October 2021

Providing Suggestions of Expanded Text from Abbreviated Text Input

Ajit Narayanan

Meredith Morris

Subhashini Venugopalan

Michael Terry

Sherry Tongshuang Wu

See next page for additional authors

Inventor(s)
Ajit Narayanan, Meredith Morris, Subhashini Venugopalan, Michael Terry, Sherry Tongshuang Wu, Shanqing Cai, and Carrie Cai

Providing Suggestions of Expanded Text from Abbreviated Text Input

ABSTRACT

This disclosure describes techniques to provide suggestions of expanded text from abbreviated or compressed text that has been input by a user. A language model is used to determine and present the most likely full words and phrases that match user intent based on the user's abbreviated text input, such as the first letter of each word of a phrase and/or omission of one or more words of the phrase. The described techniques can greatly improve speed of text entry to devices via a keyboard or other input modality.

KEYWORDS

- Text prediction
- Text completion
- Text suggestion
- Autocomplete
- Virtual keyboard
- Abbreviation expansion
- Accelerated text entry
- Language model

BACKGROUND

Text entry can benefit from word suggestions or predictions (also referred to as text completion). Many software-based input methods such as those on mobile devices with onscreen keyboards have word suggestion systems built-in. Language models take user typed context as input to make suggestions for completions of input text or subsequent words and phrases. Keyboard word suggestion systems can provide next word predictions, word auto-correction, word completion, next sentence response, next-word or next-phrase prediction, and next sentence prediction. These systems can significantly improve typing speed. While existing word suggestion methods are capable of improving text entry speed, they are based on full words that have been typed by the user, and expect the user to enter characters to form full words and/or complete phrases. This imposes an upper limit on the rate improvement in typed input that can be achieved.

DESCRIPTION

This disclosure describes techniques for presenting suggested expanded text (words and phrases) to a user based on input text in abbreviated form (e.g., compressed form). Instead of typing characters to compose full words, a user can type abbreviations of entire phrases or sentences. For example, the abbreviated text input can include the first character or multiple characters of each word in a phrase, a subset of the words of a complete phrase, a partial mix of characters and words, and/or variants thereof. Described techniques generate multiple suggestions that each expands the abbreviated input into a full phrase possibly intended by the user. Large language models (LLMs) that are tuned on conversations can be used to generate suggestions.

By inputting abbreviated text input and selecting from relevant expanded text suggestions for that input, a user can significantly increase the speed of typed input compared to inputting full words and/or complete phrases. Typing in abbreviations and other compressed input can provide dramatic savings in keystrokes, which can benefit the speed of text entry from keyboard interfaces and other input devices such as augmentative and alternative communication (AAC) devices for users with disabilities. For example, the speed of communication is often determined by the rate of text entry. Described techniques allow communication at a rate fast enough for natural, real-time conversation by allowing the user to input a highly compressed form of text input to a device that intelligently expands the input into a complete phrase or sentence that can be used in a communication process such as a chat conversation or presentation, or in other processes.

