Abstract

1 Introduction

The Digital Tools Summit in Linguistics (DTSL) was convened in Michigan in June 2006 to foster the design and development of innovative tools for linguistic research. Six working groups, each with a designated focus area, met over the course of the two-day workshop to envision a new tool for application within its focus area and to begin the design specification for that tool. This document is the final report of DTSL Working Group 2. Our stated focus was ’Automation and Data Migration’.

Our discussions led us to two broad developments. First, we proposed a broad modular architecture for the development of interchangeable ’widgets’, each performing a particular task and perhaps tailored to a particular language, a particular domain, or a particular theoretical approach. The end result of this vision would be a framework of interoperable modular components which support the production and use of interlinear glossed text (IGT) and which plug into linguistic databases of various sorts. As long as we’re dreaming, we envision a framework supporting all levels of linguistic structure, all languages, all media, and all linguistic subfields.

This modular architecture is schematized in Figures 1 and 2. Figure 1 shows inputs and outputs to the system (labeled ‘linguistic analysis ensemble’). Figure 2 shows a first pass classification and enumeration of the modules we envision. Central to the system is a database storing the data and analyses as well as specialized, third-party data such as the General Ontology of Linguistic Description Farrar and Langendoen (2003). The rest of the widgets can be classified into four groups: Input handlers, editors for different types of data (including IGT via the annotator), analysis tools, and output generators.

This modular framework is inspired by the Unix model, where the operating system is actually a highly interoperable set of small, specialized, independent programs. Applying this notion to digital tools for linguistics should ideally lead to a scenario where individual linguists can configure their ideal working environment by selecting among multiple tools for different functions.1 To be truly successful, this framework will require broad input into the design of the modular system and broad uptake across research groups.

Rather than take on the formidable task of actually designing the modular architecture, our working group instead decided to focus on one particular widget. We propose a tool which takes raw, unanalyzed text as input and from that input gathers and displays groups of words/wordforms which it has determined are

---

1We note, however, that some linguists don’t want to do their own configuration: It should be equally possible for digital tools providers to maintain some off-the-shelf basic configurations that should be suitable for most people.
likely to be morphologically related to one another. Each group is to be organized around a particular key morpheme, and the display should present each morpheme with enough context to allow the linguist using the system to make decisions about the status of the words and forms displayed — hence the name KMIC — Key Morphemes in Context.

The tool described in this document exploits techniques from natural language processing, typological information about morphological properties, and language-specific morphological information. This widget would potentially be called by various other modules in support of a variety of tasks, assisting the linguist in developing an analysis of the morphology of a language, as well as allowing the linguist to test previously-developed hypotheses. The paradigms should also be available to be fed into a lexicon or lexicon-development system such as Shoebox or Fieldworks Language Explorer (FLEX). Finally, the output of the system may be useful for producing morphological analyzers for the language, which could in turn serve as component of NLP systems (e.g., CALL or MT) or as components of spell checkers (for community use as well as for linguists’ use in verifying the data input into the system).

2 Functionality

The primary functionality of the KMIC widget is to present the linguist with sets of words likely to contain a morpheme in common. It does this on the basis of various kinds of information, either separately or in combination: unsupervised segmentation of raw, unanalyzed text; a hand-built morphological analyzer, with possibly partial coverage of the data; a lexicon of known morphemes and allomorphs; typological information about the language in question.
3 Objectives

4 Requirements

General best practice requirements in our field apply: The tool should be open source, unicode-compatible, cross-platform, and support XML. It should be well-documented, both for users and for future developers. It should be situated within the modular framework described above, able to interact with the central database as well as other widgets. For example, the KMIC will most likely be called from within an annotator/editor.

A more specific requirement is that this tool should favor recall over precision. That is, for our envisioned use cases, it’s more useful if the system returns more forms but includes more irrelevant ones than if it finds very few (but always relevant) forms. If a linguist is looking for instances of a particular morpheme with a text frequency of .01% and the KMIC returns a set of words of which 50% contain the morpheme, the linguist is unlikely to be very hampered by the other 50%.

Furthermore, the KMIC will require efficient indexing of large quantities of data: lattices of possible morphemes representing multiple candidate segmentations of all the words in a potentially large corpus. Under certain scenarios, the addition of new data to the corpus might ideally result in just an incremental update to the stored data. In others, it might retrigger a full reanal-
ysis (since some of the analysis is done on the basis of all the data in the corpus).

