Morphological Parsing of Inuktitut

Gina Cook
Engineering and Computer Science
Concordia University
Montréal, QC Canada
ginacook@alumni.concordia.ca

Abstract

This paper provides some motivations as well as an overview of morphological parsing. It focuses on two common algorithms which rely on compression to unsupervisedly learn morphological boundaries on words any language. This paper also discusses some of the pitfalls in using compression to approximate the problem of morphological parsing. Morphological parsing of Inuktitut is investigated using an algorithm based on discovering precedence rules from a seed list and an artificially dense corpora.

Contents

1 Introduction

This paper serves three main purposes. It's primary purpose is to inform the author of various approaches in the field of morphological parsing. Its secondary purpose is to do a case study on morphological parsing of Inuktitut, an agglutinative language with few resources but a dire need for text processing. Its third purpose is to implement a novel algorithm which uses precedence relations between morphemes to constrain the model's hypothesized morphemes.

The paper is organized as follows: Section **??** discusses the various applications in Natural Language Processing which motivates the need for morphological parsing, Section 3 introduces stemming and morphological parsing as well as the distinction between them. Section 4 explains that morphology is systematic, particularly in terms of precedence relations between morphemes. Section 5 explains Inuktitut morphology in particular. Section 6 discusses previous work on the topic of morphological parsing and Section 7 introduces the algorithm developed by the author.

In this paper I attempt to show that morphological information, more specifically the information which is found in agglutinative languages, is easy to access and removes a number of complexities which are found in languages like English where words can be multiply ambiguous between parts of speech, and multiple syntactic parses are available for each sentence due to attachment. In a language where all the “words” are actually just morphemes inside 1 word most of the problems disappear. However certain complications arise when trying to segment words into morphemes, and the assumptions behind current compression techniques which are used to discover morpheme boundaries are actually inherently unable to discover accurate morpheme boundaries.

2 Applications of Morphological Parsing

Morphemes are defined as the smallest meaningful parts of a word. For example, the English words ‘government,’ ‘governments, ‘governmental’ have the same stem (‘government’) but have differing forms as they contain extra morphemes which serve mostly semantic (-s *plural*) or syntactic (-al *adjective*) roles. Morphemes are also called morphs[1]) can be visualized as finite state automaton as shown in fig. 1. Morphology will be explained in more detail in § 4.

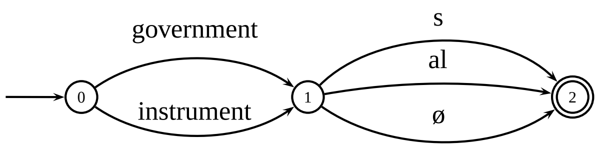


Figure 1: Morphology can be easily visualized as Finite State Automaton.

For Natural Language Processing tasks morphemes (or morphs) can be either a hindrance, in tasks such as Information Retrieval where the goal is to access meaning; or they can be a source of information in tasks such as Named Entity Recognition, Machine Translation, Natural Language Understanding and Natural Language Generation where the goal is to access syntax-semantic relations between words. In fact, simplistic morphological parsing can be a source of a quick boost in accuracy for many tasks such as Part of Speech Tagging and Dependency Parsing. However, this boost is most noticeable in languages where there is a one-to-one mapping between suffix α and the part of speech of its stem and where there is a one-to-two/three mapping between case suffixes the semantic role of its stem.

[1]Morphology means the study of word forms, its coinage is based on Greek *morphos* meaning *form* or *shape*. The term ‘morph’ is often used in Natural Language Processing when there is no need to distinguish between morphemes and allomorphs, or when this distinctions is unclear. The distinction between morpheme and allomorph is rarely needed in Natural Language Processing but in the case of processing Inuktitut it becomes important, thus it will be discussed in § 4.

Ideal languages for morphological parsing include languages with regular (one-to-few, and few-to-one) word formation rules for large classes of words, known dictionaries, and phonemic writing systems. The difference between Phonemic and Allophonic writing systems will be explained later in §(4).

3 Types of Morphological Parsing

This section discusses two types of morphological parsing, stemming in § 3.1 which reduces words to their stem and discards the extra morphemes (prefixes and suffixes) and full morphological parsing in § 3.2 which attempts to find boundaries between morphemes and uses the morpheme to learn information about the words.

3.1 Stemming: remove morphemes

Stemming (reducing a word to its stem) is often used in information retrieval, speech recognition, machine translation and other applications which deal with large vocabularies.

Stemmers are an essential preprocessing step in agglutinative languages such as Turkish and Inuktitut (see fig. 2) and highly-inflected languages such as Spanish where the number of possible forms for each stem range between 20 to 100 forms. To put this into a Natural Language Processing setting, in Turkish and Finnish corpora the number of types (words) continues to grow as the number of tokens (corpus size) increases regardless of corpus type (planned speech in publications vs. spontaneous speech in speech recognition) and corpus size, as shown in fig. 3).

Stemming is less necessary in English where each word has a limited number of forms ranging between 1 to 10 forms and the number of types (words) doesn’t increase as the corpus size increases (fig.3.

a. Agglutinative Languages: Turkish, Finish, Inuktitut
tusaa+tsia+runna+nngit+tualuu+j+u+nga 8 in 1 word
b. Highly Inflected Languages: Spanish
tem+e+r+í+amos 5 meanings in 1 word
c. Compounding Languages: German, Dutch
Anwend+ung+s+programm 4 meanings in 1 word
d. Isolating Languages: Mandarin, Indonesian
méi ting dǒng 1 meanings 3 words

Figure 2: Language Types differ in how many morphemes (meaningful units) they have per word.

In order to do any information extraction or natural language processing, a stemmer which reduces words to their stems must be used to make the feature space more dense so that patterns and statistically significant trends can be detected and machine learned.

In sum, the morphology of agglutinative languages makes it difficult to use statistics and machine learning to find patterns in the corpus. It’s impossible to infer information from one observation. Stemming is successful as it generalizes multiple types (words) to

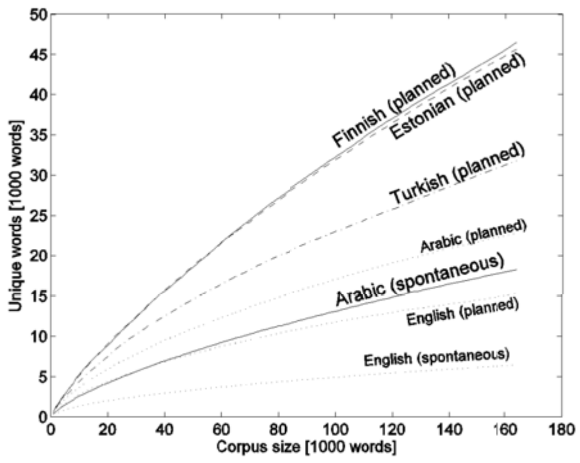


Figure 3: Agglutinative Languages a high type-to-token ratio, resulting in sparse observations per type making statistical analysis nearly impossible (Kurimo 2008).

their common element, a stem. The frequency of stems in a corpus is a function of meaning and content of the text, this is language independent and thus puts Information Retrieval in very different languages such as Turkish and English, back on an equal footing.

3.2 Morphological Parsing: use morphemes

Morphological parsing is a relatively young area in Natural Language Processing; this is most likely due to the historical market of Natural Language Processing, which has mostly been in English, Western European languages, or East Asian languages. These are not languages which encode systematic information in their morphologies, thus the boost in accuracy from morphological parsing would be negligable and too labor intensive to implement when frequency based methodology is able to capture roughly 70-90% of the systematicity. However, needs for NLP are changing with the rise of multilingual documents from the European Union which includes agglutinative languages such as Finnish, Hungarian and Turkish, the interest in morphological stemming, and it's more complicated cousin morphological parsing, has grown. The potential information gain from morphological parsing is yet unknown, but the fact that grammar roles are marked on nouns and that adverbs are contained within the verb they modify, will certainly reduce syntactic attachment ambiguities and likely reduce semantic scope ambiguities and thus yield better performance.

Supervised morphological parsing techniques have high accuracy rates (95% Brent 2001) but require a hand annotated corpus or a lexicon. Unfortunately for agglutinative languages, dictionaries/lexicons can never be exhaustive and must themselves be generative, and thus be morphological parsers. However, there is still light at the end of the tunnel despite this cyclic chicken and the egg problem, and the of the sheer necessity and complexity of building a morphological parser for agglutinative languages. Agglutinative languages are far more systematic than other languages, as will be discussed in § 4. And systematicity is what machine learning learns best (when given the right input). Inuktitut as a newly written language which slated to become the language of education in Nunavut is an important candidate language to study to improve morphological parsing, both for agglutinative languages and for less morphologically rich languages as well.

4 Morphology as the key to grammar

This section will briefly overview the relevant terminology which is required before discussing Morphological Parsing of Inuktitut. Section 4.1 claims that the development of Natural Language Processing methods and tools for Inuktitut are useful not just for Inuktitut, but that the systematicities which can be machine learned in Inuktitut are part of the 5-20% accuracy which cannot be learned about English, from English data; the English data which due to grammatical factors rather than frequency factors is so sparse that patterns are obfuscated by frequency.