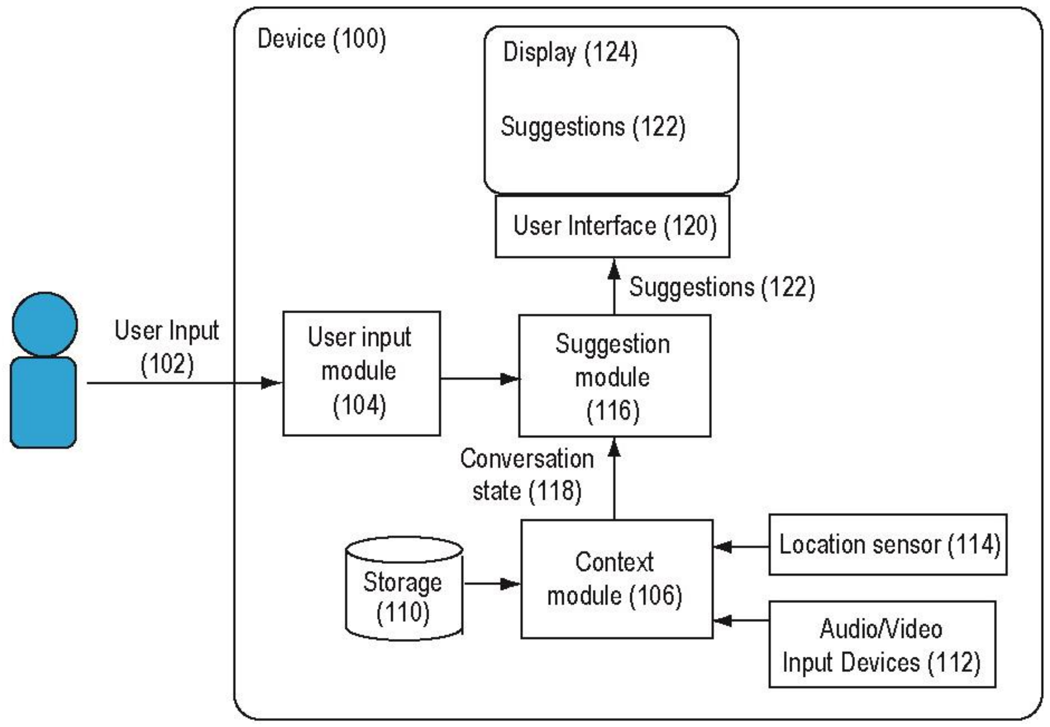

Fig. 1: System for providing expanded text suggestions for abbreviated user input

Fig. 1 shows an example of a system to perform described techniques to provide expanded text suggestions for abbreviated user text input. The system can be implemented on a device (100) such as a desktop or laptop computer, portable user device (e.g., a smartphone), server device, etc. The user is provided with options to enable or disable described techniques for determining text suggestions, and can permit specific types of data to be utilized and can deny use of other types of data.

User input module (104) receives input (102) from a user and transforms the input to an equivalent of text input provided by keystrokes (or key selections) on a keyboard. In some examples, the input may be provided by the user during a conversation with one or more other users, e.g., a text chat via user devices, or the input can be provided in other contexts. Input can be provided by the user via one or more different input modalities, such as physical keyboard, input device such as mouse, trackball, trackpad, etc., touchscreen, a microphone capturing voice or speech input, systems controlled by eye-gaze, facial gestures, smart glasses, wearable computers, brain computer interfaces, etc. These systems can be used in general-purpose computing devices and/or used to provide accessibility for people with disabilities.

The user input module receives an abbreviated (compressed) form of input that is incomplete, e.g., has omitted characters and/or words from the complete text that the user would like to communicate. In some examples, the input characters can be only the first character of each word of the complete text. In some cases, the input characters can be multiple characters in one or more words, e.g., consecutive or separated characters in the word (e.g., “child” abbreviated as “ch”, or “alaska” abbreviated as “ak”). In further examples, the abbreviated input can omit one or more entire words from a phrase or sentence (e.g., “No, the weather is too gloomy” abbreviated as “weather gloomy”). Some abbreviated input may omit punctuation, and such punctuation can be included in complete words or sentences suggested by the system. Some abbreviated input can omit capitalized letters. For example, some systems can interpret a capital letter in the input to indicate capitalization of a suggested completed word.

Furthermore, the input may include some words that have omitted characters and other words that are complete. For example, the user input may be “od,irhtvstonehenge”, where the first eight characters indicate “one day, I really hope to visit” and the word “stonehenge” is retained in the form it is received. This can be useful since the system may not be able to guess the word “stonehenge” from only a first letter “s” due to the numerous other places that could be represented by that letter.

Some systems can allow abbreviations in user input provided using a reduced alphabet. For example, a single symbol can represent a group of letters or characters. In some examples, characters or groups of multiple characters that sound similar to each other can be represented by a single symbol, such as letters “p” and “b” represented with “p”, e.g., the abbreviation can be an approximate phonetic abbreviation. This may reduce the time to input such characters and/or reduce the error rate per keystroke. Alternatively, an expanded or larger alphabet than the standard alphabet can be used. In some examples, special characters can represent “ch” or “sh,” or particular parts of speech (e.g., nouns, verbs, etc.) can be indicated via capitalization. In further examples, a number can indicate the number of characters following an abbreviated letter, e.g., “s8” can indicate that eight characters follow the letter “s” in the completed word.

A related feature can make optimal use of keyboards by transliterating an input sentence into a different script or different language and can use abbreviations in the transliterated language. For example, to abbreviate text in Chinese or Indic languages, the first letter can be selected from each transliterated English word.

Context Module

With user consent, a context module (106) gathers various elements of user context and conversation context that can be used to predict expanded text from the user’s keystrokes. The context module outputs conversation state data (108) that indicates various determined context elements for the received abbreviated text input.