5 Design

We envision the KMIC system as part of the modular framework discussed in 1. Rather than being part of a larger lexicon management system, KMIC is seen as a stand-alone component with the ability to interface with the user’s system(s) of choice. In order to be interoperable with a variety of data structures, the tool might first query the database it is plugged into in order to learn the fields and data structures being used by the database. Or it may not be plugged into a database at all but rather included as part of a workflow managed not by another software component but by the linguist herself (or himself).

The primary input to the system is raw text. The prototypical usage situation is that of the field linguist approaching morphological analysis of data which has been collected, archived, transcribed, and (possibly partially) translated. The linguist runs KMIC on the transcribed data and uses the results to aid morphological analysis. 5.1 speculates on algorithms for arriving at the desired results.

One key feature of the KMIC system as envisioned by our working group is its ability to combine and exploit multiple sources of information in arriving at the word groupings it proposes to the user. 5.2 describes in more detail these information sources and how they are to be accessed. 5.3 discusses how this information is to inform the analyses proposed by KMIC.

5.1 Data processing

To prepare to answer queries, KMIC’s first task is to segment all of the word tokens in the input data. This segmentation will be handled by a collection of separate systems, the optimal integration of which will be an interesting research question. The first type of system is unsupervised automatic segmenters, such as Creutz and Lagus (2005), Goldsmith (2001), and Monson et al. (2004). These systems generally work better with agglutinating rather than fusional languages, and with languages with one-two affixes per word rather than more complex words. Nonetheless, we anticipate that they will potentially provide interesting input, even for more complex languages. The canonical output of automatic segmenters preserves the phonological or orthographic content of the surface word and inserts morpheme boundaries. They also collect stems into sets that allow the same affixes.

The second type of system is a hand-built (or otherwise supervised) morphological analyzer. These systems typically include morphophonological (morpho-orthographic) rules, and thus produce as output forms with not only morpheme boundaries inserted but also regularized phonology. For a language with a complete (perfect) morphological analyzer, KMIC will be a less interesting tool. However, KMIC should be able to take advantage of partial morphological analyzers such as morphological analyzers under development or perhaps morphological analyzers built for related languages.

A third source of information for segmentation is a lexicon for the language, that is, a list of morphemes that have been isolated by the linguist already. When used in isolation, this data source will limit the search domain to those morphemes that have been included in the lexicon already. There are many ways a (partial) lexicon could be used to improve automated segmentation or morphological analysis, but this depends on those components be-

---

2 It is not entirely clear how the translation fits into the picture. Does KMIC see only the transcribed text, or does KMIC also have the capacity to draw in information from the translation in performing its analysis? We suspect the former, as the latter remains the work of the human.

3 We assume that the sentences/utterances are already segmented into words.

4 How much data do they require to start getting interesting results?
ing built to accept additional lexical information ‘on the fly’. Short of that, a lexicon could be used as a filter on the output from other systems, preferring, for example, candidate parses which include known morphemes.

In addition to tools which handle morphology directly, KMIC might also incorporate information from other tools, such as a part of speech tagger, a named entity recognizer, or a bilingual lexicon, depending on what exists for the language in question or what can easily be ported from a related language.

The final potential source of information is typological properties of the language, as identified by the linguist. For example, the unsupervised segmenters should be parameterizable based on whether a language is primarily prefixing or suffixing. Likewise, one could imagine a parameter which ‘tuned’ them to expect relatively clean concatenation of morphemes or relatively complex morphophonology, potentially obscuring morpheme boundaries.

KMIC should not only be able to intelligently combine information from these different source, but also allow the linguist to select which ones are ‘active’ for any given search. On the one hand, this could be useful as an experimental set-up for comparing the performance of various automatic segmentation algorithms under different conditions (different language types, different amounts of information, different corpus size, etc). On the other hand, for different tasks (see below) users might find different inputs to produce more useful results.

Even with a single system doing the analysis, there is often (legitimate) ambiguity in morphological parsing. Thus once KMIC has combined the information from its various segmentation systems, it should maintain not a single parse for each word, but rather a weighted lattice of possible segmentations. With a set of such lattices (one for each word in the input text, minimally just the unsegmented word in cases of parse failure), KMIC then groups the words into sets that share a morph (morpheme form) in common. Any given word will belong to as many groups as there are morphs in its lattice. Its membership in those groups will be associated with a confidence value, according to KMIC’s confidence in the particular segmentation represented. This can be used for sorting the results to queries, to put the best information towards the top.

