Interactive Word Completion for Morphologically Complex Languages

William Lane and Steven Bird
Northern Institute
Charles Darwin University

Abstract

Text input technologies for low-resource languages support literacy, content authoring, and language learning. However, tasks such as word completion pose a challenge for morphologically complex languages thanks to the combinatorial explosion of possible words. We have developed a method for morphologically-aware text input in Kunwinjku, a polysynthetic language of northern Australia. We modify an existing finite state recognizer to map input morph prefixes to morph completions, respecting the morphosyntax and morphophonology of the language. We demonstrate the portability of the method by applying it to Turkish. We show that the space of proximal morph completions is many orders of magnitude smaller than the space of full word completions for Kunwinjku. We provide a visualization of the morph completion space to enable the text completion parameters to be fine-tuned. Finally, we report on a web services deployment, along with a web interface which helps users enter morphologically complex words and which retrieves corresponding entries from the lexicon.

1 Introduction

Indigenous communities in Australia's far north are engaged in a variety of efforts to maintain their languages. In the Kunwinjku-speaking communities of West Arnhem, the challenges are to maintain orality and develop literacy. Technological solutions can support community efforts to create language resources and make them accessible. These solutions include dictionaries, digital archives, and collaborative applications for documentation and language learning (Hunt et al., 2019; Bow et al., 2014; Fillmore et al., 2019). All of these technologies require text input, posing a challenge for communities whose languages are primarily oral, and where writing—especially writing that conforms to an orthography standard—is not an established practice.

In the context of morphologically complex languages, the term "intelligent dictionary" has been applied to systems which morphologically analyze the query in the course of retrieving relevant lexical entries (Johnson et al., 2013; Arppe et al., 2016). This is an important feature for speakers of polysynthetic languages like Kunwinjku, which have multiple morph slots with sometimes large or unbounded sets of fillers. However, effective use of an intelligent dictionary still requires the ability to enter the fully-inflected surface form. For speakers and learners of morphologically complex languages, producing well-formed queries may pose a significant challenge.

Littell et al. make a similar observation in their work on dictionary interfaces for North American languages, where few speakers are "both fluent in the language and trained in a systematic orthography" (Littell et al., 2017). While they address the problem of retrieving dictionary entries given a noisy query, we go further, guiding users as they type by suggesting completions up to the next morph boundary.

The typical definition of "autocomplete" implies word-level completion (Cai et al., 2016; Dunlop and Crossan, 2000), but this is impractical for polysynthetic languages where single words are capable of expressing the meaning of an entire sentence. For low-resource polysynthetic languages, we propose a version of autocomplete which serves a different purpose: guiding writers who are building confidence, rather than being seen as a time-saving tool for literate users. The different context and use case give rise to a different function for autocomplete: completion of the current morph, taking into account its position in the word and the range of legal possibilities here. The feature is akin to preemptive spellchecking. It guides users towards valid possibilities at a granularity which preserves morphological integrity.

This paper is organized as follows. In Section 2 we review recent work using finite state transducers for analysis of morphologically complex languages, along with issues of language modelling and flexible text input. We also present the special challenges of polysynthesis for the autocomplete problem. Section 3 details an approach to flexible text input: We present a finite state algorithm for morph-based autocomplete, and extend this model to allow for spelling variation. In Section 4 we measure the completion space of two morphological analyzers equipped with the extension: Kunwinjku and Turkish. Section 5 introduces an interface to the Kunwinjku Dictionary that incorporates these methods. Using this interface, we solicit feedback from two speakers in Section 6.

In sum, this work presents a reframing of text autocomplete, intended for morphologically rich languages. We propose a novel and portable algorithm for morph-based autocomplete using finite state methods. The application of this model in an intelligent dictionary interface and subsequent trial of the tool with speakers highlights the challenges of designing text-based language technology for oral-language communities who speak morphologically complex languages.

2 Background and Related Work

2.1 Finite state analysis for morphologically complex languages

The proposed approach builds on the established practice of applying finite state transducers for modelling the phonological and morphological systems of natural languages (Beesley and Karttunen, 2003). In recent years, researchers have demonstrated the suitability of finite state models for a variety of morphologically complex, low-resource languages including Cree, Haida, Kunwinjku, Odawa, Tsuut'ina, and Yupik (Snoek et al., 2014; Harrigan et al., 2017; Arppe et al., 2017a; Arppe et al., 2017b; Bowers et al., 2017; Chen and Schwartz, 2018; Lachler et al., 2018; Lane and Bird, 2019).

The FST formalism is implemented by a number of frameworks including HFST (Lindén et al., 2013), Foma (Hulden, 2009), OpenFST (Allauzen et al., 2007), and Pyini (Gorman, 2016). For this work we use Foma, as we are extending previous work on the Kunwinjku FST (Lane and Bird, 2019).