4.1 Aggultinative languages provide training ground

Agglutinative languages are far more systematic than other languages, most morphemes are in a one-to-one or one-to-few correspondence with semantic meaning, semantic role or grammatical function. These are the key information which is extracted in Named Entity Recognition, Machine Translation and Natural Language Generation and Understanding. In addition, the ordering of morphemes within a word is fixed, thus the function of a morpheme is wholly decided by its position in the word. In addition, because one word communicates a lot of information, sentences are short and syntactic ambiguity is almost nonexistent. These systematicities do not exist in languages like English where words can appear in many different orders, sentences are long with many possible parses, and words can multiple meanings and multiple parts of speech. English is a veritable chaos when compared with agglutinative languages such as Inuktitut and Turkish.

Most importantly, developing the field Natural Language Processing for agglutinative languages will allow the glass ceiling on accuracy for English languages to drift above its current 80-95% as the remaining errors in English processing are not due to frequency or statistically significant patterns, but are due to language's structure itself. In agglutinative languages on the other hand, it is easy to detect the effects of language structure, so it is a fertile test bed to develop methods which are able to learn these systematicities. Given the very different natures of the two systems, combining methods from both language types will improve the accuracy as the methods which work for English and those which work for Agglutinative languages make complementary errors.

On a further note, the apparent differences between English (where NLP built a solid foundation) and agglutinative languages only exist in written language, where words are indicated with spaces. As a case in point, English and German are both compounding languages and thus have similar grammar, but English keeps a space between words so the morpheme task is easier. Indeed using words as a level for tokenization is an arbitrary choice which is mostly made by ease of digital representations for white space vs. characters.

In conclusion, as one languages word is another language's morpheme, the problem of Morphological Parsing can be seen as an easy Syntactic Parsing task (given that the morpheme order is fully determined), with the additional challenging problem that the "words" are unknown. Unfortunately, the key to unlocking the systematically in Agglutinative languages is segmenting the word into morphemes, and this is the most difficult problem in morphological parsing of Inuktitut.

5 Inuktitut morphology

Before discussing morphological parsing in detail it will help to have a profile of the language under study in this paper. This paper will outline the challenges in creating a lexicon for Inuktitut, an agglutinative language spoken in the Arctic circle in Canada, Alaska and Greenland. Inuktitut has recently become one of two official languages for the Province of Nunavut. As an official language and the media for education in Nunavut, it has become critical that various useful software involving Natural Language Processing be developed, such as spell checkers, grammar checkers, dictionaries, thesaurus, search engines, academic databases etc; all of which require a lexicon and morphological parsing.

5.1 Inuktitut Resources

Johnson and Martin of the Canadian Institute for Information Technology have implemented a number of prototypes for these tasks which are available at www.inuktitutcomputing.ca. They have also published their algorithm for unsupervised learning of Inuktitut mor on the Nunavut Hansard Corpus (Parliament proceedings of Nunavut in English and translated into Inuktitut), their algorithm will be discussed in § 6 where approaches to the problem of morphological parsing are discussed. Some online newspapers and magazines are available as well.

5.2 Inuktitut words

Inuktitut words contain between 2-12 morphemes. Like all languages, the morphemes are strictly ordered such that if negation precedes tense in one word, it will precede it in another word, and all morphemes which instantiate tense will not precede negation. This can be thought of as a template or as a binary branching structure as shown in fig. 5.3. The first line in the example shows the morphs/allomorphs divided by spaces to align information below them. The second line shows the lexical entries for the morphemes. The third line shows the English gloss. The forth line contains the English translation. The precedence relations between morphemes can be read off the template from left to right or from bottom to top off the tree.

5.3 Inuktitut writing

It is important to note that first line is in the romanized (ascii) script, rather than the Inuktitut script. Inuktitut has historically

tusaa	*+tsia*	*+runna*	*+nngit*	*+tualuu*	*+junga*
tusaa	tsiaq	junnaq	nngit	tualuu	junga
hear	well	able	not	much	1sg.pres.decl.non-spec

'I can't hear very well.'

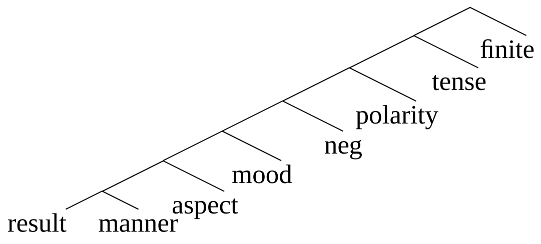


Figure 4: Inuktitut words are systematic templates for syntactic and semantic information

been written in both scripts as tools and fonts (and education) were nto available in the syllabic writing system that was designed for Inuktitut around the turn of the 20th century. However, now that Unicode fonts are readily available and the status of Inuktitut as an official language the trend is going towards the syllabic script which better reflects the patterns in the Inuktitut language.

Inuktitut is traditionally an oral language. The writing system is less than 100 years old and was designed by linguists. Like other agglutinative languages such as Turkish and Finish, the surface written forms do not always match the underlying morphemes; this is where the distinction between morpheme and allomorph become important. In fig. 5.3 we can see that when the actual mental morphemes[2] /tsiaq/+/junnaq/ are said next to each other, the result is the surface allomorphs [tsia]+[runna] where the /q/ and /j/ have coalesced and become an [r]. Thus, the words are written exactly as they are pronounced. This is unlike English, which is written phonemically. For example, the English words 'cats' and 'dogs' both contain the plural morpheme which is written as an $< s >$ but they are pronounced differently. The $< s >$ on cat is pronounced as an [s], while the $< s >$ on dog is pronounced as a [z]. The distinction between morpheme/phoneme and allomorph/allophone can be visualized as input output tree as shown in fig. 5.

5.4 Allophonic writing systems

Being an allophonic writing system means that discovering the morphemes in continuous Inuktitut text is very similar to the problem facing human language learners; not only must the word boundaries be found, but the morphemes could have different forms depending on the sounds around them. In Inuktitut the most frequent allophonic variation is that stop consonants {p,t,k} at the end of one morpheme, change to fully match the first consonant

[2]Morpheme is a coinage meaning *mental form* from the Greek morph+nemos *mind*. Morphemes are traditionally indicated in linguistic circles with /j/ and allomorphs are indicated using []. Orthography is written with $< j >$.

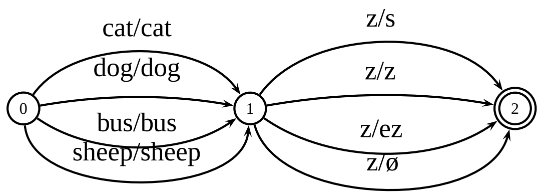


Figure 5: The distinction between morphemes and allomorphs can be seen as a Finite State Transducer, from elements of the Lexicon (morphemes) to elements in context of other elements (allomorphs). English plural is mentally a /z/ and can appear as a [z] or [s] depending on the consonants that are next to it.

of the following morpheme. In fig. 6 the morpheme /tikit/ (line 2) becomes [tikip] (line1) when it appears before [p].

tikip	*+punga*
tikit	punga
arrive	1sg.pres.question.non-spec

‘Am I walking?’

Figure 6: Morphemes are mental representations of morphs. Allomorphs are morphs in the context of other morphs. Allomorphs often show sound changes at their edges to allow the pronunciation of the word to be easier. In Inuktitut stops {p,t,k} the end of one morpheme change to match the stop of the next morpheme.

The fact that these allophonic changes of certain sounds are entirely predictable based on the phonological features of the sounds following it, will become important in later sections; particularly when you consider that most morphological parsers assume that predictable sequences of letters are morpheme internal, rather than between morphemes, exactly the opposite in allophonic writing systems.

6 Approaches to Morphological Parsing

Finding morphemes inside of words, or finding words in sentences can be seen as a problem of compression; how can a text be encoded so that it takes up fewer bytes. One solution is to create a key which assigns each word a number or bit sequence which can represent the word in a text. This key from this light is very similar to a lexicon. The correlation between compression of text, and finding morphemes is shown in fig. 7.

There are roughly four approaches to Morphological Parsing including Transition Likelihood, Minimum Description Length, Paradigms, and Minimal Word Edit Distance, all of which look at it as a problem of compression. I will only discuss examples of the first 2 here as they have been the most successful.