For example, the context elements can include previous user input such as messages or other content that has been input by users to user devices in a current or ongoing conversation. For example, a previous message of “What kind of dog do you have?” in the conversation allows a following abbreviated input of “ihagr” to be more readily predicted to be “I have a golden retriever” based on the context or subject indicated by the previous message. With user permission, context elements can also include messages from outside the conversation, e.g., the user’s previous conversations, past emails, SMS messages, texts, etc. For example, with user permission, previous messages can be stored in and retrieved from data storage (110).

Context elements can also include, with user permission, audio and video signals obtained from audio/video input devices (112) such as cameras, microphones, etc. that capture voice and images of users participating in the conversation. Voice input can be recognized using automatic speech recognition. Context elements can also include, with user permission, identities of users in a conversation, e.g., detected from captured speech or images. Context elements can also include, with user permission, a current time and/or times of previous user input, current and past geographic locations of the user devices based on data from a location sensor (114) such as a GPS sensor, and any other available signals that can help the system predict a user’s intended message and narrow down the possibilities for expanded text suggestions.

Some systems can obtain context elements that indicate a style of communication of the user or users in the conversation. For example, with user permission, a corpus of previous user conversations and emails can be examined to infer a lexicon of words that the user is likely to use more frequently than typical users. Certain collocations of words or phrasal patterns that are idiosyncratic to the user can also be determined. In some systems, a generalized style can be indicated instead of or in addition to a specific style of the user, e.g., by referencing proxy corpora. For example, an anonymized corpus of English as spoken by people in the area or country in which the user device is currently located (determined with user permission) can be used to determine a certain usage of language; or an anonymized corpus of emails between software engineers can be similarly used for a user who is a software engineer. In some systems, a formality of the conversation can be used as a context element, where the intended complete text may be different based on the formality. Formality can be based on the communication medium of the conversation, e.g., recognized speech and/or email may be more formal, and text messages are more likely to be less formal and include emojis. Formality can also be based on a recipient of a user message, e.g., a supervisor or a friend.

Suggestion Module

A suggestion module (116) receives abbreviated text input from the user input module and also receives the conversation state (118) from the context module. From these inputs, the suggestion module determines suggestions of expanded text that the user is intending to communicate via the abbreviated text input. In some systems, the suggestions can be determined using a language model, e.g., a machine-learning model. For example, a large language model (LLM), e.g., a neural conversational model, can be used to determine suggestions. The LLM is a deep neural network that can have over a hundred billion parameters, trained over billions or trillions of words taken from diverse sources such as dialogs, newspaper articles, books, and web documents. With user permission, training can include context elements from various users, such as example context elements described above. Through training, the LLM develops an emergent property of being able to process contents and meaning of language and related knowledge, which it can apply to expanded text suggestions tasks with high accuracy and sensibility.

To use a LLM to provide expansions for abbreviated input, a technique called prompting can be used to query for text suggestions from the language model. Prompting is a mechanism to direct an LLM to output meaningful suggestions for a particular task, based on either the previous training and no further examples (zero-shot learning) or based on the previous training and a small number of examples (e.g., two to five examples, called few-shot learning). In both cases, the prompts provided to the LLM take the form of natural language.

The suggestion module can operate as follows. First, based on the type of the abbreviation expansion task, a prompt is constructed that includes a description of the task in natural language followed by a small number of examples of the particular abbreviation-expansion task, and finally a given context and the abbreviated text. A temperature parameter is specified to control the randomness of the responses of the LLM. Other parameters of the sampling process include the maximum length of the text output and the number of highest probability (top-K) tokens that are considered at each sampling step. The sampling is based on an autoregressive sampling process in which the output of each sampling step is added to the input for the next step.

Each sampling step is based on the LLM’s output logits over a vocabulary. This is made possible due to the task in which the LLM is trained, namely predicting the next token of text based on preceding text. A configurable number of outputs can be sampled from the LLM for a given abbreviation input (e.g., between 128 and 4096). Drawing multiple samples increases the likelihood of obtaining the desired expanded text from an abbreviation and increases the diversity of options for the user to choose from. Some examples of prompts are shown below.

- **To direct reconstruction from abbreviations of each word.**

Context:	{Do you want to walk around the park?}
Abbreviation:	{n,twitg}
Full:	{No, the weather is too gloomy.}

- **To direct reconstruction from full keywords.**

Context:	{Do you want to walk around the park?}
Abbreviation:	{weather gloomy}
Full:	{No, the weather is too gloomy.}

- **To direct reconstruction from abbreviations of words mixed with keywords.**

Context:	{Do you want to walk around the park?}
Shorthand:	{n,twit gloomy}
Full:	{No, the weather is too gloomy.}

Several techniques have been proposed for “prompt tuning” for an LLM to produce accurate results, e.g., using a “soft prompt” that is learned by back propagation. These techniques can be used to make the suggestion module more robust.