The conventional approach for building an FST morph analyzer is to divide the task into two stages: (a) accounting for the lexicon and morphotactics, and (b) defining the morphophonological rules which use the context at morphotactic boundaries to produce valid surface forms (Beesley and Karttunen, 2003). Many researchers have followed this pattern to build FST analyzers for morphologically complex languages (Çöltekin, 2010; Harrigan et al., 2017; Bowers et al., 2017; Andriyanets and Tyers, 2018; Moeller et al., 2018; Cardenas and Zeman, 2018; Lane and Bird, 2019). Others deviate slightly, e.g., Chen and Schwartz implement a mechanism for stage 2 which applies morphophonemic rules at morpheme boundaries working from left to right (Chen and Schwartz, 2018). In any case, the practice of defining a lexicon of morphs to be concatenated to form the intermediate representation of a word is ubiquitous. Accordingly, the ability to mark morph boundaries at this intermediate level serves as one of the basic assumptions of the work reported here.

2.2 Language modelling for morphologically complex languages

Some researchers have attempted to build language models for morphologically complex Indigenous languages. Maheshwari et al (2018) collected a corpus for Mik'maq—a polysynthetic North American language—and developed n-gram and neural character-based language models. Their results highlight the difficulty of learning predictive models—with statistical or neural methods—from small corpora.

Neural polysynthetic language modelling was also the theme of a workshop whose activities are reported by Schwartz et al (2020). They find that morphological segmentation of the training data resulted in better language models. They deployed their Tensorflow models to a modified Divvun backend (Moshagen et al., 2013) to create a prototype mobile keyboard with predictive text, and propose an algorithm for generating predictions up to the next morph boundary from a neural language model.

TSO		DIR		ASP		MSC1^		BEN		MSC2		GIN		BPIN*		COM		Root*		RR		TAM		*Total*
157	x	3	x	2	x	24	x	2	x	4	x	78	x	32	x	2	x	541	x	2	x	5	=	$4.9x10^{12}$

Figure 1: An estimate for all morphotactically valid verb morph sequences covered by the Kunwinjku FST implemented in Lane and Bird (2019). Slots marked with an asterisk (*) are open class.

2.3 Flexible text input for morphologically complex languages

Kunwinjku is primarily an oral language, and the standardized orthography system is a recent invention. As such, any text-based tool deployed in this context must address the fact that few people command the orthography. Littell et al (2017) address the flexible text-input problem by mapping the user input to an orthographically-agnostic representation. Then, they derive a "comparison form" of the query by collapsing similar sounds into a smaller set of equivalence classes. The unweighted Levenshtein distance between the query and potential results is taken for both the comparison and orthographically-agnostic representations, and a final ranking is achieved by computing the weighted average.

2.4 Special challenges of polysynthesis

Kunwinjku (ISO gup) is a Gunwinyguan language spoken by about 2,000 people in the West Arnhem region of Australia's Northern Territory. Typologically it is classified as polysynthetic, and exhibits many of the hallmark features of that classification: agglutination, synthesis, valence-altering morphology, and reduplication (Evans, 2003). Consider, for example, the Kunwinjku word *benebadyibawong*:

(1)	bene-	bad-	yi-	bawo-	ng	(manu	kabbala)
	3ua.3sg.past-	now-	COM-	leave-	PastPerf	(VE:DEM	boat)

"The two of them left him (in the boat)" [E.10.162]

Figure 1 shows the 12 morphotactic slots implemented by the Kunwinjku morphological analyzer for verbs. In Kunwinjku, some morph slots represent an open class. For example, body part incorporated nouns (BPINs) form a class of nouns which either are body parts or are in some way associated with a body part (e.g., a person's shadow). These can be incorporated into the verb, usually to specify that the verb predicate is true of the body part but not necessarily the whole: *nga-bid-keleminj*, "I-hand-feared" or "I was afraid for my hand" (Evans, 2003, 464). Apart from open classes, some closed morph classes can be large. The first slot represents the class of pronominals, and must be filled in order to make a well-formed verb. The pronominals are fusional morphs which indicate the subject, object, and tense of the verb. The Kunwinjku FST represents this class with 157 fillers.

Next we consider the autocomplete task for a polysynthetic language such as Kunwinjku. Suppose that a user begins typing *benebadyibawong* into a text field. At *bene*, autocomplete would attempt to finish the whole word. One might reasonably expect that the number of possible completed words of an given prefix would be the product of the number of possible morphs in each subsequent slot. However, there is more complexity to factor in: in some slots—like MISC1 which contains adverb, aspect, and quantifier prefixes—multiple fillers can co-occur. For example *woh* is a MISC1 adverbial filler meaning "a little bit" and *djarrk* is a MISC1 adverbial meaning "together," which can co-occur in the word *karri-woh-djarrk-durrkmirri*, "Let's work together a little bit." In terms of implementation, this manifests itself as a self-referencing edge on the MISC1 class, which enables a potentially unbounded stacking of fillers.

Thus the space of full-word completions for a morphologically complex language like Kunwinjku is unmanageable. Even if we ignored cycles, the number of word completions following *bene* is on the order of 10^{10} (Fig. 1). For this reason, we pursue an alternative definition of text autocomplete, one better suited to the morphological richness of polysynthetic languages.

