IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF ILLINOIS
EASTERN DIVISION

BUFFALO PATENTS, LLC, **Plaintiff,** **v.** **MOTOROLA MOBILITY LLC,** **Defendant.**	**Case No.: 1:22-cv-04540** **JURY TRIAL DEMANDED**

ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Buffalo Patents, LLC ("Buffalo Patents" or "Plaintiff") files this original complaint against Defendant Motorola Mobility LLC ("Motorola" or "Defendant") alleging, based on its own knowledge as to itself and its own actions and based on information and belief as to all other matters, as follows:

PARTIES

1. Buffalo Patents is a limited liability company formed under the laws of the State of Texas, with its principal place of business at 1200 Silver Hill Dr., Austin, Texas, 78746.

2. Defendant Motorola Mobility LLC is a limited liability company organized and existing under the laws of the State of Delaware and registered to do business in Illinois. Motorola may be served through its registered agent, C T Corporation System at 208 S LaSalle St, Suite 814, Chicago, Illinois 60604.

3. Motorola, an indirect subsidiary of Lenovo Group Limited, is involved in the development and sale of hardware and software relating to mobile products, such as smartphones. Motorola has a place of business located at 222 W. Merchandise Mart Plaza, Suite 1800, Chicago, Illinois, 60654.

JURISDICTION AND VENUE

4. This is an action for infringement of United States patents arising under 35 U.S.C. §§ 271, 281, and 284–85, among others. This Court has subject matter jurisdiction of the action under 28 U.S.C. § 1331 and § 1338(a).

5. This Court has personal jurisdiction over Motorola pursuant to due process because, *inter alia*, (i) Motorola has done and continues to do business in Illinois; (ii) Motorola has committed and continues to commit acts of patent infringement in the State of Illinois, including making, using, offering to sell, and/or selling accused products in Illinois, and/or importing accused products into Illinois, including by Internet sales and/or sales via retail and wholesale stores, inducing others to commit acts of patent infringement in Illinois, and/or committing at least a portion of any other infringements alleged herein in Illinois; and (iii) Motorola regularly places its products within the stream of commerce—directly, through subsidiaries, or through third parties—with the expectation and knowledge that such products will be shipped to, sold, or used in Illinois and elsewhere in the United States. Thus, Motorola has established minimum contacts within Illinois and purposefully availed itself of the benefits of Illinois, and the exercise of personal jurisdiction over Motorola would not offend traditional notions of fair play and substantial justice. In addition, Motorola is amenable to service of process for this action.

6. Venue is proper in this district as to Motorola under 28 U.S.C. § 1400(b). Venue is further proper as to Motorola because it has committed and continues to commit acts of patent infringement in this district, including making, using, offering to sell, and/or selling accused products in this district, and/or importing accused products into this district, including by Internet sales and/or sales via retail and wholesale stores, inducing others to commit acts of patent

infringement in this district, and/or committing at least a portion of any other infringements alleged herein in this district.

7. Furthermore, Motorola has a regular and established place of business in this district, including at least at 222 W. Merchandise Mart Plaza, Suite 1800, Chicago, Illinois, 60654.

BACKGROUND

8. The patents-in-suit broadly cover technology used in electronic devices commonly used today, such as mobile phones, tablets, and other devices. More particularly, the patents-in-suit describe key improvements to electronic devices in the area of intelligent message recognition systems.

9. U.S. Patent Nos. 6,904,405 ("the '405 Patent") and 8,204,737 ("the '737 Patent") generally relate to the fields of speech and handwriting recognition technology. The patented technology is used today in connection with virtual keyboards on smartphones and other devices that include speech and handwriting conversion to text data. The '405 Patent discloses, *inter alia*, using language models to improve conversion of speech and handwritten data into text data by training one of the language models. The '737 Patent discloses, *inter alia*, converting speech and handwritten data into text data, where one of the language models is adjusted.

10. The inventor of the technology of these patents has multiple patents relating to the fields of speech and handwriting recognition, cryptography, digital signatures, and audio signal reconstruction. As an undergraduate at the University of Washington during the mid-1990s, he also invented a radio receiver technology with novel ways of tuning a radio among several channels.[1]

[1] https://www.wrfseattle.org/story/edwin-a-suominen-engineer-and-inventor/

11. The inventions disclosed in the ’405 and ’737 patents have been cited during patent prosecution multiple times by many of the leading technology companies worldwide, including Alcatel-Lucent (later acquired by Nokia), Amazon, Apple, Cisco, Dell, Facebook, GM, Google, HP, Honda, Honeywell, IBM, Intel, Lenovo, Microsoft, Mitsubishi, Motorola Solutions, Nuance, Palm (later acquired by HP), Panasonic, Philips, Qualcomm, Research in Motion (now Blackberry), Robert Bosch, Samsung, Siemens, Sony, SRI International, and the US Navy.