Some systems can filter the responses generated by the LLM to ensure that the suggestions presented to the user correspond to the input abbreviated text and that non-compliant results are rejected. For example, filtering can ensure that only expansions that contract to an input abbreviation are presented while expansions that do not contract to the abbreviation are discarded. Filtering may be more accurate if the user provides more information to constrain the output text. For example, as described above, in some systems the user may type a number after a letter to identify the number of characters following a character (“s8” for “something”), which can act as a powerful constraint for filtering even if specified for one or two words in the input.

In addition to few- or zero-shot learning, the LLM can be fine-tuned by using a labeled abbreviation-full-phrase dataset to enhance its accuracy at the task. Such datasets can be obtained from a publicly available conversation dataset or, if user consent is obtained, from the user's own historical conversation records.

Data for tuning the LLM system can be generated for both abbreviations (omitted characters in a word) and elision (omitted words from a phrase), where such data can be synthetically generated from a corpus typical of the user's communication. Abbreviation involves only extracting the first letter of each word. Elision can be performed by removing high-frequency or core words from the sentence, or removing words from a fixed list of stop-words. Combined with prompt-tuning techniques, this can be used to create a representative set of examples from which a high-accuracy system can be created.

Since the LLM is sampled at the decoder stage, the probability (or perplexity) of the suggestion that the LLM is proposing can be estimated. This can be used, for example, to rank suggestions in the order of frequency, so that high-frequency suggestions are presented to the user earlier than other suggestions, saving a user's time and cognitive effort.

Some systems can include error correction and/or error resiliency. Due to the extreme nature of compression of the input, a single keystroke error in specifying abbreviated text could result in a major discrepancy between the suggestion and the correct expanded text. For example, if the user makes a mistake when abbreviating the full text of "when I was a child, I lived in Alaska" by typing "wiwac,idia" (abbreviating "lived" as "d" instead of "l"), the most likely suggestions that result could be totally different than the desired text. In addition, the user may make mistakes when transforming an intended phrase into abbreviated text.

Resiliency to errors can be built into the system. In one technique, if it is determined that the most likely output sentence given the input string of characters has a low probability, the system can suggest expansions for a more restricted portion, such as a prefix, of the text input instead. This allows the LLM to process the text input iteratively, e.g., multiple LLM steps can be chained to achieve multi-step processing. In the example above, only the first portion "wiwac" can be processed and suggestions provided by the system, and the user can be prompted to enter the rest of the text input. Second, in the case of systematic errors (e.g., wrongly assuming the word 'genius' starts with 'j'), suggestions can be made internally for alternatives that replace these characters with likely corrections, and the system can suggest the most probable predictions from those substitutions with a message similar to "Did you mean…"

Other models can also be used. For example, n-gram language models, Recurrent Neural Network (RNN) based language models (e.g., Long Short Term Memory (LSTM) RNN based language models), or new Transformer based language models can be used. Simpler character-based language models can also be used. Decoding of abbreviations can be performed by a finite-state transducer (FST).

User Interface

A user interface module (120) receives suggestions (122) of expanded text from the suggestion module and presents a user interface on a display screen (124) or other output component of the device to receive input and display the suggestions.

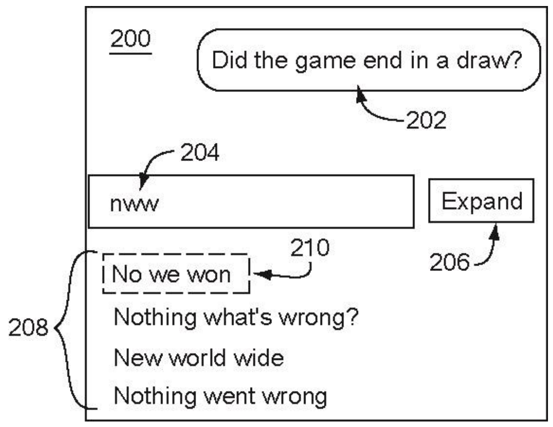

Fig. 2: Example user interface for providing expanded text suggestions

One example of a user interface (200) that provides suggestions to expand abbreviated input text is shown in Fig. 2. In this example, a first user is in an ongoing chat conversation with a second user who has input a previous message (202) in the conversation. The first user inputs abbreviated text (204) in an input field using an onscreen keyboard, and also selects an expand button (206) to request suggestions from the device that expand the abbreviated text. The abbreviated text and previous message (as context) are sent to the suggestion module on the device, which determines a set of suggestions (208). The first user selects the first suggestion (210), which is then output by the device in the chat conversation.

