Input Methods—Machine Learning and HCI Perspectives

David R. Mortensen

March 28, 2024

The Importance of Input Methods

An essential ingredient in using language with technology is getting the language into the technology. Technology to do this is called an INPUT METHOD EDITOR or just an INPUT METHOD. One of the challenges in creating a level playing field for languages of the world is provided equitable ways of entering them into computers, tablets, and phones. While this is often overlooked, it is technology that is used by most people in the worlds most days. It is worthwhile to think about it and think about how it can be made better.

This handout about input methods draws heavily on two reports from Google[1].

[1] Daan van Esch, Elnaz Sarbar, Tamar Lucassen, Jeremy O'Brien, Theresa Breiner, Manasa Prasad, Evan Crew, Chieu Nguyen, and Françoise Beaufays. Writing across the world's languages: Deep internationalization for gboard, the google keyboard, 2019; and Andreas Kabel, Keith Hall, Tom Ouyang, David Rybach, Daan van Esch, and Françoise Beaufays. Handling compounding in mobile keyboard input, 2022

Input Methods for English

Input methods have existed from English since the dawn of computing. Originally, these took the form of physical keyboards modeled on typewriters. With the emergence of mobile technology, other forms of input (usable on the keypads of phones or the surfaces of tablets) became necessary. Now, English is most often input via smartphones with predictive text entry (autocomplete, next word prediction, and spelling correction).

Physical versus Soft Keyboards and Layouts

Physical keyboards in the English speaking world typically have had a QWERTY layout (a name that derives from the first six letters on the top letter row (starting from left to right). The include just the letters needed to type English, as well as various kinds of punctuation. There is no in-build means for typing characters with diacritics, etc. However, input methods have been defined that allow users to type additional characters by, for example, typing a sequence of keys or, as is the case with the default input methods on MacOS, holding down a key until a menu of options appears, then selecting that option with a number key.

Physical keyboards contrast with soft keyboards, which are represented on a touch screen.

Old-Style Predictive Text

Early mobile phones would not accommodate a typewriter-like keyboard. Instead, text had to be entered with the number pad. Each number corresponded to a small set of letters and, with sequences of taps, it was possible to enter words.

Multitap An input method where the user taps the number key until the desired symbol appears.

T9 An input method in which the user types a sequence of number keys, corresponding to the first few letters of the desired word, and is then presented with options. The user iterates through these options using the "next" key.

Modern Predictive Text: Auto-Complete

However, we've moved beyond phones with 3×4 key pads. Modern mobile phones use soft keybards, meaning—among other things—that the same device can be used, in equal footing, with different scripts and languages. As noted in [2], taking input from a soft keyboard involves a kind of decoding—finding the word sequence $w*$ that best matches the sequence $\mathbf{x}$ of coordinates of taps.

[2] Andreas Kabel, Keith Hall, Tom Ouyang, David Rybach, Daan van Esch, and Françoise Beaufays. Handling compounding in mobile keyboard input, 2022

Decoding, with such keyboards is typically associated with three interrelated functions:

- Word completion
- Next word prediction
- Spelling correction

As we'll see, these are implemented with small, on-device language models.

Types of Input

Now, let's look in more detail at types of input.

Physical Keyboards

Physical keyboards have a fixed layout. These vary with language, though many layouts are influenced by the American QWERTY layout. Typically, there is a one-to-one mapping from keys to letters (although this isn't the case for all languages and keyboards). For example, keyboards may have DEAD KEYS. A dead key for an acute accent may be pressed before ⟨a⟩ to produce ⟨á⟩. A keyboard may also have modifier keys (like the shift keys on English QWERTY keyboards that change the case of a letter pressed at the same time). The function keys like Control, Option, Command, Alt, AltGr and Meta are also modifier keys, and they sometimes change the letter that is produced when they are depressed with another key.

Soft Keyboards

Soft keyboards free text input from fixed layouts. They offer the following advantages over physical keyboards:

- They can fit on a mobile device
- They are easily adaptable to new languages
- They allow for more rows/columns of keys that a physical keyboard
- Layouts can be dynamic—tapping one key can "switch" the display to a different layout (as with the shift or symbol keys in some input methods).
- Layouts can be personalized on a per-user basis

Soft keyboard input methods are typically reliant, for their completion, prediction, and correction functionalities, on a small n-gram language model. This typically has to be limited to 1–10 megabytes (so it is tiny).

Handwriting Recognition

You can also enter text using handwriting recognition (on a touch pad or screen). This is especially suitable for writing systems in which the number of symbols is far greater than can be displayed on a screen at once.

Speech Recognition

Finally, ASR is a popular way of inputting text for SMS messages and notes. It is script agnostic and is helpful for people whose disabilities make it difficult to type on a touch screen or physical keyboard.

Machine Learning Factors

Input methods tend to rely on very old technology

- N-gram language models
- 3-grams
- 128K words
- A few million n-grams

Why in the ear of large language models are input methods made with such small language models?

- Limited processing power

- Limited storage space

Can you think of ways to improve the performance of IM LMs which still satisfying these constraints?

The training data for IM LMs has to be carefully curated. Why?

Decolonizing Input Methods

Input methods as we know them know are highly colonial—they reflect European, and especially America, hegemony over the linguistic and technological worlds. This presents problems for people who want to use their language to interact with technology.

In Kanuri, a language spoken around Lake Chad in North Central Africa, there is a letter ⟨ə⟩. This does not exist on the soft keyboards widely available to Kanuri users, so they have to substitute ⟨3⟩ for it.

On the other hand, there are good input methods for Chinese, Japanese, Korean, and some languages written with Cyrillic scripts, but many other languages are not well represented. For example, even though Yakut (a Turkic language of Siberia) is written with the Cyrillic script, in uses certain letters that don't appear in most other Cyrillic scripts, including Russian.

Even from a language with as many speakers of Arabic, they have not been good input methods available on mobile phones and computers. For Arabic, an improvised "online orthography" called Arabizi has developed. Likewise, speakers of Hindi have had to use a Latin-script online orthography.

Users of these online orthographies are often reluctant to abandon them, even if good input methods for their languages become available. Why might this be?

The Internal Logic of Non-Roman Scripts

Non-roman scripts often have an internal organization that does not match a QWERTY keyboard well. For example, Devanagari scripts are organized around place and manner of articulation and it might make sense to organize a soft keyboard according to the same principles. However, many Devanagari keyboards are now organized around the "phonetic" associations between Devanagari symbols and Roman letters.

Morphological Complexity

What about morphological complexity? In languages with complex morphology, including languages with very productive compounding, the size of the vocabulary is very large. However, the size of the LM must remain very small. As a result, tokenizing on white space is inadequate. One solution: use morphological tokenization using FSTs.[3]

Why would we do this? OOV compounds. When we type them, one of three bad things will happen:

[3] Andreas Kabel, Keith Hall, Tom Ouyang, David Rybach, Daan van Esch, and Françoise Beaufays. Handling compounding in mobile keyboard input, 2022

- They will be literally decoded as misspellings
- They will be split into components
- They will be auto-corrected into another word

Solution: build a model of how compounds can combine and apply it in decoding.

References

Andreas Kabel, Keith Hall, Tom Ouyang, David Rybach, Daan van Esch, and Françoise Beaufays. Handling compounding in mobile keyboard input, 2022.

Daan van Esch, Elnaz Sarbar, Tamar Lucassen, Jeremy O'Brien, Theresa Breiner, Manasa Prasad, Evan Crew, Chieu Nguyen, and Françoise Beaufays. Writing across the world's languages: Deep internationalization for gboard, the google keyboard, 2019.