We posit that autocomplete up to the next morph boundary represents a level of semantic granularity more comparable with full word completion in an analytic language like English. It also has the benefit of being more tractable in terms of the number of possible completions at any given point in a word.

```
Multichar_Symbols
rr

LEXICON Root
Pronominals;

LEXICON Pronominals
[1sg.2sg]:0^ Verb;
[3pl.pst]:birri^ Verb;
[3pl.nonpst]:kabirri^ Verb;
[2sg.3sg]:yi^ Verb;

LEXICON Verb
wokmadbu:wokmadbu^ Conj;
yakbu:yakbu^ Conj;

LEXICON Conj
[NonPst]:n # ;
[Imp]:0 # ;
[PstPerf]:~om # ;
[PstImperf]:ni # ;
```

(a) Lexc morphotactic definition which recognizes the verb roots *wokmadbu* and *yakbu* with pronominal prefixes and TAM suffixes

```
read lexc example.lexc
define Lexicon;
define V [ a | e | i | o | u ];

define MorphBoundary "^";
define Cleanup MorphBoundary -> 0;

# Delete vowels preceding ~ marker
define Alt1 V -> 0 || _ "^" "~" .o.
                        "~" -> 0;

define Grammar  Lexicon .o.
                Alt1    .o.
                Cleanup;
regex Grammar;
```

(b) Foma definition showing the composition of the analyzer and some morphophonemic rules

Figure 2: Minimal working example demonstrating the proposed approach

3 Methods

In this section we describe the FST model which generates morph-based completions along with an extension which allows for imprecise input. The main resource assumed in this work is an FST model of morphology in the target language. The starting point is an FST which maps a surface form to its morphological analysis. For example, the analysis of *wokmadbuni*, "I was waiting for you to call," is *1sg.2.past-wokmadbu-PP*.

This approach assumes that the FST model marks morpheme boundaries. A survey of recent work on FST morphological analyzers for morphologically rich languages shows that marking morpheme boundaries as part of an intermediate representation is a common practice (Lovick et al., 2018; Bowers et al., 2017; Chen and Schwartz, 2018; Andriyanets and Tyers, 2018; Snoek et al., 2014; Çöltekin, 2010). Thus, we anticipate that this is not an onerous requirement.

Figure 2 shows morphotactic rules of the language, producing intermediate forms that are processed by the alternation rules. This minimal working example represents the foundation on which we build in the next section.

3.1 Morph-based autocomplete

We start with a morph analyzer, e.g., `Grammar` in Figure 2(b). The process is described in Algorithm 1 using the XFST syntax (Beesley and Karttunen, 2003). We implement the algorithm as an extension to the Foma file of the Kunwinjku morphological analyzer. For example, in order to extend the toy example presented in Figure 2 to perform morph-based autocomplete, we preserve the Lexc file exactly as presented in Figure 2(a), and we modify the Foma file from Figure 2(b) with the FST extension outlined in Algorithm 1. Figure 3 shows that extended Foma file. Algorithm 1 is summarized as follows.

Steps 1, 2: Define important character classes: `Bx` is the set of boundary symbols defined in the FST grammar, `Ax` is any character except boundary symbols.

Steps 3, 4: Steps 3 and 4 define some helper transducers: `InsertBx` optionally inserts a boundary symbol in any context, and `RemoveBx` will remove a boundary symbol in any context.

Algorithm 1 Finite State Morph-based Autocomplete

```
Require: Grammar                ▷ The FST which transforms a valid surface string into its morph analysis
1: define Bx ["^"];                                                  ▷ Define morph boundary symbol(s)
2: define Ax [?-Bx];                                ▷ Any Character except morph boundary symbol(s)
3: define InsertBx [ 0 (->) Bx ];                                                ▷ Optionally insert Bx
4: define RemoveBx [ Bx -> 0 ];                                                           ▷ Remove Bx
5: define UpToBx [ ?+ [ 0:Ax ]+ 0:Bx ];                          ▷ Any string of characters up to Bx
6: define Prefixes [ [ ?* [ 0:? ]* ] .o. [ Grammar Bx ].l ].u;                    ▷ Language of prefixes
7: regex InsertBx .o. UpToBx .o. Prefixes .o. RemoveBx ;                      ▷ Gives morph completions
```

```
read lexc example.lexc
define Lexicon;
define V [ a | e | i | o | u ];

# Delete preceding vowel
define Alt1 V -> 0 || _ "^" "~" .o.
                        "~" -> 0;
define Grammar  Lexicon .o.
                Alt1;
                # NB: omit Cleanup step

define Bx [ "^" ] ;
define Ax [ ? - Bx ] ;
define InsertBx [ 0 (->) Bx ];
define RemoveBx [ Bx -> 0 ];
define UpToBx [ ?+ [ 0:Ax ]+ 0:Bx ];
define Prefixes [ [ ?* [ 0:? ]* ] .o.
                [ Grammar Bx ].l ].u;

regex InsertBx .o. UpToBx .o. Prefixes .o. RemoveBx;
```

Figure 3: The Foma file which implements the morph-based autocomplete extension on the example given in Figure 2

Step 5: Step 5 defines an FST which recognizes any input, `?+`, and transduces all possible character continuations up to a morph boundary symbol, `[ 0:Ax ]+ 0:Bx`.

Step 6: Step 6 accepts the prefix candidates formed by Step 5, `?*`, and recognizes all possible character completions from that input, `[ [ ?* [ 0:? ]* ]`, which we compose with the lower side of the original FST grammar, `[ Grammar Bx ].l`, to restrict recognition to only those prefixes and completions which form full, grammatically-licensed words. Taking the upper side of this transducer, `[ Grammar Bx ].l ].u` has the effect of recognizing only those prefix completions which lead to those licensed words.