6.1 Transition likelihood and Alignment

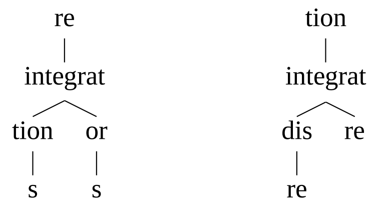
I will discuss three approaches which I have roughly grouped into the category of alignment or transition likelihood, Harris 1995, Johnson & Martin 2003 and Bernhard 2007. These systems are visualized in fig. 8.

```
Data:
@ kati rsui sima giaqanir mik
@ kati maniu laur tu mik
@ tusaa jimmaringulaur sima juq
@ tusaa jimmaringu laur sima juq
@ mali tsiariaqa laur tugut
@ tiki laur tugut

Key:
4        sima
8        laur
9        tu
10       tusaa
5        giaqanir
7        maniu
16       tugut
1        @
14       mali
13       jimmaringu
11       jimmaringulaur
2        kati
3        rsui
6        mik
12       juq
17       tiki
15       tsiariaqa

Compressed Data:
T= 32
1 2 3 4 5 6 1 2 7 8 9 6 1 10 11 4 12 1
10 13 8 4 12 1 14 15 8 16 1 17 8 16
```

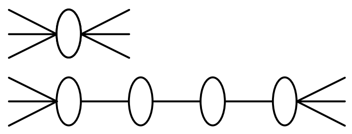
Figure 7: Compression requires a sequence of data and a key, Language requires frequency information and a key.

6.1.1 Harris 1995 Tries

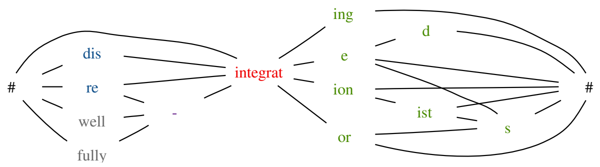
Harris 1955 used tries (tree like branching structures which spell out words as you descend from root to leaf) to discover transition likelihood between letters. A visualization of the trie approach is recreated in fig. 8. However, as can be seen in the figure, this approach is unable to capture some systematicity as many suffixes (in the forward trie) or prefixes (in the backward trie) appear on multiple other suffixes and thus will have repetitive branches.



Forward and Backward Tries, Harris 1955



Hubs and stretched hubs, Johnson & Martin 2003



Alignment, Bernhard 2007

Figure 8: Aproaches to capture transition likelihood between letters. Harris 1955's Tries were less successful as many suffixes (-s) or prefixes (re-) must be repeated at the leaf nodes.
Johnson & Martin's stretched hubs allow sequences of letters to be considered

6.1.2 Johnson & Martin 2003 Hubs

Johnson & Martin 2003 used acyclic[3] deterministic finite automaton (DFA) to represent transition likelihood. Their system performed better on Tom Sawyer than Goldsmith's Linguistica (discussed below). They achieved precision scores of 92% and recall scores of 59% on learning English morphemes. They also reported some results from running their algorithm on Inuktitut. They report the limitations of learning Inuktitut suffixes as there are numerous suffixes within an Inuktitut word. On the reduced problem of finding only Inuktitut roots (not the entire morphology), they report a precision of 32% and recall of 8%.

6.1.3 Bernhard 2007 Alignment

Bernhard 2007 more recently used the equivalent of Finite State Automata to align morphemes. Her approach was very successful in some languages and not in others, achieving an F-measure[4] of 60% accuracy for English, but only 20% for Turkish, an agglutinative language.

6.1.4 Transition Likelihood Summary

Systems designed around tries or finite state automata are able to capture paradigmatic morphology (where a number of morphemes can be substituted for one another). It is yet unclear why these systems perform well in English and German, but less well in representative agglutinative languages like Turkish.

The lower performance in Turkish may be due to the fact that Turkish has vowel harmony, a process where the vowels in the suffix harmonize with the +round -round feature of the root vowel. The Turkish writing system reflects allomorphs rather than morphemes, thus introducing non local systematicity between morphemes as the vowels need to match for rounding. Depending on the prior probabilities set for morpheme length, this may result in fragmentation of the automaton. Morphemes would be divided into three sections with a changing vowel in between, or it may result the opposite, super-morphemes which are in fact composed of multiple morphemes with matching vowel harmonies as they are more likely to follow each other than to be followed by their unharmonic allomorphs.

Perhaps preprocessing Turkish to 'undo' vowel harmony, or having two automata, one for +round roots, one for -round roots could allow the models to learn the Turkish morphological system better.

6.2 Minimum Description Length (MDL) HMMs

In this section we turn to another type of compression algorithm, strives to reduces the file size of a corpus by recursing until very small improvement is made.

6.3 Brent 2001 MBDP-1

In the domain of supervised morphological parsing, Brent &Tao 2001 (building on Brent 1993) used Minimum Description Length (MDL) to find word boundaries (not morpheme boundaries) in continuous text. They tested their algorithm on Chinese (PH Corpus of Newspapers) and English (Hansard Corpus of Canadian Parliament with word boundaries removed). After training on segmented text, their algorithm achieved roughly 95% precision and recall on the English Hansard corpus, and similar accuracy in the Chinese newspapers. The algorithm (named MBDP-1) works by bootstrapping its own lexicon off the segmented training corpus.

In their paper they discuss a common trend in Morphological parsing, that large training corpora do not necessarily yield better results on testing corpora. This often attributed to the foreign words and technical words which are introduced in larger corpora, and larger vocabularies. Foreign and Technical words do not follow the system of the language and thus introduce patterns which

[3]In an acyclic Finite State Automaton the final node doesn't transition to the start node, i.e. no probability relations between the end of one word and the next word should be encoded in the model. This is different from many Hidden Markov Models implementing Part of Speech Taggers where probabilities are kept across sentence boundaries. I used Resnik 1999's implementation of an HMM to explore my data and ran into the problem that I was unable to stop the cyclicity which was built into the HMM, short of processing word by word. This is when I realized what they meant by 'acyclic.'

[4]F measure is the harmonic mean of precision (how many of the results returned are correct), and recall (how many of the correct results were returned by the system).

the algorithm over generalizes, and erroneously tries to apply to words native to the language.

6.3.1 De Marken 1995 Composition and Perturbation

De Marken 1995 worked on segmenting English with the spaces removed, with a view to investigate the mechanisms that children might use to learn a language. He used MDL to find the most useful (predictable and frequent) string of sounds to encode in his lexicon. He began with all letters as words, and then grew the words so that frequent combinations like 'th' and 'ng' became words, then 'ing' etc until full words or even full collocations were stored in the lexicon if they were frequent enough and improved the compressed size of the encoded corpus. Minimum Description Length operates on the principle that using smaller byte sequences for very frequent patterns reduces the overall file size in bytes.

De Marken introduced the concepts of composition (where morphemes are hierarchical structures internally composed of smaller morphemes, down to individual sounds) and perturbation, the level in the morpheme's structure where the meaning is not compositional and thus it is useful to give that level to morpheme status. These concepts can serve as a bridge between 'morphemes' which are found by compression (often simply frequent sound patterns) and 'morphemes' which are expected in natural language tasks.

6.3.2 Goldsmith 2001 Linguistica

Goldsmith 2001's Linguistica software uses MDL to find signatures, groups of suffixes that appear on a set of roots; in many cases these signatures closely approximate what grammarians refer to as paradigms. His software achieved 83% precision on the Brown English Corpus. Goldsmith notes a common error in morphological parsing which I also found, that final letters in a stem are often included in the suffix. This is also a problem Inuktitut as discussed in § 5.4, where transitions between morphemes are actually more predictable and frequent than morpheme internal sequences.

6.3.3 Creutz 2005 Morfessor

Creutz 2005, 2008's Morfessor software also uses MDL to find morphemes, however it is more general than Linguistica and show's a rather consistent performance despite language type in the 2008 Morpho Challenge where it serves as the baseline algorithm against which all others are compared. In Morfessor, words are represented as three latent morph categories (prefix, stem and suffix) and one noise category. This is recreated visually in fig. 9 morphs can be interpreted as the strings which transition between categories. Context sensitivity is introduced via transition probabilities in the automaton on transitions from one category to the next. These morphs and transition probabilities are learned using a recursive morpheme segmentation, Verterbi dynamic programming to find maximize the likelihood of the HMM and then resegmenting the words again.

Morfessor comes in two flavors, one using Maximum Likelihood (ML) and another using Maximum a Posteriori (MAP) to arrive at a final transducer (Creutz 2005 notes that these have been shown to be equivalent by Chen 1996). Morfessor includes parameters such as morpheme length and length distribution which can allow the model to better learn different languages. Morfessor achieves a very respectable f-score of 70% in English and 65% in Finish, an agglutinative language.

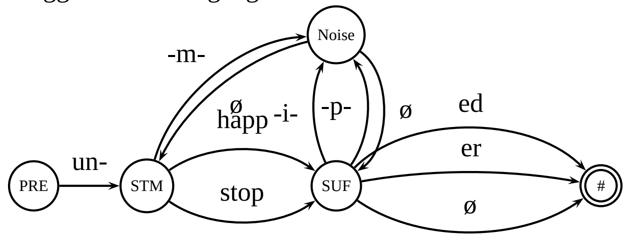


Figure 9: A re-created approximation of what Morfessor's Hidden Markov Model (HMM) might look like. It contains 4 latent (unobserved states), transition properties are not show as this is a toy model.

6.3.4 Exploring Morfessor

I initially wanted to use Morfessor to learn Inuktitut morphology but in the process I realized that the assumptions behind MDL and compression techniques are error-prone in Inuktitut to such a large degree that all MDL algorithms achieve low F-scores (less than 25%accuracy) on Inuktitut. Morfessor achieved a 7% precision and 7% recall rate on my Inuktitut corpus which is described in § 7.