A variation of the user interface omits the expand button, and may execute the suggestion system every time the user inputs a character, thus displaying various relevant suggestions dynamically. This lets the user know when the system starts diverging from their desired output, at which point the user can pick a prefix match, or backtrack (e.g., delete one or more characters of the input) to cause a suggestion to be displayed that is a closer match to the desired expansion.

Some systems can provide additional features. For example, for some words, the expansion may be so variable that the system cannot reasonably guess what the user wants to say. In this case, the system can make a partial suggestion. In one example, after a previous message (context) of “What do you want to do today?”, the abbreviation “iwltgttm” is input by the user. The system determines the prefix “I would like to go” for the portion “iwltg” of the input (or “I would like to go to the” for portion “iwltgtt”), and can suggest a few different expansions for the letter m (e.g., “museum,” “movies,” or “mall”). If none of the suggestions for “m” are desired, the user can select the prefix and then input the abbreviated text again, or type out the remaining full word(s) manually. This can be implemented by displaying matches against a prefix instead of strictly filtering only for matches against the entire abbreviated text. A variation of this feature can allow the user to specify a number after selecting a suggestion, which indicates the number of contiguous words of the suggestion to select. For example, if the user intends to say “I would like to go to Tijuana, Mexico," the user can select a suggestion that matches a portion of the intended words (“I would like to go to the museum”) and then type “6”, such that the first six words of the suggestion are selected for output.

Other variations can improve input speed. For example, once the user has selected the expand button, the user can type additional characters which are fed to a filtering algorithm as potential second letters of words. In the example above, after the user inputs “iwltgttm”, the system can enter an extra-filtering mode. If the user intended the letter “m” to be the word “metro”, the user could now type the letter “e.” The system then does further sampling of the LLM, and filters for matches where the letter “e” matches the second letter of any word. Thus, if the user types the letter “o” after expansion, the system could make suggestions like “I would like to go to the *m**o**vies*” but also suggestions like “I would *l**o**ve* to go to the museum.” Each additional input decreases the entropy of the system, allowing users to constrain the suggestion algorithm further.

In this manner, significant keystroke gains can be achieved, while ensuring that the accuracy of the system is high enough, and the user’s cognitive burden low enough, to be useful to the user. The described techniques can be utilized to improve text entry in various operating systems, applications (e.g., messaging, email, word processing, etc.), and virtual keyboards. A cloud application programming interface (API) that incorporates the described text expansion and entry services can also be provided.

Users are provided with options to grant permissions to and/or to disable described features entirely. The various features of the system are implemented only with user permission to access user information that serves as input to the system (e.g., user messages, camera and microphone input, a user's location, user context information, a user's preferences, etc.). Users may be provided with controls allowing the user to make an election as to both if and when systems, programs or features described herein may enable collection of user information, and if the user is sent content or communications from a server. Certain techniques are not implemented if users deny permission. In addition, certain data may be treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user. Thus, the user may have control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

This disclosure describes techniques to provide suggestions of expanded text from abbreviated or compressed text that has been input by a user. A language model is used to determine and present the most likely full words and phrases that match user intent based on the user's abbreviated text input, such as the first letter of each word of a phrase and/or omission of one or more words of the phrase. The described techniques can greatly improve speed of text entry to devices via a keyboard or other input modality, and can thereby improve the rate of communications to enable natural, real-time conversation in text conversations.

REFERENCES

1. Hard, Andrew, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. "Federated learning for mobile keyboard prediction." *arXiv preprint arXiv:1811.03604* (2018).
2. Adhikary, Jiban, Jamie Berger, and Keith Vertanen. "Accelerating Text Communication via Abbreviated Sentence Input." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6574-6588. 2021.
3. Adiwardana, Daniel, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang et al. "Towards a human-like open-domain chatbot." *arXiv preprint arXiv:2001.09977* (2020).
4. Collins, Eli and Zoubin Ghahramani, "LaMDA: our breakthrough conversation technology available online at" https://blog.google/technology/ai/lamda/
5. Lester, Brian, Rami Al-Rfou, and Noah Constant. "The power of scale for parameter-efficient prompt tuning." *arXiv preprint arXiv:2104.08691* (2021).