Step 7: Step 7 composes the defined transducers to yield the FST which takes input flexibly considering the possibility of a morph boundary symbol at any position, `InsertBx`, uses this input to form prefix candidates, `UpToBx`, restricts the space of prefix candidates using the grammar, `Prefixes`, and finally deletes the boundary symbols from the suggested morph completion strings, `RemoveBx`.

3.2 Handling orthographic variation

With the model defined in Section 3.1, we can map a sequence of input characters to a set of morph completions. While the proposed autocomplete mechanism can be viewed as preemptive spellchecking, it assumes that the first few characters typed are correct. A user could easily find themselves searching down a blind ally: they won't reach the desired word because they made a mistake earlier on. We would prefer the completion model to allow for course correction *after* mistakes have been made. Accordingly, we extend the method to match the input based on string similarity (Levenshtein, 1966). We define the transducer described by Hulden (2013) which makes ***n*** changes to the input (a deletion, an insertion, or a change) where ***n*** is the edit distance parameter:

(2) `define C1 [?* [?:0|0:?|?:?-?]  ?*]<`***`n+1;`***

`C1` is combined with the autocomplete extension described in Section 3.1 by composition with the `Compl` transducer. Specifically, Step 7 becomes:

(3)
```
regex C1 .o.  [InsertBx .o.  UpToBx .o.  Prefixes .o.
RemoveBx];
```

4 Evaluation and Results

We first measured the performance of autocomplete in terms of its reduction of the completion space for a given input. The space of candidates decreases drastically compared to word-based autocomplete for a polysynthetic language. Overall, at any point of character input, the space of morph completions is always smaller than the word-based equivalent by many orders of magnitude.

4.1 Strict vs. Flexible Morph Complete

Figure 4 gives a more comprehensive picture of the space of morph completions for the Kunwinjku model. We generated 10,000 words from the Kunwinjku FST, and calculate the distribution of possible transitions at every character position from 0 to the length of the longest word in the sample. The figure shows that when the first character is typed, the median result set contains ~100 completions, but with a long tail reaching towards an upper bound of ~1,000. The median quickly drops to under 25 with subsequent input, then gradually tends towards 1 as more input is provided.

Figure 5 shows the distribution progression for the Kunwinjku data when passed through the Flexible Morph Complete Model. In this version, we see that allowing edit distance of 1 shifts the early distributions up by nearly an order of magnitude before converging with roughly the same timing as the Strict Morph Complete Model. This chart shows that allowing for fuzzy edit distance early in the sequence would likely be overwhelming for the user, but that the feature could be used effectively later in the sequence without significantly enlarging the size of the expected completion set. We posit that while Levenshtein distance of 1 would likely be insufficient to support spell-checking of a complete word form, it could be well-fitted to our context of morph-based autocomplete. For every character typed, the set of completions is updated, and the possibility of being one character off is considered each time. Therefore the cumulative magnitude of flexibility allowed per word corresponds to the number of characters typed, and sequences accepted as the final form is interactively assembled.

4.2 Portability

In order to establish the portability of the morph autocomplete method, we also applied it to a morphological analyzer for Turkish (Çöltekin, 2010). Turkish is an agglutinative language which makes extensive use of suffixes on nouns and verbs. We used the same process established above with the Kunwinjku model to generate Turkish words and plot the space of completions (Fig. 6). The only adjustment to the completion algorithm is to change the definition of morph boundary symbols (Bx) to those used in the Turkish FST. All other aspects of Algorithm 1 are language independent. While the Kunwinjku model covers the morphotactics of verbs and some nouns, the Turkish model is considered far more complete in terms of its coverage of the full language. This difference is made clear in observing the general shape of Figure 6, which shows that the sample contained words which had no possible completions as soon as 2 characters into the sequence.[1] This indicates that the random sample contains a variety of word

[1]Examination of the data used to generate the Turkish completion space plot shows that their model supports the morphological analysis of emojis and emoticons, and that our random sample contains a number of them. The result is that character indexes 1 and 2 are skewed somewhat to reflect their presence as “morphs” in the language. Our view is that the data visualization should simply express the space of possible completions for a given model, so we leave the data and plot unchanged.

lengths between 2 and 45 characters, which is consistent with the fact that the analyzer claims extensive coverage of the language.

Another feature of Figure 6 is that the median number of completions stays under 25 at nearly any position in the sequence. This demonstrates that Turkish may be a more attractive candidate for morph-based autocomplete than Kunwinjku, by virtue of the fact that it is more likely to return a more manageable number of results for any query in any position.