After examining the source code for Morfessor I found that the algorithm was ordering morphemes in alphabetical order before the initial search for morphemes. Hypothesizing that the initial segmentation that Morfessor performs have important consequences for the end model, I changed the source (see fig. 10) so that Morfessor was working off a vocabulary list which was in suffix order (alphabetical from the end of the words). The majority of systematicity in Inuktitut is at the ends of words, so if Morfessor finds these patterns early its segmentations, it will be more accurate. I tested Morfessor on both alphabetical and suffixal orders using a small sample Inuktitut text from Inuktitut Magazine 2009 v.104. The segmentations were then checked and compared using the Inuktitut Morphological Analyzer at http://inuktitutcomputing.ca/Inuktitut/Uqailaut/analyser.jsp?lang=en . Unfortunately, the Inuktitut Morphological Analyzer is still a prototype and is unable to parse many words. For words that I was able to check fig. 10 shows the segmentation differences from running Morfessor with an alphabetical vs. suffixal ordered list. I found a 10% increase in segmentation, 23 out of 277 words had different segmentation. Of these words, 1 changed segmentation and 2 lost segmentation. The representative words of these pattern are shaded in gray in fig. 10.

7 My Algorithm & Implementation

Now that I have reviewed the important considerations of Morphological parsing in general and in agglutinative languages like Inuktitut in specific I will discuss my algorithm and its current implementation. My algorithm capitalizes on the fact that precedence relations with words are strictly followed. In addition, a template

```
Original source sorted in alphabetical order
before calling Morfessor:
$(ZCAT) $< | sed 's/^[0-9]*/1/' | sort -u |
gzip > $@
TMPS += traindata.nowcounts.gz

My modified source sorts in suffixal order:
$(ZCAT) $< | sed 's/^[0-9]*/1/' | sort -u |
rev | sort |rev | gzip > $@
TMPS += traindata.nowcounts.gz

Results:
22 more words out of 278 were segmented
1 in 278 changed segmentation
2 in 278 lost segmentation
277 stems were found from both source
51 suffixes were found from the apha order
81 suffixes were found in the suffixial order
```

Words were more segmentation were found or changed				
Malil+lugu				
malillugu/STM		malil/STM	lugu/SUFF	
		follow	'future'	
Nuna+minik				
nunaminik/STM		nuna/STM	minik/SUFF	
		land	3sg.possess	
Uqausi+vini+nginnik				
uqausi/STM	vininginnik/SUFF	uqausi/STM	vini/SUFF	nginnik/SUFF
		language	past	3pl.possess
Inuusiliriji+nginnik				
inuusiliri/STM	jinginnik/SUFF	inuusiliriji/STM	nginnik/SUFF	
		Experienced worker	3pl.possessive	
Inuusir+mik Inuu+sir+mik				
inuusirmik/STM		inuu/STM	sirmik/SUFF	
		Be alive	Habit+unspec	
Words where segmentation was lost				
Avatiliri+nirmik				
avatiliri/STM	nirmik/SUFF	avatilirinirmik/STM		
limits+change	abstract noun+ 3pl.gen relfexive			

Figure 10: I changed Morfessor so that the initial search is on lists organized by suffixal order. On a small test sample this yielded roughly 10% gain in accuracy.

which encodes these precedence relations can be learned from a corpus. Finally, using a small seed of morphemes (15) it is possible to learn both morpheme boundaries and the allophonic changes which occur at those boundaries. I will first begin by discussing my considerations that went into building the algorithm.

7.1 Considerations

In the process of exploring the literature on Morphological Parsing and attempting to run various available algorithms on my corpora to compare their performance, (I ran Morfessor, Linguistica, Goldwater and Resnik's Part of Speech tagger (the latter two are not discussed above for space reasons)) I realized that there was something these algorithms were doing that didn't apply to allomorphemic writing systems, like the one used for Inuktitut, and many other agglutinative languages to a lesser degree. Compression algorithms (the most popular algorithms in morphological parsing) operate on the assumption that predictability of the next character implies word status. Most authors are clear that they don't mean that compression is the whole story, a true mental lexicon is more than just compression. It seems to me that in many languages where orthography reflects almost nothing of the sound patterns in the language, compression can indeed capture up to 70% of the morphemes. However, once the orthography includes systematic sound patterns (which are very systematic as discussed in § 5.4) the compression algorithms are discovering phonotactics and allophonic variation rather than words.

However, the hypothesis that compression works when the orthography does not reflect sound patterns (phoneticians) is untested and probably only true to a small degree. Blanchard and Heinz 2008 tested Brent's MBDP-1 on phonetically transcribed English, which has a high orthography to sound pattern relationship and the algorithm still performed rather well at around 65% percent. Of course with Blanchard and Heinz' modified MBDP-1 which uses phonotactic information it was able to obtain 75%.[5]

What is certain, is that because of the allomorphy discussed in § 4, it is the *edges* of words that are in fact the most frequent and systematic parts of words, rather than morpheme internal letter sequences. In this section I will introduce my algorithm and implementation for simultaneously learning the alternations which occur at word boundaries and learning the morphemes of Inuktitut. I will begin with introducing my corpus.

7.2 Inuktitut Corpus

I originally began working with the Nunavut Hansard corpus of 2,660,000 words (downloaded from the National Research Council's Inuktitut COmputing website). After working with it for over a month I started to suspect that there were problems with trying to use the corpus for statistical analysis.In short, I was discovering the Hansard had lots of what appeared to be spelling mistakes, particularly with double consonants, double vowels and variation between q or no q. Some examples are shown below.

```
Examples of spelling variation:
10 aaggaaqtuqangittuq
6 aaggaaqtuqanngittuq
2 aanniaqannangittulirijikkuni
5 aanniaqannangittulirijikkunni
```

Given the fact that almost every word in the corpus looked like it had an alternative spelling which was equally frequent, I was almost certain that they were spelling mistakes. In their paper "Aligning and Using an English-Inuktitut Parallel Corpus" Johnson et. al 2003 discuss the factors involved in creating the Nunavut Hansard corpus.

It appears that the transcripts were created by translating from the original English to Inuktitut and were typed in romanized

[5]Phonotactic refers the interaction of sounds and patterns of sounds, in this case at the edges of morphemes.

(ASII) Inuktitut rather than syllabics. Thus the irregularities in the spelling might be due to who was typing the words. After all, the speakers of Inuktitut know if a word has a long vowel or a long consonant or a [q], these are recoverable spelling errors which they might not even consider important for readability[6]. As there is no spell checker yet for Inuktitut it is difficult for the translators to use a standardized orthography. So yet again, working with a new language in NLP is a bit of a chicken and egg problem.

I first attempted to make the corpus more systematic by removing all long vowels and consonants. Then I noticed pairs like those shown below which appear to be different dialects (looked up nirsa,niqsa). I researched dialectal variation in the Inuktitut literature and found that there are numerous Inuktitut dialects which pronounce words differently, particularly those containing q and r which frequently appear as allophones of one another in most dialects. I concluded that as Inuktitut has a short written history, speakers of different dialects might spell differently. I was seeing some connection with the low recall rates reported in Johnson & Martin 2003 on the Nunavut Hansard corpus.

```
Examples of r/q variation:
32 mikiniqsa ni
10 mikiniqsauju ni
3 mikinirsa ni
37 mikinirsauju ni
29 mikiniqsa ni
8 mikiniqsauju ni
```

To remedy my lack of corpus I decided to look for Inuktitut documents on the web to create my own corpus (I have done this before by spidering a Romaninan newspaper with good results). Documents in Inuktitut are relatively rare, and hard to find (hence the development of an Information Retrieval tool for Inuktitut on the National Research Council's Inuktitut Computing website).

After some searching I found Inuktitut Magazine which is published in pdf format in Inuktitut syllabics, Romanized (ASII) Inuktitut, English and French. A veritable wealth in terms of parallel corpora. I extracted the romanized Inuktitut from 2009 volumes 102-104. Each volume yielded roughly 10,000 English words and 6,000 Inuktitut words. I take this to indicate that written Inuktitut corpora are roughly 1 half the size of English corpora due to the lengthy words in Inuktitut.

In total I processed 3 issues of Inuktitut Magazine, to yield an Inuktitut corpus of 17,000 words which could be considered the equivalent of an English corpus of 30,000 words, certainly if we count morphemes rather than words. I had no problems with data quality using the Inuktitut Magazine. I will attempt to get the rest of the issues which are available online (25 issues, although there appears to be a link translation problem so only 3 are currently accessible) or by contacting the publisher.

[6]Many dialects of written Arabic do not write vowels, as these are predictable from the context where the word is used.

