokenizers, without any consideration of specific language knowledge in the construction of the tokenizer beyond inclusion of an independently-developed and reusable morphological model.

As noted, the simplified statistical models employed for these experiments did prove useful for improving the performance of a hybrid tokenizer in a limited domain. More complex models, however, are needed to precisely identify unknown words whose boundary statistics are too similar to other words for MI scores to reliably distinguish. Additional research is also warranted on higher-recall statistical models that maintain reasonable precision to improve filtering efficiency. In principle, separate statistical models tuned for the different use cases could be employed for simultaneous filtering and filling, with an effective filter providing more opportunity for filling model to act by removing spurious morphological analyses and thus correctly identifying more spans as unknown.

While I achieved good results on Arabic, English, and Korean, these three languages do not cover the whole spectrum of variation seen in scripts for natural languages. A better picture of the generalizability—or limits thereof—of these algorithms could be obtained by testing on additional languages like Turkish (which is highly agglutinative and alphabetic), Thai (which uses an alphasyllabary with no spaces), Japanese, and Chinese. Japanese is particularly interesting for the use of multiple scripts simultaneously, with transitions between kanji and hiragana providing significant token-boundary clues which may be easily picked up by a statistical model. Similarly, it would be useful to test on English corpora which include tokens that contain spaces, such as open compounds, borrowings, and idiomatic phrases. ENGLEX has some of these in its lexicon, but the answer keys derived from OANC annotations do not recognize them. Being able to evaluate the morphological tokenizer against an answer key whose tokenization conventions more closely match those of the analyzer's lexicon would improve the case for using this method over simple whitespace splitting even in languages (like English) which do use whitespace for the majority of token boundaries.

Additional small improvements to recall could be made by addressing the problem of overlapping tokens. This is most easily demonstrated by ambiguity in English periods—in a sentence like "The Peloponnesian War concluded in 404 B.C.", the final period is both a part of the token "B.C.", and an independent token marking the end of the sentence—two tokens overlap by a single character. Various forms of non-concatenative morphology or sandhi can create other cases where it is clear that there are multiple tokens, but not clear where precisely one token ends and another begins. This is difficult to handle when tokens are tied to specific positions in the input stream, but would be sidestepped if the data structure used to represent a lattice could preserve the logical ordering of tokens independent of their positions in the input stream.

The precision results are not as encouraging as those for recall. However, this is entirely to be expected in the absence of a path-selection step, which is present in the best language-specific systems with which this new approach can be compared. The low precision scores, and correspondingly reduced F-scores, do not therefore diminish the utility of the filling hybrid approach for lattice generation. Having identified a cross-linguistically useful lattice generation algorithm, the next step in the pipeline is, therefore, identifying a

cross-linguistically useful path selection algorithm to match with it. Building on the work of Suzuki et al. in Japanese [14], it should be possible to design a generic on-line probabilistic parser which would then only require input of a language-specific syntactic model to do simultaneous tokenization, morphanalysis, part-of-speech tagging, and parsing, employing both bottom-up and top-down information at every stage. This would be more complicated, but still doable, given a lattice in which tokens are topologically sorted, but missing a consistent connection to input indices. Given the relative complexity of developing usefully-complete syntactic models, however, there is of course room for exploring the cross-linguistic applicability of shallower statistical models, like the smoothed token n-grams used by Norvig [9] or POS-tagging models (which, like syntactic models, would also assist in disambiguation of tokens with multiple morphological analyses). A slightly more exotic approach would be determining most-likely paths based on word association models like those produced by word2vec [31]. This would pair particularly well with the morphological algorithm's ability to extend recognition to logical tokens like idiomatic phrases and open compounds which contain spaces.

Bibliography

[1] S. Ager. (1998) Thai language, alphabet, and pronunciation. [Online]. Available: http://www.omniglot.com/writing/thai.htm 1

[2] ——. (1998) Written and Spoken Japanese. [Online]. Available: http://www.omniglot.com/writing/japanese.htm 2

[3] R. De Mori and F. Brugnara, "HMM Methods in Speech Recognition," in *Survey of the State of the Art in Human Language Technology*, R. Cole, Ed. New York, NY, USA: Cambridge University Press, 1997, pp. 21–30. [Online]. Available: http://dl.acm.org/citation.cfm?id=278696.278713 4

[4] X. Aubert, C. Dugast, H. Ney, and V. Steinbiss, "Large vocabulary continuous speech recognition of Wall Street Journal data," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2. IEEE, 1994, pp. 129–132. 4

[5] D. B. Paul, "The Lincoln large-vocabulary stack-decoder based HMM CSR," in *Proceedings of the Workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 399–404. 4

[6] J. R. Van Aken, "A statistical learning algorithm for word segmentation," *arXiv preprint arXiv:1105.6162*, 2011. 5, 6, 7, 10, 12