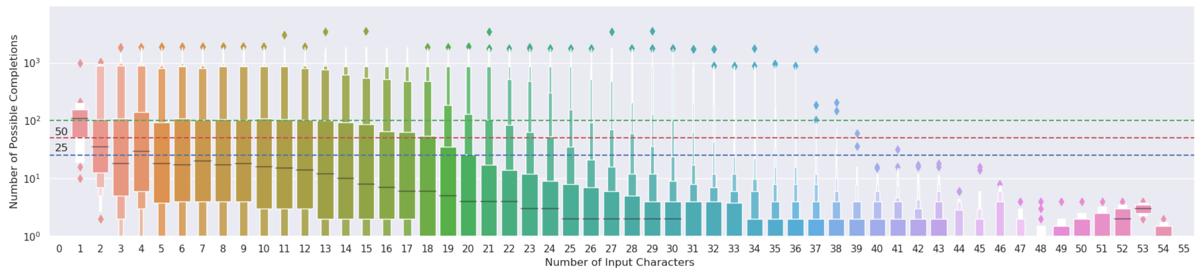


Figure 4: Kunwinjku morph completions, given a sample of 10,000 words generated by the Kunwinjku morphological analyzer

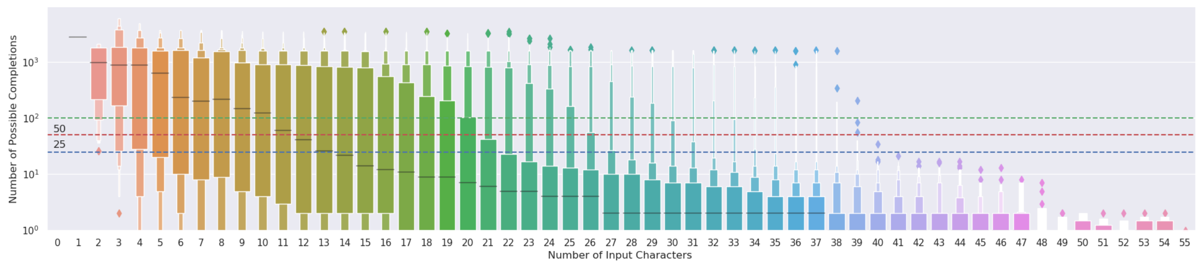


Figure 5: Kunwinjku morph completions with edit distance 1 allowance for current morph completion, given a sample of 10,000 words generated by the Kunwinjku morphological analyzer

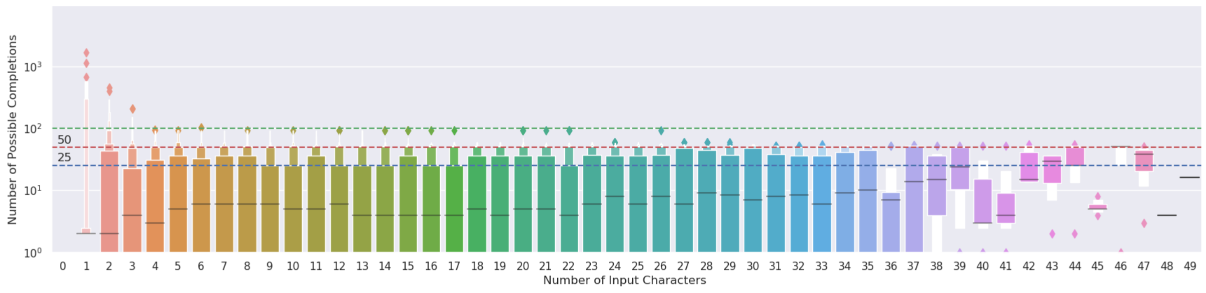


Figure 6: Turkish morph completions, given a sample of 10,000 words generated by the TRmorph Turkish morphological analyzer

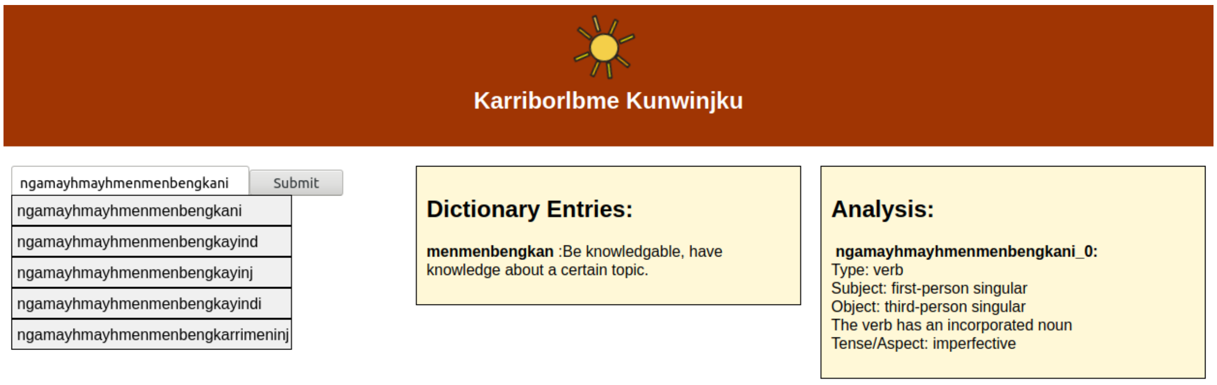
5 Discussion

5.1 Deploying flexible text input for a dictionary interface

We deployed the Strict Morph Complete and Flexible Morph Complete Models as web services, and developed a simple UI which applies these models in an interface for retrieving entries from an existing online Kunwinjku dictionary (Garde et al., 2019).[2][3] Given the findings in Section 4, we decided to

[2]The Kunwinjku dictionary can be accessed at https://www.njamed.com/

[3]The morph completion interface to Njamed can be accessed at https://cdu-tell.gitlab.io/karriborlbme/


Karriborlbme Kunwinjku

ngamayhmayhmenmenbengkani Submit

ngamayhmayhmenmenbengkani
ngamayhmayhmenmenbengkayind
ngamayhmayhmenmenbengkayinj
ngamayhmayhmenmenbengkayindi
ngamayhmayhmenmenbengkarrimeninj

Dictionary Entries:

menmenbengkan :Be knowledgable, have knowledge about a certain topic.