7.3 Corpus2morphology

My implementation is available at http://users.encs.concordia.ca/v_cook/research/Inuktitut/bin/ and can called using the bash script titled `corpus2morphology` with a corpus as an argument. The algorithm is essentially designed to operate the way that I do when I start working on a new language. I first ask a speaker for a dense group of words and make hypotheses about the relations between the morphemes. In this sense, the corpus serves as a speaker, and I retrieve only words which are relevant to morphemes that I'm currently working on. This reduces the noise which could affect the learning task and reflects the selective hearing which has been documented in the psycholinguistics literature (auditory illusions where we hear words that we can recognize). The initial dense mini corpus ranges between 1 to 30 words depending on the morphemes which were selected to form the dense corpus.

7.3.1 Implementation

The algorithm is currently implemented as series of Perl and bash scripts to perform modular parts of the algorithm. Each script is upgradeble without affecting the entire implementation. I consider it a rough draft for my summer project, which will incorporate the information I learned while reading the morphological parsing literature and developing my final project into an evolving algorithm using genetic programming which implements the basic concepts discussed here, but taking into consideration a larger number of factors.

I used these scripts to see what are the entrance and exit conditions of different steps in the model. This implementation is language independant and can be run on any corpora of any language but it is not recursive as I ran into memory problems because of my inefficient data structures. I will need to improve my data structures for my summer project. I'm leaning toward Java and storing each word as an object but I will need some guidance from computer scientists before I implement it. If I'm able to achieve high F-scores on the Morpho Challenge 2008 corpora, I would like to submit my system to the Morpho Challenge 2009 in August.

7.3.2 Overview and order of steps

I will use the `corpus2morphology` script shown in fig. 11 to introduce my algorithm. To begin the script sets the language to Inuktitut so that the output files will be named language dependant. Next the script removes previous output files by calling `morphology2clean`.

The first step in the algorithm is to take the corpus and tokenize it into a vocabulary list containing frequency information about each word using `corpus2wordlist`. This word list is then input into Morfessor to get a ranked list of possible morphemes using `wordlist2rankedpossiblemorphs`. From the ranked possible morphs a seedlist is created. Using `seedlist2precedencerelations` the precedence relations between morphemes (such that roots precede suffixes but suffixes do not precede roots) are put into a file called precedencerelations. Next the script creates an initial dense corpus from the seedlist by taking words from the corpus which contain 1 of the root seeds, 1 of the medial seeds and 1 of the suffix seeds using `seedlist2initialdensecorpus`. Next this initial corpus is processed for two pieces of information, first it is processed to extract the precedence relations between individual morphemes using `initialdensecorpus2morphrelations`, second the initial dense corpus is processed to extract the precedence rules for final sounds of the first morpheme, and the second sound of the second morpheme `morphrelations2phonotactics`. After this the morphrelations are converted into a new (expanded) template) using `morphrelations2template` and the process begins again using `precedencerelations2densecorpus`. In the next sections I will discuss each of the scripts and the motivations in more detail.

7.3.3 Corpus2wordlist

The `corpus2wordlist` script takes an ASII text as input, tokenizes on non letters and creates 3 frequency lists which can serve as input to the later stages of the algorithm. The script also produces a list of the longest words, measured by number of syllables.[7] The script is shown in fig.12 and a sample output of words sorted by suffix order and by length in syllables is shown in fig. 13.

The frequency lists are either available by frequency order, alphabetical order or suffix order. For my model of Inuktitut I chose the suffix order as the number of terminal suffixes in the language is roughly 50, which is far fewer than the number of roots in the language. Thus by choosing suffix, order the data is organized by part of speech (-mik only appears on nouns, and -tunga only on verbs) and is presented in a dense manner to the algorithm. The choice of suffixal order is the best choice for almost all of the worlds languages with the exception of a few languages which have no morphology at all.

The script starts with a corpus rather than a word list so that it is possible to return to the corpus later to check for a word in context to learn meaning information for the morphemes. This corpus serves as the ‘outside world’ for my machine learner. As with human learners, I expect some grounding, or some seeds or bootstrapping will be needed to fully capitalize on the systematicsity of morphology in agglutinative languages. This remains distant and perhaps not so useful path of implementation for the future of my parser, as well as all morphological parsers that I found in the literature.

7.3.4 Wordlist2rankedpossiblemorphs

I need a list of possible morphs which serve as seed candidates in order to create a seed list of interesting morphs to investigate. As Morfessor can output a vocabulary list with segmented words in any language, the output of Morfessor when sorted by frequency can serve as a ranked list of possible morphemes. It also provides an (unreliable) metric for whether they will appear initial, final and

[7]Syllables play an important role in the length of most morphemes in most languages, morphemes consisting of 1 syllable are the most frequent as suffixes, and morphemes consisting of 2 syllables are th most popular for roots. Thus the number of morphemes per word can be roughly approximated as Total sylables-2. This varies by language and is a parameter which can be set to a default value and learned by the model as it goes.

```
Sample output of sorted by suffix order:
1 piusiqaqtiqqaummaanga
1 ullurumitaarijumannirmaanga
1 kiluliulaurmaanga
1 ajuqqaanga
6 angajuqqaanga
1 unialaurtaanga
1 kiigumalaurtaanga
1 uqautilaurtaanga
1 nammininirigianga
1 takugianga
1 qirlirtulianga
1 tamagijarianga
1 tusarianga
1 anaanatsianga
1 isumagisimajanga
1 niruartaugutigisimajanga
1 isumaliarisimajanga
1 titiraujaqsimajanga
1 isumalirutigililaursimajanga
1 atulaursimajanga
1 kataijanga
1 upigijanga
1 takuksautitsijjutigijanga
1 katimavigijanga
1 tukisinarutigigunnamijanga
1 tauksiarlsijanga
2 tigujanga
1 takujanga
1 ilanga
1 gavamanga
1 taimanga
1 puigutjaisimagunnanituinnanga
2 nunanga
1 sivuniksanga

Sample output by length(in number of sylables):
17 saimmatitsigasuarutaujaarnaturinalauqsimavuq
17 allasimajuliuqpaliagutiqarumaliqsungattauq
17 aatutiqangitsiammarittuuqutigilaursimajara
16 uqalimaagaksaliarijauqattalirajartuunnik
16 tusaumautiqattautitsiarunnarnirsaugajarnirmut
16 sivumugiallagutiqaqsimaliraluartillugit
16 piruqpalialaqititaugutigingunnatanginnik
16 pinasuagatuarijaunngusutsimagalaaganilu
16 katiqatigiinnirsaugutigijunnarsigajarmauk
16 isumaksarsiurutiqallattaaqattaraluarpita
15 taimannganimmarialunitauqataummigamilli
15 pivallirtitsijummariujunnarasugillugu
15 pigunnaniqsauqigigajannguataraluaminnit
15 kiinaujaqaqtitaugutigigunnatanginnilu
15 kiinaujaliurasuarutiksalirinasuarluta
15 isumagiqasiutigiaqangikkaluarpitigut
15 ikpigusugiallagutiginirsarilirakkit
15 aviktursimaugutigivalliatuinnartattinnik
15 aulatsigunnarnirsaujjutigijunnarsunijjuk
15 angiqatigiigasuarutiqarumaalirmijut
```

Figure 13: Corpus2wordlist sample output for Inuktitut

```
#!/bin/bash

LANGUAGE="Inuktitut"

echo "Removing all files and memories..."
#./morphology2clean
./morphology2clean $1 #if you want to remove the word lists too

echo "Creating a wordlist..."
./corpus2wordlist $1

echo "Creating  rankedpossiblemorpheme lists..."
./wordlist2rankedpossiblemorphs $1-suffixOrder

echo "Creating initial precedence relations..."
./seedlist2precedencerelations $LANGUAGE-seedlist

echo "Creating an initial dense corpus..."
echo "       (contains a hard coded seed list,
change this to follow from above step in later versions)"
./seedlist2initialdensecorpus $1-suffixOrder

echo "Creating precedence relations between morphemes..."
./initialdensecorpus2morphrelations.pl $LANGUAGE-densecorpus

echo "Creating a list of permissible morpheme boundaries ie, z > a, in xyz > abc ..."
./morphrelations2phonotactics.pl $LANGUAGE-morphrelations-frequency

echo "Creating a (new) template by generalizing individual morphs to their class..."
./morphrelations2template $LANGUAGE-morphorelations-frequency

echo "Creating a new dense corpus..."
./precedencerelations2densecorpus $1-suffixOrder
```

Figure 11: Corpus2morphology calls all the other scripts in the correct order with the correct argument so that another user can see how the algorithm is implemented.

```
#!/bin/bash
if [ -z "$1" ]; then
   echo usage: $0 directory "Provide a corpus. This script will generate a
    word list for the corpus: sorted by suffix order, sorted by frequency order
    and sorted by alphabetical order. "
   exit
fi

FREQ="frequencyOrder"
SUFFIXES="suffixOrder"
ALPHA="alphabeticalOrder"
SYLABLES="lengthInSylables"

# From Unix for Poets

#zcat $1 > corpus #if its a gzip corpus

tr -sc '[A-Z][+/][a-z]' '[\012*]' < $1 | sort | uniq -c | rev | sort |
rev | sed 's/^ *//' > $1-$SUFFIXES
#cat $1-$SUFFIXES | grep '^2 ' > $1-$SUFFIXES-2

tr -sc '[A-Z][+/][a-z]' '[\012*]' < $1 | sort | uniq -c | sort -nr |
sed 's/^ *//' > $1-$FREQ
#cat $1-$FREQ | grep '^2 ' > $1-$FREQ-2

tr -sc '[A-Z][+/][a-z]' '[\012*]' < $1 | sort | uniq -c |
sed 's/^ *//' > $1-$ALPHA

tr -sc '[A-Z][a-z]' '[\012*]' < $1 | sort -u > words
tr -sc '[aeiou\012]' ' ' < words | awk '{print NF}' > syl
paste syl words |sort -nr | sed 's/\t/ /' > $1-$SYLABLES
rm words
rm syl

#rm corpus # if its a gzip corpus

# keep all files in the memory folder instead
#mv $1-* memory
```

Figure 12: Corpus2wordlist create a vocabulary list in 3 formats from a text corpus divided on non-letters.

medial in a word (initial+ +medial+ +final). These positions are equivalent to prefix, stem and suffix in English, and Root, medial suffix and suffix in Inuktitut.