COUNT I

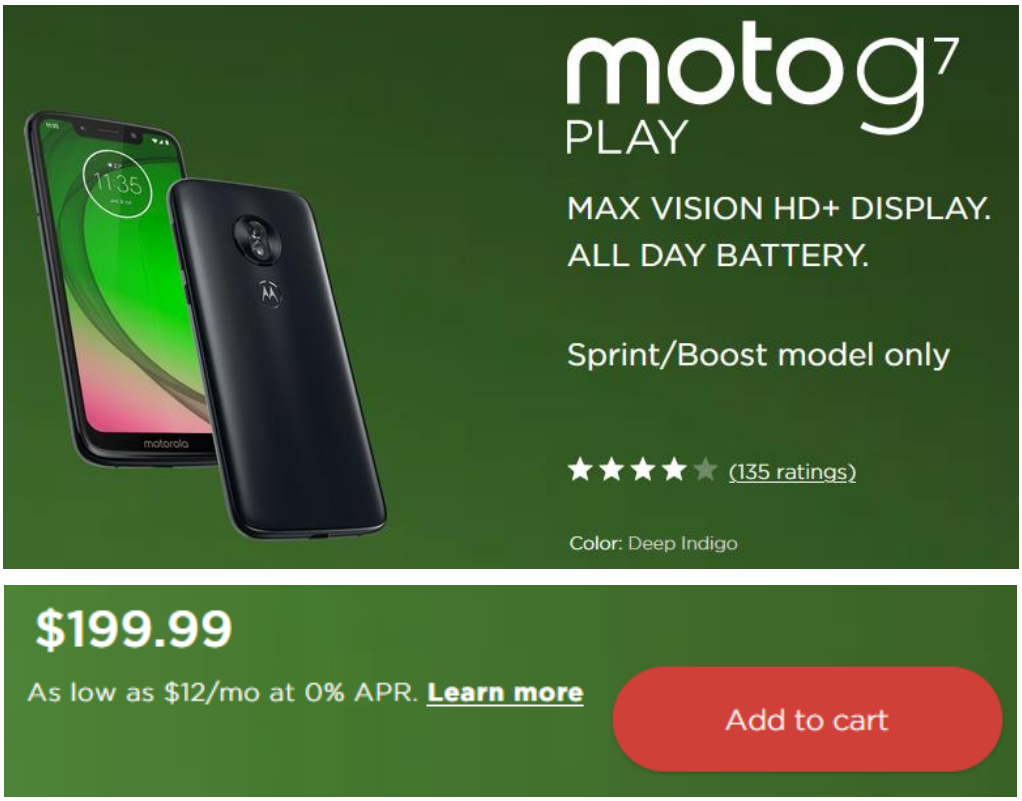
DIRECT INFRINGEMENT OF U.S. PATENT NO. 6,904,405

12. On June 7, 2005, the ’405 Patent was duly and legally issued by the United States Patent and Trademark Office for an invention entitled “Message Recognition Using Shared Language Model.”

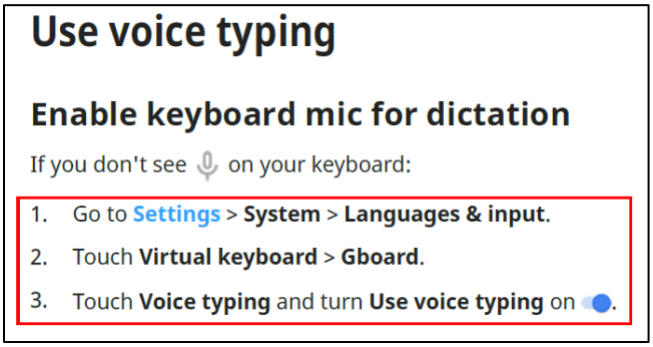
13. Buffalo Patents is the owner of the ’405 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the ’405 Patent against infringers, and to collect damages for all relevant times.

14. Motorola made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Moto G7 Play smartphone and other products[2] that include the Gboard application or similar virtual keyboard technology (“accused products”):

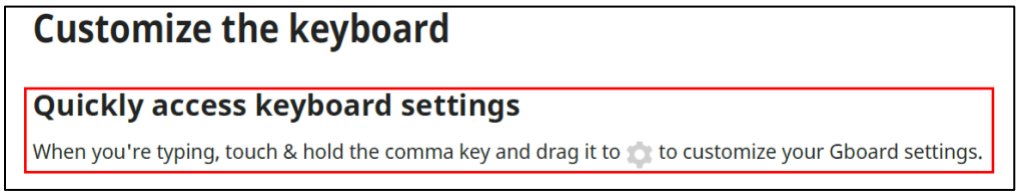
[2] *See* Appendix.



Source: https://www.motorola.com/us/smartphones-moto-g7-play/p?skuId=525



Source: https://help.motorola.com/hc/3170/90/pdf/help-moto-g7-play-90-global-en-us.pdf (Page 41)



Source: https://help.motorola.com/hc/3170/90/pdf/help-moto-g7-play-90-global-en-us.pdf (Page 175)

15. By doing so, Motorola has directly infringed (literally and/or under the doctrine of equivalents) at least Claim 7 of the ’405 Patent. Motorola’s infringement in this regard is ongoing.