[7] "Treebank tokenization," http://www.cis.upenn.edu/~treebank/tokenization.html, 1999. 5

[8] M. A. Islam, D. Inkpen, and I. Kiringa, "A generalized approach to word segmentation using maximum length descending frequency and entropy rate," in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed., 2007, vol. 4394, pp. 175–185. 6, 8, 11, 14

[9] P. Norvig. (1999) Statistical Natural Language Processing in Python. [Online]. Available: http://nbviewer.jupyter.org/url/norvig.com/ipython/HowtoDoThingswithWords.ipynb 7, 10, 15, 19, 43

[10] F. Peng, F. Feng, and A. McCallum, "Chinese segmentation and new word detection using conditional random fields," in *Proceedings of the 20th International Conference on Computational Linguistics*. COLING, 2004, pp. 562–568. 8, 15

[11] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying Conditional Random Fields to Japanese Morphological Analysis," in *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*, vol. 4. Association for Computational Linguistics, 2004, pp. 230–237. 8, 9, 14, 15

[12] M. Asahara and Y. Matsumoto, "Extended models and tools for high-performance part-of-speech tagger," in *Proceedings of the 18th Conference on Computational Linguistics*, vol. 1. Association for Computational Linguistics, 2000, pp. 21–27. 9

[13] K. Uchimoto, S. Sekine, and H. Isahara, "The unknown word problem: a morphological analysis of Japanese using maximum entropy aided by a dictionary," in *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2001, pp. 91–99. 9

[14] H. Suzuki, C. Brockett, and G. Kacmarcik, "Using a broad-coverage parser for word-breaking in Japanese," in *Proceedings of the 18th Conference on Computational Linguistics*, vol. 2. Association for Computational Linguistics, 2000, pp. 822–828. 9, 14, 43

[15] L. R. Kearsley, "Stream-KIMMO: On-line Morphological Analysis and Applications to Tokenization," 2013, Unpublished. 9, 10, 23

[16] K. Koskenniemi, "A general computational model for word-form recognition and production," in *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the ACL*. Association for Computational Linguistics, 1984, pp. 178–181. 9, 23

[17] R. Daland, "Word segmentation, word recognition, and word learning: a computational model of first language acquisition," Ph.D. dissertation, Northwestern University, 2009. 11, 12

[18] C. A. Rytting, "Segment predictability as a cue in word segmentation: Application to modern Greek," in *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*. Association for Computational Linguistics, 2004, pp. 78–85. 12, 13, 15, 25, 33

[19] M. R. Brent, "An Efficient, Probabilistically Sound Algorithm for Segmentation and Word Discovery," *Machine Learning*, vol. 34, no. 1, pp. 71–105, 1999. [Online]. Available: http://dx.doi.org/10.1023/A:1007541817488 12, 13, 15, 25

[20] M. R. Brent and T. A. Cartwright, "Distributional regularity and phonotactic constraints are useful for segmentation," *Cognition*, vol. 61, no. 1, pp. 93–125, 1996. 12

[21] L. R. Kearsley, "Unsupervised Statistical Word Boundary Prediction With Character Bigram Mutual Information," 2014, Unpublished. 13, 14, 25, 32, 33

[22] N. Ide and K. Suderman. (2002) The Open American National Corpus (OANC). [Online]. Available: http://www.anc.org/data/oanc/download/ 13, 17

[23] J. H. Cloe Jr., "An Evaluation of Electronic Annotated Readers for First Graders in Chinese Dual Immersion to Improve Reading Comprehension and Character Recognition," Master's thesis, Brigham Young University, 2012. 14

[24] E. C. Todd, "The Use of Dictionaries, Glosses, and Annotations to Facilitate Vocabulary Comprehension for L2 Learners of Russian," Master's thesis, Brigham Young University, 2014. 14

[25] S. Ager. (1998) Arabic alphabet, pronunciation, and language. [Online]. Available: http://www.omniglot.com/writing/arabic.htm 17

[26] ——. (1998) Korean alphabet, pronunciation, and language. [Online]. Available: http://www.omniglot.com/writing/korean.htm 18

[27] M. Maamouri, A. Bies, T. Buckwalter, and H. Jin, "Arabic Treebank: Part 1 v 3.0 (POS with full vocalization + syntactic analysis)," *LDC2005T02*, 2005, Linguistic Data Consortium. 17

[28] N.-R. Han, "Klex: Finite-state lexical transducer for Korean," *LDC2004L01*, 2004, Linguistic Data Consortium. 17

[29] A. Pasha, M. Al-Badrashiny, M. Diab, A. E. Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, "MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), 2014, pp. 1094–1101. 22

[30] E. L. Antworth, "ENGLEX: an English lexicon for PC-KIMMO version 1.0," 1991, SIL. 23

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Twenty-seventh Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 3111–3119. 43