The script `wordlist2rankedpossiblemorphs` can call Morfessor with three possible options; either with no frequency information per word, with the frequency information for each word and using the length of morphemes at the median of 5 with a gamma probability distribution (the gamma distribution differs from the normal distribution by having a long tail. The morpheme length of the gold standard (the list of roots and suffixes from Inuktitut Computing) is roughly a gamma distribution with a mean length of 5 as shown in fig. 14.

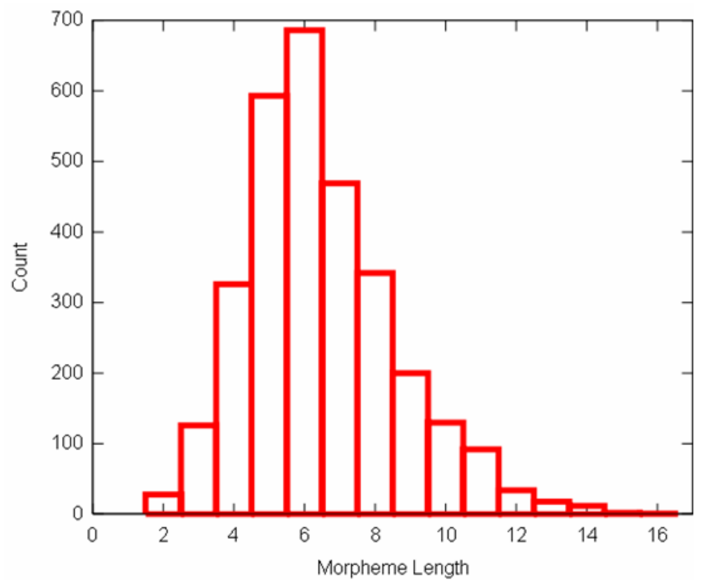


Figure 14: Morpheme length in Inuktitut roots and suffixes roughly approximates a gamma distribution with a mean of 5 characters long. (‘Gold standard’ is taken from root and suffix lists provided by Inuktitutcomputing.ca)

Despite knowing the actual distribution for Inuktitut (see fig. 14) I found that the default option gave the best results. Once the segmentation results from Morfessor are obtained they are filtered to just the lines which contain segmented words (ie. not the headers of the output file) are grep-ed out. Those lines are piped into sed which replaces all word boundaries (+) with a marker on each morpheme (+ +) so that its position word internally (initial, medial and final) are encoded in the spacing. Then the top 15 lines of this output can be used as a seed list. Each seed comes with its position information, initial+ +medial+ and +final. Depending on the language and the type of algorithm used to find a ranked list of possible morphemes (Morfessor is one of many options) the composition of the seedlist could be theoretically all initials, all medials or all finals.

Between initial, medial and final, the most useful morphs for segmentation are the medials, as you get 3 morphemes and 2 morpheme boundaries for the price of one morpheme. Of course finding medials unsupervisedly is very tricky and error prone, so I looked at some of the proposals which Morfessor offered for medials. Many medials it found were only 1 or 2 letters long (see fig. 16) and were not likely to be morphemes.

I looked up the medials that were 4 letters long in the Inuktitut Morphological Analyzer online. Morfessor found the morph *sima* which means ’to be in a state acquired through a completed action’ according to the Inuktitut Morphological Analyzer. I should note that the Inuktitut Morphological Analyzer is a conglomeration of grammars and field work notes of various people. Often the glosses it provides doesn’t make any linguistic sense, for example -tunga they gloss as nominitive:his her, its. His her its are possessive, why not put he, she it if its nominative? In this case I would hope their gloss refers to what linguists refer to as result state. What matters for this paper is that *sima* is a recognized morpheme in Inuktitut, so I added it to my seed list. Although the Inuktitut Morphological Analyzer didn’t have it, I added *laur* which I know from a grammar of Inuktitut (Mellon 1995) is an allomorph of *lauq* which is one of the past tenses. I also choose *nngit* ’negation’ as it causes morphological changes in the stems before it because its long ng causes the previous consonant to delete.

The intuitive way to pick seeds might be to pick ‘pristine’ morphs which are real mental morphemes. It is very important to note that I chose a variety of seeds. I chose not just morphemes, but also allomorphs so that I would find phonotactic information that would indicate the complementary distribution of the allomorphs. For example, if I only chose morphemes which appeared with their underlying final consonant {p,t,k} then I would only be able to segment off following morphemes that started with vowels (the only time the consonants appear). By having both forms with the {p,t,k} and without I get data for both and can discover that roots end in {p,t,k} on when they precede morphemes that start with vowels. In this sense, my seed list bootstraps both allomorphy and morpheme boundaries. Finding morphemes in different environments is something that we do when doing field work in a new language, although it is usually a second step done as part of the phonology of the language rather than the morph-syntax of the language (and often involves two different linguists, a specialist in phonology, and a specialist in morph-syntax).

The output of Morfessor can also be viewed with the latent categories labels that Morfessor assigns to the morphemes (PRE for prefix, STM for stem, SUF for suffix, see also fig. 9). As the STM category corresponds what Morfessor hypothesizes for medials I also looked at the Morfessor output for STM using its own category lables. The stem category is a hodgepodge of initials and medials (as Morfessor assumes that words can start with a stem, or a prefix). A selection of the output which were 4 letters or longer is shown in fig. 17. Morfessor found *qatigii* which according to the Inuktitut Morphological Analyzer is a combination of *qati* and *gii* so this segmentation would not introduce any error if it were used as a seed list. Morfessor also found *liri* meaning ‘to work with, on’ and *jju* ‘reason or cause.’

From these investigations (which took some linguistics and resourcefulness to interpret the output of the NRC’s Inuktitut Mor-

```
#! /bin/bash
$LANGUAGE="Inuktitut"

### Step 1  Use Morfessor to find frequently occurring strings which could indicate morphemes
echo "    Not using Morfessor to generate rankedpossible suffixes or seedlist.
The seedlist is hard coded";

#choice 1: use Morfessor with all frequencies of words set to 1
#zcat $1 > wordlist
 sed 's/^[0-9]*/1/' < $1 | sort -u | rev | sort | rev > traindata
./morfessor1.0.pl -trace 3 -data traindata > $1-morfessorseg-nofrequencies
rm traindata

#choice 2: use Morfessor normally
#./morfessor1.0.pl -trace 3 -data $1 > $1-morfessorseg-usedfrequencies

#choice 3: use Morfessor with a gamma distribution on morpheme lenght,and
median length of 5 (gamma distribution is like the normal bell curve but
with a long tail or more deviation on the positive side)
#./morfessor1.0.pl -trace 3 -gammalendistr 5 -data $1 > $1-morfessorseg-gamma

#compress the output if you want to
#gzip $1-morfessorseg

###Step 2 Filter Morfessor output into a ranked list of suffixes (final morphs),
 adverbial/roots (medial morphs) and roots/prefixes (initial morphs)

#get lines with words on them, and change + to be attached to morphs from
 xyz + abc to prefix xyz+ +abc  suffix
grep '+' <$1-morfessorseg-nofrequencies | sed 's/ + /+ +/' >$1-morfessorseg-nofrequencies

#get a frequency count of each morpheme in each position
./corpus2wordlist $1-morfessorseg-nofrequencies-positionsensitive

#print out the first 15 frequent morphs as a seedlist, replacing xyz+
 as xyz=root xyz, +xyz+ as xyz=medial, +xyz as xyz=suffix
sed 15q < $1-morfessorseg-nofrequencies-positionsensitive | sed
's/\^\([^+]\)*\+$/\1=root/' | sed 's/\^\+\([^+]\)*\+$/\1=medial/' | sed
 's/\^\+\([^+]\)*$/\1=suffix/' > $LANGUAGE-seedlist
```