Analysis:

ngamayhmayhmenmenbengkani_0:
Type: verb
Subject: first-person singular
Object: third-person singular
The verb has an incorporated noun
Tense/Aspect: imperfective

Figure 7: The web interface which analyzes inflected queries and displays a dictionary entry. The query *ngamayhmayhmenmenbengkani* "I knew a lot about birds" retrieves a dictionary hit, and a morphological analysis, as well as several alternative morph-completions.

provide text field completions after 3 characters of input. Further, we used Strict Morph Complete to retrieve completions up until 12 characters of input, and Flexible Morph Complete after that. Doing so reflects our effort to maximize flexibility without unduly increasing the size of the completion space.

To improve the relevance of the completions, we switched to using HFST to compile the model, because of its support for weighting lexical entries. Using the Kunwinjku Bible as a corpus, we calculated the probability distributions of morphs over their lexical class, and annotated the lexc file accordingly. This basic approach allows more frequent morpheme compositions to rise to the top of the completions suggestions list. However, concerns about the space scalability of weighted FSTs for large grammars mean that it may make sense to view the ranking problem as independent of the completion model. The exploration of this question, we leave to future work.

5.2 Trying out the system with speakers

5.2.1 Speaker 1

We arranged to sit with a speaker of Kunwinjku and elicit her perspective on using the dictionary web interface fitted out with interactive word completion. Despite her oral mastery of the language, she described herself as having difficulty reading and even more trouble writing. She reported that she can only spell simple words up to five characters long, such as *dja*, "and," or *djenj* "fish."

The session began by opening the existing online dictionary of Kunwinjku in a browser, and observing that typing fully-inflected word forms does not return any results. We then opened the smart text input interface and try dictating some of those same words: *karridjarrkdukmirri*, "we all work together." The participant typed the first eight characters, sounding out the word and considering her options at every keystroke: `k-a-d-i-d-j-e-r-k-`. She exhibited a preference for writing *d* where the standard orthography prefers *rr*, and an approximate knowledge of vowels. Her orthographic choices appear ideal to ensure that she is composing without completion options up until this point. Further, she deviated from the standard orthography more than once before reaching the point where the system backed off to the Flexible Model, meaning that she would not see any suggestions when she reached the 12 character threshold.

We tried a few more words with a similar result: the speaker struggled to navigate them on her own. We observed that with a researcher at the keyboard pointing out possible useful completions for the speaker to confirm, the process of word-building seemed to work better, and the participant showed interest in reading and explaining the resulting analysis and dictionary entries.

5.3 Revisiting orthographic variation

Working with a speaker of the language helped us recognize that orthographic variation can come two forms: (a) systematic errors, which cover the type of variation where the user consciously associates a

```
. . .

define bilabialStops ["b" | "bb"];
define velarStops ["k" | "kk" | "g"];
define velarNasal ["ng" | "ngg" | "ngk"];
define apicalStops ["d" | "dd" | "rd" | "rdd"];
define apicalNasals ["n" | "rn"];
define apicalLateral ["l" | "rl"];
define palatalStops ["dj" | "djdj"];
define flap ["d" | "rr"];

define SystematicVariation flap -> flap .o.
                         apicalNasals -> apicalNasals .o.
                         apicalLateral -> apicalLateral .o.
                         apicalStops -> apicalStops .o.
                         velarStops -> velarStops .o.
                         palatalStops -> palatalStops .o.
                         palatalNasals -> palatalNasals .o.
                         bilabialStops -> bilabialStops;

. . .

regex PhoneticVariation .o. [C1 .o. [InsertBx .o. UpToBx .o.
                                     Prefixes .o. RemoveBx]];
```

Figure 8: The *.foma* which implements phonetically similar graphemes as equivalence classes, and extends the Flexible Morph Complete.

particular grapheme with a sound or class of sounds, and (b) typographic errors, which represents the class of random input errors.

The edit-distance approach targets typographic rather than systematic errors, because it weighs the insertion, deletion, or change of any character equally. Moreover, a low edit-distance threshold can easily be overwhelmed by input from an un-confident user, who may exhibit a number of systematic errors which consumes the quota for random error.

In light of this dynamic, we further extend the autocomplete models to recognize some phonetically-similar graphemes as members of equivalence classes. The velar stops in Kunwinjku, for example, are *k*, *kk*, and *g*; knowing which one to select in any given word can be difficult. By collapsing similarly-pronounced graphemes into equivalence classes we target the likely sources of systematic variation. This approach is similar to the derivation and use of "comparison forms" in the Waldayu dictionary interface apps (Littell et al., 2017).