Both the segmentation process and the search process may be affected by decisions resulting from the application of filters as described in 5.2 and 5.3. A simple English example is given below. The raw text is shown in 1, and a likely segmentation in 2.

(1) John swims happily.

(2) John swim+s happi+ly.

Depending on the needs of the linguist, we can imagine a number of different types of queries. In other words, it will be helpful to provide KMIC with a starting point, particularly for the purposes of displaying the results. Imagine if the system were to simply spit out all of the groupings it finds—very little would be gained. The examples below illustrate some of the types of queries a linguist might want to perform. Our linguist, who we’ll call Joe, is working on an exotic language called English.

- **suspected morpheme**: Joe suspects that -ly may be a productive morpheme in his language, so he asks KMIC to locate all words in the text containing that morpheme.

- **position in word**: Joe’s language seems to favor suffixes for inflectional morphology. Joe asks KMIC to look for groupings centered around word-final morphemes. In this case, the system might return a grouping containing swim+s, run+s, play+s, cat+s, mitten+s, think+s, dog+s, ... This example highlights the need for the linguist’s interpretation of KMIC’s results, as
we see word-final -s appearing on nouns, verbs, and words ambiguous between the two. On the other hand, Joe may not yet have discovered the polysemy of the morpheme in question, and seeing the morpheme appearing on words of different categories may help him improve his analysis.

- **number of morphemes in word:** Joe may be interested in studying morpheme interaction in his (let’s pretend for the sake of the example) highly-inflecting language. The ability to search by number of morphemes per word facilitates the location of possibly relevant examples without painstaking manual search.

- **frequency:** if Joe is just beginning analysis in his language, he might find it fruitful to start by looking at those morphemes which appear most frequently in the data he has collected. We could see this as an empirically-based way of getting an early foothold for morphological analysis.

- **singletons (hapax legomena):** looking at singleton morphemes may be useful for identifying roots in the language. In addition, singleton morphemes may indicate errors either in analysis or in data entry.\(^5\)

- **shared signature:** as another means of identifying roots, Joe might be interested in morphemes that show up in combination with the same set of other morphemes as each other, or as some seed example he is working with. For example, if Joe knows that sing is a verb, he may wish to find other verbs (of the same inflectional paradigm) by searching for other morphemes that can appear before the same suffixes (or before nearly the same suffixes, to allow for some data sparsity in the corpus). (Ref Goldsmith on this idea?)

- ‘**I’m feeling lucky**’ (with apologies to Google): perhaps Joe has done a lot of analysis but is feeling at a loss for where to go next. In this case KMIC selects a grouping for which it has a relatively high confidence level.

Once it has processed the data in accordance with the user’s query, KMIC returns one or more groupings of words. An attractive possibility is for KMIC to return not only the word groups but also a range of supporting information. For example, assuming proper interaction between KMIC, the central database and the annotator, each word in the group should be linked back to its occurrence(s) in the text, allowing user to identify relevant examples and perhaps mark them in some way for later use. The word groups might also be linked to the filters and other heuristic factors which led to the grouping decision. In this way, the linguist could examine the rules or characteristics identified by the KMIC system for elements which may assist in further analysis of the language at hand, and also track the performace of KMIC under various input conditions. Another possibility is for the system to link its analysis to an instance of the text within a lexicon management system (e.g. Shoebox) in such a way that the linguist can select segmentations and analyses to be ported directly back into the lexicon. Finally, the output from KMIC may be useful for producing morphological analyzers and spell-checkers which may then be incorporated into the workflow to aid in the analysis of future texts. Automatic morphological analyzers would also be very useful for faster and more efficient production of interlinear glossed texts.

We now turn to the filters which harness language-specific and/or typological characteristics in order to customize KMIC’s search process.

\(^5\)It will be important for Joe to have some way of tracing individual examples within the groupings back to the original input data.
5.2 Filters

The KMIC tool as envisioned is not language-specific but rather can be ’tuned’ to a language through incorporation of typological information and selection of appropriate paradigmizers. With this we envision the possibility of creating many different types of ’paradigmizers’, each tuned to particular contexts or linguistic properties. For example, paradigmizers for concatenative morphology would seek out very different types of patterns than would paradigmizers for Semitic root-pattern morphology. This section describes a number of possible filters which may be used by the system as heuristic aids to selecting groups of related forms.