Figure 15: Wordlist2rankedpossiblemorphs

```
#774 + r +
#653 + u +
#547 + ni +
#374 + qa +
#353 + ti +
#347 + si +
#336 + a +
#270 + tau +
#268 + i +
#264 + ksa +
#259 + sima +  'to be in a state'
#254 + nginni + nothing
#239 + li +
#232 + q +
#228 + uti +
#204 + vi +
#187 + ngi +
#183 + tu +
#183 + lir +
#170 + laur + allomorph of lauq past
#169 + mi +
#166 + titsi + nothing
#163 + ta +
#161 + ju +
#160 + liri + 'to work with, on'
#156 + qaq +
#154 + nga +
#153 + ja +
#133 + ji +
#132 + mmari + mmarik 'certainly'
#122 + titau +
#121 + liur + liuq 'building progressively'
#118 + vallia + 'probably'
#116 + ngit + nngit 'negation'
```

Figure 16: Wordlist2rankedpossiblemorphs outputs a list of

```
#115 ngit STM
#110 tillu STM
#104 ikajur STM
#102 qatigii STM  - okay combination
of qati and gii
#100 qanu STM
#94 suru STM
#93 sivu STM
#87 liri STM - 'to work with, on'
#81 saqqi STM
#81 uqalima STM
#80 jju STM - jjut [jjut,jjuti] 'reason
or cause' (starts with a long consonant
so it deletes a preciding conson),
(after q becomes q+jj=r)
#80 nuna STM
#79 katima STM
#77 kiinauja STM
#76 ullu STM
#73 tusa STM
#71 ilinniar STM
#71 titi STM
#70 iqqa STM
#69 uti STM
```

Figure 17: Wordlist2rankedpossiblemorphs output in Morfessor format, where STM refers to what Morfessor guesses is a stem. It also contains what I refer to as medials so I looked here for possible seeds.

phological Analyzer) and from looking at a grammar of Inuktitut (Mellon 1995) I created the seedlist shown in fig. 18.

```
Initials:
isuma .v 'think'
kati  .v 'gather'
tusaa .v 'listen' /tusaa/ or /tusaq/
malik .v 'follow' /malik/
mali allomorph of malik
tikit .v 'arrive' /tikit/
tiki allomorph of tikit
nuna .n 'land'

Medials:
sima 'result state'
lauq 'past'
 eg [tarrijar+ta+u+laur+sima+juq]
 /tarrijaq+taq+uq+lauq+sima+juq/
 attend.movie+repetitively+frequently+
 previously+perfect+he 'he has often
 gone to the movies, he used to go
 to the movies often.'
laur   allomorph of  lauq
nngit 'negation'
vallia 'probably'

Finals:
tunga .v 'I do something' [junga]
jara .v 'I do it' /tara/
mik .n 'unspecific noun' [mii, miik, mi]
```

Figure 18: Seed list compiled with the suggestions from Morfessor, and checking them in an Inuktitut grammar (Mellon 1995) and with the Inuktitut Morphological Analyzer on Inuktiut Computing.ca

While implementing the `wordlist2rankedpossiblemorphs` script investigation I realized that my terminology might be more language general (initial, medial and final) than Morfessor's terminology (Prefixes, Stems and Suffixes) which allows Stems to appear initially and medially. In my system, if something is found more initial than an 'initial' a new position in the template is created, and the map from template position to prefix vs stem can only be done with some external semantic or meaning information, not from the form of the morpheme itself. In looking at Morfessor's output not as my terms (initial+ +medial+ and +final) in its own terms (pre stm suf) it brings home the limitations of the Morfessor model which were seen visually in the HMM in fig. 9. It doesn't distinguish between the ordering among suffixes, and it cannot. Morfessor is not designed to discover the template in Inuktitut, or capitalize in the restriction that a template places on morpheme hypotheses. Although having a number of possible categories/positions in a word can increase the complexity of an algorithm, the fact that the categories are strictly ordered (not factorially ordered) brings the complexity back down to a manageable level.

7.3.5 Seedlist2precedencerelations

The script `seedlist2precedencerelations` creates an initial template shown in fig. 19 which says that the word boundary marker can precede a root, or a suffix. (I used @ to indicate word boundaries rather than #, as # indicates comments in bash and I felt using @ could still lead to bugs, but probably fewer bugs than #).

7.3.6 Seedlist2initialdensecorpus

The script `seedlist2initialdensecorpus` takes any one of the set of initials, followed by anything, followed by any one of the set of medials, followed by anything, followed by any one of the set of finals, followed by anything; essentially it finds words which contain known morphemes in the proper order, but garbage/potential new morphemes can appear between them, as well as at the beginning and ends of the words. In this way, with the seedlist of 3 categories of morphemes, we get a new template with 7 positions if some 'garbage' is found at the beginning, between all morphs, and at the end of the word. In reality, using the seedlist described in fig. 18 this yielded words between 4 and 7 morphs long. If the seed list only contained initials (or finals) the new templates would be maximally 2 morphs long. If the seed list contained only medials then the new templates would be maximally 3 morphs long. Thus the rate at which the full template is learned depends on the types of morphs which are seeded (medials provide the best seeds), and the morphs which serves as seeds depend on an algorithm that ranks possible morphemes. Most algorithms availible would do this based on compression techniques, thus the seeds (if genereated automatically with out a linguistic expert) could be bad seeds. On the other hand, hand gernerating a list of seeds takes a linguist nonexpert in inuktitut roughly 4-5 hours to complete, that's human time well spent.

Using the seedlist shown in fig. 18 the initial dense corpus was composed of 6 words, ranging between 3 and 5 morphs. The initial dense corpus is shown in fig. 21. Note that none of these words appeared more than once in the corpus, but by looking for any member of a set of morphemes 6 words were returned. Increasing the size of the sets would yield more information. Reducing the number of required morphemes (in this case it was 3, one from each set) will also yeild more words in the dense corpus.

7.3.7 Initialdensecorpus2morphrelations

The script `initialdensecorpus2morphrelations` extracts the precedence relations between individual morphemes in each word so that the relations can be later analyzed to get information about which sounds can be in contact across morph boundaries (phonotactics) and to build a new template for another iteration of looking for these new morphemes. This script is shown in fig. 22.

The output of the `intialdensecorpus2morphrelations` (shown in fig. 25) is next analyzed to find phonotactic information.

```
#!/bin/bash
# start the precedence relations for a template/grammar

LANGUAGE="Inuktitut"

echo "   Initially assuming an optional prefix and obligatory root+suffix..."
echo "       Underlying representation @prefix+root+suffix@ or @root+suffix@"
echo "                (this corresponds to argument+head assumptions about language)"
echo "
@ > prefix
prefix > root

@ > root
root > suffix
suffix > @
"> $LANGUAGE-precedencerelations_initial

echo "   Creating precedence rules for seed morphemes..."
cat $LANGUAGE-seedlist > $LANGUAGE-precedencerelations_initial

#echo "   Initially assuming a template of prefix root medial suffix..."
#echo "prefix root medial suffix" > $LANGUAGE-template

cat $LANGUAGE-precedencerelations_initial > $LANGUAGE-precedencerelations-history
cat $LANGUAGE-precedencerelations_seedlist >> $LANGUAGE-precedencerelations-history

echo "
A history of the precedence rules are kept in $LANGUAGE-precedencerelations-history.
   This file is kept to gather probability information. It serves as a long term memory,
   is not used as input to later steps."
echo "
The generalized frequency information for the precedence rules are in
   $LANGUAGE-precedencerelations-frequency. This is the file which is used
   as input in later steps."
sort < $LANGUAGE-precedencerelations-history  |uniq -c >
 $LANGUAGE-precedencerelations-frequency
```

Figure 19: Precedence relations are initialized to allow root+suffix, and prefix+root+suffix.

```
1 + +kati+ +rsui+ +sima+ +giaqanir+ +mik+ +
1 + +kati+ +maniu+ +laur+ +tu+ +mik++
1 + +tusaa+ +jimmaringulaur+ +sima+ ++ +juq+ +
1 + +tusaa+ +jimmaringu+ +laur+ +sima+ +juq+ +
1 + +mali+ +tsiariaqa+ +laur+ ++ +tugut+ +
1 + +tiki+ ++ +laur+ ++ +tugut+ +
```

Figure 21: Initial dense corpus as a result of the seedlist in fig. 18. The corpus expanded from a template of 3 positions to a template of 5 positions for tusaa+jimmaringu+laur+sima+juq . In addition note that the relative ordering can be learned for the two medials, laur and sima so that the medial category can be split.

7.3.8 Morphrelations2phonotactics

Similar to the script which extracts precedence relations between morphemes `morphrelations2phonotactics` writes precedence relations between the final sound of one morpheme and the initial sound of the following morpheme. The script is shown in fig. 23 and the out put is shown in fig. 24.

The output of `morphrelations2phonotactics` shows that the rule that {p,t,k} are deleted before consonants is true of this dense corpus. Only vowels and [r] are found before other morphemes. Thus once the algorithm has exausted the explorations of the maximal template from the words present in the corpus it can use this information to hypotheize morpheme boundaries internal to the morphemes which it has evidence for and it can learn that 2 morphemes are allomorphs of eachother if they appear in complementary distribution (for example one appears after morphemes which end in a vowel, and another minimally differnet morpheme differning only in the initial letter appears after morphemes which end in a consonant). The information about the individual morphs is retained in the morphrelations-history file for precisely this type of look up.