The flexibility allowed by this extension is narrow enough in scope that it does not impact the sampled distributions of morph completions shown in Figure 4. This allows us to implement it in addition to the edit-distance mechanism of the Flexible Morph Complete Model, which continues to target possible random errors with a low threshold.

5.3.1 Speaker 2

On a separate occasion, we sat with another speaker of Kunwinjku who has mastery of the oral language, and expresses a desire to improve her writing. She picked the word she wanted to construct: *karribidyikarrmerrimen*, "we help each other." She entered `g-a-d-i-b-i-d-`, at which point the third result in the list contained the correct prefix: *karribidyikarrme*. The substitutions *g*∼*k* and *d*∼*rr* are accounted for as systematic variation, as we had updated that portion of the model after working with Speaker 1.

However, the participant was so focused on typing that she did not see the suggestion, and continued typing `g-a-d-i-b-i-d-g-i-g-a-rr-d-i-m-a-`. At this point she paused, noticing the lack of suggestions, and began amending her spelling to explore where she may have diverged. Eventually we worked collaboratively to settle on the intended entry, whose retrieval prompted her to share knowledge of constituent pieces of the word, and how it would be used in context.

The experience with Speaker 2 suggests that our orthographic flexibility settings may still be too restrictive, though it was encouraging to see that additions to the systematic variation part of the model from Speaker 1 had a direct positive impact of the quality of suggestions for Speaker 2. Overall the interaction was positive, and at the end of the 20 minute session Speaker 2 volunteered her interest in returning to work on using the interface again, even suggesting that she could ask members of her family for good words to try out next time.

6 Conclusion

Digital technologies present an opportunity for supporting community goals concerning transmission of language and knowledge. For instance, knowledge captured in spoken language, videos, and photographs can be centrally stored and access-controlled according to community policy. Long distance communication can be facilitated by text messaging technologies. Software which helps automate the building of words or codifies pedagogical activities might build capacity in literacy and language competency.

All of these technologies are accessed through interfaces, and text input is probably the most ubiquitous. However, the morphological complexity of languages like Kunwinjku can present particular challenges: rich morphology lends itself to long words and sparse vocabularies, which confound word-based modelling approaches typically used for analytic languages like English. Moreover, low literacy means low confidence with spelling, and dialectal variation means variability in pronunciation and orthographic convention.

In this work we address some of these challenges by proposing a finite state algorithm for interactively building words by suggesting proximal morph completions. We leverage knowledge of common systematic errors to tune the model's robustness in the face of variation. The model separately targets random error by allowing variation and edit distance of 1 per keystroke. The solution drastically reduces the space of completions compared to full word completions, at a useful semantic granularity. Working within a finite state framework was beneficial because the space of completions is constrained by what we know to be true about the language, and can be easily updated as we learn more.

We deploy the technology in a dictionary interface to explore interactive word building with two speakers, and found that the manually-tuned error tolerances were often not flexible enough to provide guidance to speakers working solo. However, in both cases, building words collaboratively facilitated directed and undirected construction of words, and led to interesting discussion about the orthography and language.

The work reported here provides a proof of concept for a new style of word completion, one where language learners and people with low literacy can interactively build morphologically complex words. Interactive word building for morphologically complex languages can make other technologies—such as assisted transcription, or text retrieval—more accessible, and offers new pathways for transmitting language and culture in the digital era.

Acknowledgements

We are grateful to the Bininj people of Northern Australia for the opportunity to work in their community. Our thanks to Antti Arppe and several anonymous reviewers for helpful feedback on earlier versions of this paper. This research was covered by a research permit from the Northern Land Council, and was supported by the Australian government through a PhD scholarship, and grants from the Australian Research Council and the Indigenous Language and Arts Program.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *International Conference on Implementation and Application of Automata*, pages 11–23. Springer.

Vasilisa Andriyanets and Francis Tyers. 2018. A prototype finite-state morphological analyser for Chukchi. In *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*, pages 31–40, Santa Fe, USA. Association for Computational Linguistics.

Antti Arppe, Jordan Lachler, Trond Trosterud, Lene Antonsen, and Sjur N Moshagen. 2016. Basic language resource kits for endangered languages: A case study of Plains Cree. In *Proceedings of the 2016 CCURL Workshop. Collaboration and Computing for Under-Resourced Languages: Towards and Alliance for Digital-Language Diversity*, pages 1–9.

Antti Arppe, Christopher Cox, Mans Hulden, Jordan Lachler, Sjur N Moshagen, Miikka Silfverberg, and Trond Trosterud. 2017a. Computational Modeling of Verbs in Dene Languages: The Case of Tsuut'ina. In *Proceedings of the 2016 Dene Languages Conference*, pages 51–69. Alaska Native Language Center, University of Alaska, Fairbanks.

Antti Arppe, Marie-Odile Junker, and Delasie Torkornoo. 2017b. Converting a comprehensive lexical database into a computational model: The case of East Cree verb inflection. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 52–56, Honolulu, USA. Association for Computational Linguistics.

Kenneth Beesley and Lauri Karttunen. 2003. *Finite-State Morphology: Xerox Tools and Techniques*. CSLI, Stanford.

Catherine Bow, Michael Christie, and Brian Devlin. 2014. Developing a Living Archive of Aboriginal Languages. *Language Documentation and Conservation*, 8:345–60.