At outset of a project, the linguist completes a brief typological questionnaire. The information from the questionnaire will be used as to guide selection of appropriate filters on the data and selection of appropriate paradigmizers. The system selects filters and paradigmizers likely to favor analyses which fit the questionnaire responses over those which contradict the characteristics indicated by the questionnaire responses. The linguist should be able to return to the questionnaire mid-project and update values, should his/her hypotheses about the language change.

These filters may be of two different sorts. The first group of filters would be based on known language-specific characteristics, with these characteristics being indicated by the linguist in the questionnaire. Some examples of possible characteristics follow.

- this language seems to favor prefixes (or suffixes)
- this language uses both prefixes and suffixes
- this language generally allows a maximum of three morphemes per word
- this language uses infixes
- this language has reduplication
- this language has relatively clear morpheme boundaries
- this language has extensive morphophonological processes
- ... 

The type of filters described above belong to the most specific of three layers of filters. The second and third sets of filters involve more general properties of languages, the former suggested by KMIC according to the typological characteristics of the language, and the latter suggested based on notions about languages in general.6 We envision the possibility of KMIC consulting a knowledge base of general morphological tendencies, but as yet we have not worked this out in any greater detail. It’s worth pointing out that, at all three levels, we see these filters not as rules to be applied absolutely, but rather as ’squishy’ properties which may help the system to select the most likely analysis.

5.3 Using the information

[THIS SECTION IS A LOT OF MY WONDERINGS ABOUT THE ADVANTAGES AND DISADVANTAGES OF VARIOUS APPROACHES... I’M NOT CERTAIN IT’S ALL RELEVANT FOR THIS DOCUMENT]

Once KMIC has selected appropriate filters and paradigmizers, where and how are these filters to be applied? This remains an open question. Applying the filters during the segmentation process would certainly improve the quality of the segmentations and consequently the quality of the KMIC groupings, but this approach would also be computationally expensive. If the filters are instead applied during the search process, we reduce the computational

---

6For example, it might be useful if the system could incorporate knowledge about common phonological processes, such as palatalization following high front vowels. Making this usable would require mapping between orthographic representations and phonological features, however.
Another open question is that of how to integrate the results of multiple searches, performed using multiple filters. We would like the user to be able to access the filters which influenced KMIC’s decisions for a particular text, but the extent to which we will have access to that information will depend greatly on the approach used to get the results. For example, a straight statistically-based machine learning approach will not be able to report on the motives for its decisions, but a rule-based heuristic approach will be able to report which rules resulted in the results it ultimately produces. It might be most fruitful to employ a hybrid approach, using both statistical machine learning and rule-based heuristics.

We may find a multi-round search process to produce the best results, perhaps first employing the filters most specific to the language and then filtering those results using the more general second and third tiers of filters. Or KMIC may be most usefully employed as part of a more complex workflow, as imagined below.

Perhaps Joe knows something about the morphology of his language; let’s say he has a list of inflectional morphemes and knows a few simple morphotactic rules. Using utilities tied to KMIC, he could first produce a morphological analyzer for the known rules and morphemes. The text could be fed into this morphological analyzer, and the words recognized by the analyzer would receive their segmentations in this way. The unrecognized words could be fed into our automatic segmentation component, and the search process would then apply to a text in which some word forms have been segmented by the analyzer and others have not. This might be one way to reduce the influence of wrong segmentations on the results.

Depending on the manner in which the filters are used, KMIC might produce stem-based paradigms or paradigms organized around a particular inflectional or derivational morpheme. This example also raises the issue of interactive iterative searching. For example an initial stem-based query may produce the grouping shown in 3. Based on this result, the user may query on a particular morpheme (e.g. word-final -s) and get back the results shown in 4. Or perhaps the user could employ a wildcard to represent the stem and search for robust inflectional patterns, as shown in 5.

(3) swim, swim+s, swim+ming, swim+mer, swim+wear, ...

(4) swim+s, run+s, play+s, jog+s, think+s, hat+s, ...

(5) (swim, swim+s), (run, run+s), (play, play+s), ...

6 Implementation
7 Evaluation
8 Use Cases

Typical users: field linguist, native speaker, end users, archivist

Types of end users: language community, L2 teachers and learners, computational linguists, theoretical linguists

Possible uses (by user)

Field linguist: help finding/predicting errors, help confirming analyses (via feedback from system on consistency of analysis, possible errors, proposed likely analyses) Native speaker or community member: standardization of orthography, updates to Swaddish [sp?] wordlist, creating of language learning or teaching tools

Theoretical linguist: verify (or discredit) proposed rules or analyses by performing hypothesis testing on annotated corpus
References