7.3.9 Morphrelations2template

Next the list of morphrelations (fig. 25) are used to build a new template to create a new dense corpus. In the output we can see that the initial dense corpus found 4 different initials (all of which are seeded roots), initials can be followed by maniu, rsui tsiariaqa, and jimmaringu (all of which are indeed composed of 1 or 2 morphs in Inuktitut). These are inturn followed by laur or sima. Interestingly the dense corpora already provides information that laur precedes sima. Laur and sima (the seeded medials) are followed by tugut and juq (seeded finals) and also by a new category composed of giaqanir. These new morphemes are entered into the predence rules (my equivalent of a lexicon) and are used as substituttion sets to build a new dense corpus.

7.3.10 Itterate over corpus until the template is learned

The final script repeats the process until the template no longer enlarges after a few itteration. The process continues, using the substitution sets which it has learned to gather more data until

```
@ > k
@ > m
@ > t
a > g
a > j
a > l
i > l
i > m
i > r
i > s
i > t
k > @
q > @
r > m
r > s
r > t
t > @
u > l
u > m
```

Figure 24: Morphrelations2phonotactics outputs precedence relations across morpheme boundaries.

roughly the entire corpus has been analyzed. The process of finding the full template is rather quick, I estimate that it would be roughly 5 itterations using the seeds which I have chosen. With less informative seeds or with seeds that badly segment the words the rate will be much slower as more and more garbage will be introduced into the template rather than real words.

After the template is learned the question of what sort of information (phonotactic or different suffix combinations to learn parts of speech) to focus on next is unclear to me.

One of the most important questions I have is that of many machine learning problems, when should the algorithm stop? I would say if the number of slots in the template grows at a constant rate the algorithm should start over because some of the initial seeds were not morphemes and so they are creating garbage when they are segmented off.

8 Conclusion

In this paper I wanted to understand the parameters which can be learned using a corpus with allphonic informaiont and a genetic algorithm. In reading the litterature on Morpholgocial parsing and in generating my own dense corpora I was able to see what sorts of statistical information I can take advantage to capture the patterns which I use to make morphological analysis when working with a speaker of that language.

I will conclude with a very long word in English, depending on how you segment it: *You win some and you loose some.* Al-

```
2 @ > kati
1 @ > mali
1 @ > tiki
2 @ > tusaa
1 giaqanir > mik
1 jimmaringu > laur
1 jimmaringulaur > sima
2 juq > @
1 kati > maniu
1 kati > rsui
1 laur > sima
1 laur > tu
2 laur > tugut
1 mali > tsiariaqa
1 maniu > laur
2 mik > @
1 rsui > sima
1 sima > giaqanir
2 sima > juq
1 tiki > laur
1 tsiariaqa > laur
1 tu > mik
2 tugut > @
1 tusaa > jimmaringu
1 tusaa > jimmaringulaur
```

Figure 25: Initiandensecorpus2morphrelations output

though syntactic and semantic information is easy to find in agglutinative languages, getting to the information by segmenting the mophemes is a very difficult task, one that requires external information like interaction of sounds at morpheme boundaries, the notion of heirarchical structure, precedence relations and perhaps even semantic grounding. In short, analysing grammar is difficult, and this is largely an area which only slow ground has been made. Hopefully the data crunching power of machine learning will make it possible to make futher ground than a solitary field worker who struggles to organize their data and allow NLP to shatter the current 90% accuracy cieling.

References

Brent, Michael R and Xiaopeng Tao. 2001. “Chinese text segmentation with mbdp-1: Making the most of training corpora.” In *39th Annual Meeting of the ACL*, pages 8289.

Creutz, Mathias and Krista Lagus. 2005. “Inducing the Morphological Lexicon of a Natural Language from Unannotated Text.” In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR’05)*, pages 106-113, Espoo, June. http://www.cis.hut.fi/projects/morpho/

deMarken) de Marcken, Carl. 1995. “Acquiring a lexicon from unsegmented speech.” In *33rd Annual Meeting of the ACL*, pages 311313.

Goldsmith, J.A. 2001. “Unsupervised Learning of the Morphology of a Natural Language.” *Computational Linguistics*, 27:2 pp. 153-198.

Johnson, Mark. 2008a. “Unsupervised word segmentation for Sesotho using adaptor grammars.” In *Tenth Meeting of ACL SIGMORPHON, pages 2027*. ACL, Morristown, NJ.

Johnson, Mark. 2008b. “Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure.” In *46th Annual Meeting of the ACL*, pages 398406. ACL, Morristown, NJ.

Kanungo, Tapas. 1999. “UMDHMM: Hidden Markov Model Toolkit,” in *Extended Finite State Models of Language*, A. Kornai (editor), Cambridge University Press. http://www.kanungo.com/software/software.html. http://www.umiacs.umd.edu/ resnik/nlstat_tutorial_summer1998 /Lab_hmm.html

Resnik, Phillip. 1999. *Using a Hidden Markov Model.* http://umiacs.umd.edu/ resnik/ling773_sp2009/assignments/hmm.html

```
#! /bin/bash

LANGUAGE="Inuktitut"

#create a dense corpus from a few roots/prefixes, medials/roots and suffixes
rm $LANGUAGE-densecorpus

for i in isuma kati tusaa mali malik tiki tikit nuna
do
for m in sima laur lauq nngit vallia
do
for f in juq tugut mik
do
#grep $i.*$m. $1 >> densecorpus-$1
grep $i.*$m.*$f $1 | sed "s/$i/\+\t\+$i\+\t\+/" |
 sed "s/$m/\+\t\+$m\+\t\+/" | sed "s/$f/\+\t\+$f\+\t\+/" >>
 $LANGUAGE-densecorpus
done
done
done

cat $LANGUAGE-densecorpus >> $LANGUAGE-densecorpus-history
```

Figure 20: Seedlist2initialdensecorpus

```
#! /usr/bin/perl
$language="Inuktitut";
#use File::ReadBackwards;
if($ARGV[0]) {
open("STDIN", "<".$ARGV[0]);
$infile=$ARGV[0];
}else{
$infile="alignedcorpus-unspecified";
}
$morphfrequency= $language."-morphrelations-frequency";
$morphlog= $language."-morphrelations-history";
open("MORPHLOG",">>".$morphlog);

print "\nTaking in an aligned corpus, calculating precedence
relations between morphemes for each (informative) word\n\n";
while (<>){
#&printdebug( $_);
if ($_ =~ /\+\+/){
&printdebug(" This line has an empty morph  ++ in it so its not
the most informative for a new template\n\t$_\n");
}#else{ #if its a full template
chomp;
$_ =~ s/\d|\+//g;
$_ = &trim($_);
$_="\@ $_ \@";
&printdebug ("Remove digits and +\n");
@morphs = split(/\s+/,$_);
&printdebug ("@morphs\n");

# learn relations between morphs in that word
$size=@morphs;
for ($i=0; $i<($size-1);$i++){
print MORPHLOG "$morphs[$i] > $morphs[$i+1]\n";
&printdebug( "$morphs[$i] > $morphs[$i+1]\n");
}
#}#end if to look for long informative words
}#end while for the file loop

close MORPHLOG;
#system("cat $morphlog");
print "    A memory of all encountered precedence relations
between individual morphemes is kept in $morphlog\n";
system("sort < $morphlog | uniq -c  >$morphfrequency ");

print "    Precedence information between individual morphemes
was generalized with frequency counts in the $morphfrequency file.\n";
system("cat $morphfrequency");

sub printdebug() {
my $string = shift;
#print $string;
}
```

Figure 22: Intialdensecorpus2morphrelations

```
#!/usr/bin/perl -w
#
$language="Inuktitut";
#
if($ARGV[0]) {
open("STDIN", "<".$ARGV[0]);
$infile=$ARGV[0];
}else{
$infile="alignedcorpus-unspecified";
}
$interphonotacticslog = $language."-phonotactics-intermorpheme_precedence-history";
$interphonotactics =$language."-phonotactics-intermorpheme_precedence-sorted";
#$phonointra = phonotactics-intramorpheme_precedence;
open("INTERMORPH",">>".$interphonotacticslog);
#open("INTRAMORPH",">>".$phonointra);
while (<>) {
&printdebug("Working on this line $_");
$_ =~ s/^\s*\d*\s*//;
&printdebug("Remove frequency information $_");
@pair = split(/ *> */,$_);
$last = substr($pair[0],-1,1);
$first = substr($pair[1],0,1);
&printdebug("This $last  is the last char of the first word $pair[0]\n");
&printdebug("This $first  is the first char of the  second word $pair[1]\n");

&printdebug("$last > $first\n");
print INTERMORPH "$last > $first\n";

}#end while to add new phonological precedenec at word boundaries

close INTERMORPH;
system("sort -u < $interphonotacticslog > $interphonotactics");
system("cat $interphonotactics");
sub printdebug() {
my $string = shift;
#print $string;
}
```

Figure 23: Morphrelations2phonotactics