Dustin Bowers, Antti Arppe, Jordan Lachler, Sjur Moshagen, and Trond Trosterud. 2017. A morphological parser for Odawa. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 1–9.

Fei Cai, Maarten De Rijke, et al. 2016. A survey of query auto completion in information retrieval. *Foundations and Trends in Information Retrieval*, 10:273–363.

Ronald Cardenas and Daniel Zeman. 2018. A morphological analyzer for Shipibo-Konibo. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 131–39.

Emily Chen and Lane Schwartz. 2018. A morphological analyzer for St. Lawrence Island / Central Siberian Yupik. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pages 2623–30, Miyazaki, Japan. European Language Resources Association.

Çağrı Çöltekin. 2010. A freely available morphological analyzer for Turkish. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, Valletta, Malta. European Language Resources Association.

Mark D Dunlop and Andrew Crossan. 2000. Predictive text entry methods for mobile phones. *Personal Technologies*, 4:134–43.

Nicholas Evans. 2003. *A Pan-Dialectal Grammar of Bininj Gun-Wok (Arnhem Land): Mayali, Kunwinjku and Kune*. Pacific Linguistics. Australian National University.

Naomi Fillmore, Don Bemrose, and Lala Gutchen. 2019. Language, technology, community nexus: A case study of Erub Mer. In *Causes of Language Endangerment: Looking for Answers and Finding Solutions to the Global Decline in Language Diversity*, University of Sydney. Foundation for Endangered Languages Conference.

Murray Garde, Jill Nganjmirra, and Dan Kennedy. 2019. Bininj Kunwok Dictionary. `njamed.com`. Accessed: 2019-07-19.

Kyle Gorman. 2016. Pynini: A Python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 75–80, Berlin, Germany. Association for Computational Linguistics.

Atticus G Harrigan, Katherine Schmirler, Antti Arppe, Lene Antonsen, Trond Trosterud, and Arok Wolvengrey. 2017. Learning from the computational modelling of plains cree verbs. *Morphology*, 27:565–98.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32. Association for Computational Linguistics.

Mans Hulden. 2013. Advanced finite-state techniques tutorial. `http://clt.gu.se/sites/clt.gu.se/files/mkp/clttutorial.pdf`. Accessed: 2020-01-07.

Benjamin Hunt, Emily Chen, Sylvia L.R. Schreiner, and Lane Schwartz. 2019. Community lexical access for an endangered polysynthetic language: An electronic dictionary for St. Lawrence Island Yupik. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 122–26, Minneapolis, USA. Association for Computational Linguistics.

Ryan Johnson, Lene Antonsen, and Trond Trosterud. 2013. Using finite state transducers for making efficient reading comprehension dictionaries. In *Proceedings of the 19th Nordic Conference of Computational Linguistics*, pages 59–71. Linköping University Electronic Press.

Jordan Lachler, Lene Antonsen, Trond Trosterud, Sjur Moshagen, and Antti Arppe. 2018. Modeling Northern Haida verb morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan. European Language Resources Association.

William Lane and Steven Bird. 2019. Towards a robust morphological analyzer for Kunwinjku. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 1–9, Sydney, Australia. Australasian Language Technology Association.

Vladimir Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–10.

Krister Lindén, Erik Axelson, Senka Drobac, Sam Hardwick, Juha Kuokkala, Jyrki Niemi, Tommi A Pirinen, and Miikka Silfverberg. 2013. HFST: a system for creating NLP tools. In *International Workshop on Systems and Frameworks for Computational Morphology*, pages 53–71. Springer.

Patrick Littell, Aidan Pine, and Henry Davis. 2017. Waldayu and Waldayu Mobile: Modern digital dictionary interfaces for endangered languages. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 141–50, Honolulu, USA. Association for Computational Linguistics.

Olga Lovick, Christopher Cox, Miikka Silfverberg, Antti Arppe, and Mans Hulden. 2018. A computational architecture for the morphology of Upper Tanana. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*.

Anant Maheshwari, Léo Bouscarrat, and Paul Cook. 2018. Towards language technology for Mi'kmaq. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan. European Language Resources Association.

Sarah Moeller, Ghazaleh Kazeminejad, Andrew Cowell, and Mans Hulden. 2018. A neural morphological analyzer for Arapaho verbs learned from a finite state transducer. In *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*, pages 12–20.

Sjur N. Moshagen, Tommi Pirinen, and Trond Trosterud. 2013. Building an open-source development infrastructure for language technology projects. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 343–52, Oslo, Norway. Linköping University Electronic Press, Sweden.

Lane Schwartz, Francis Tyers, Lori Levin, Christo Kirov, Patrick Littell, Chi-kiu Lo, Emily Prud'hommeaux, Kenneth Steimel, Rebecca Knowles, Jeffrey Micher, et al. 2020. Neural polysynthetic language modelling. *arXiv preprint arXiv:2005.05477*.

Conor Snoek, Dorothy Thunder, Kaidi Loo, Antti Arppe, Jordan Lachler, Sjur Moshagen, and Trond Trosterud. 2014. Modeling the noun morphology of Plains Cree. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 34–42. Association for Computational Linguistics.