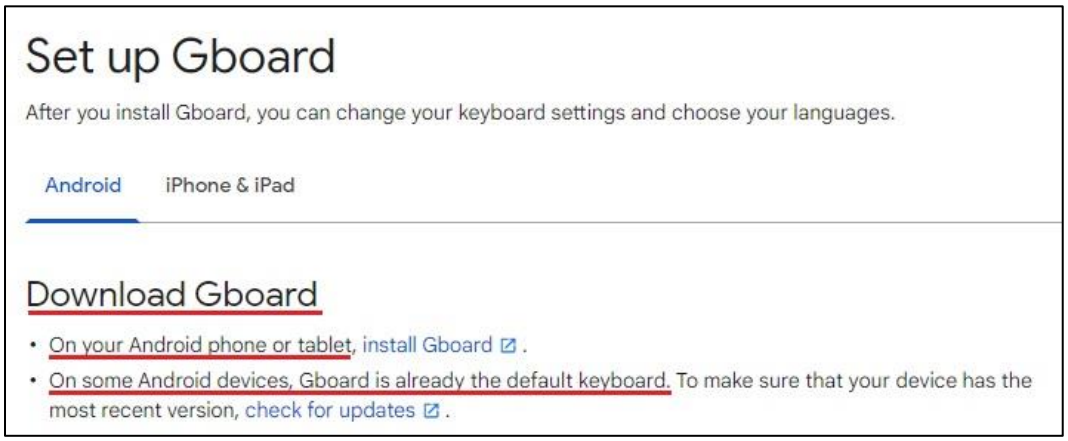
16. The Moto G7 Play smartphone is an exemplary accused product.

17. The Moto G7 Play smartphone implements a method for performing message recognition with a shared language model.

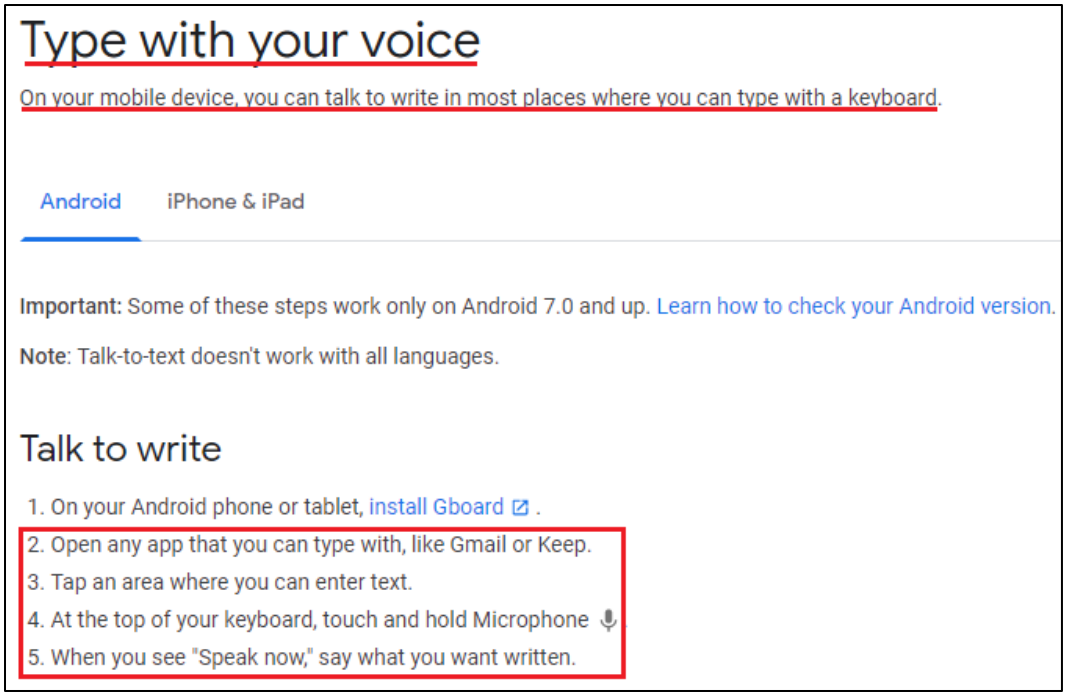
18. For example, the Moto G7 Play smartphone includes Gboard, which is the smartphone’s default virtual keyboard application. The virtual keyboard application on the Moto G7 Play smartphone includes different message recognition types, including voice to text and handwriting to text. Further, it uses an n-gram language model (“shared language model”).

> By now it's pretty clear that Gboard is one of the most popular keyboard apps for mobile devices, but even if we knew that the fact that it's got over 1 billion downloads on Google Play Store is still an impressive achievement.

Source: https://www.phonearena.com/news/Google-Gboard-keyboard-app-1-billion-downloads_id108061



Source:
https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Type with your voice

On your mobile device, you can talk to write in most places where you can type with a keyboard.

Android iPhone & iPad

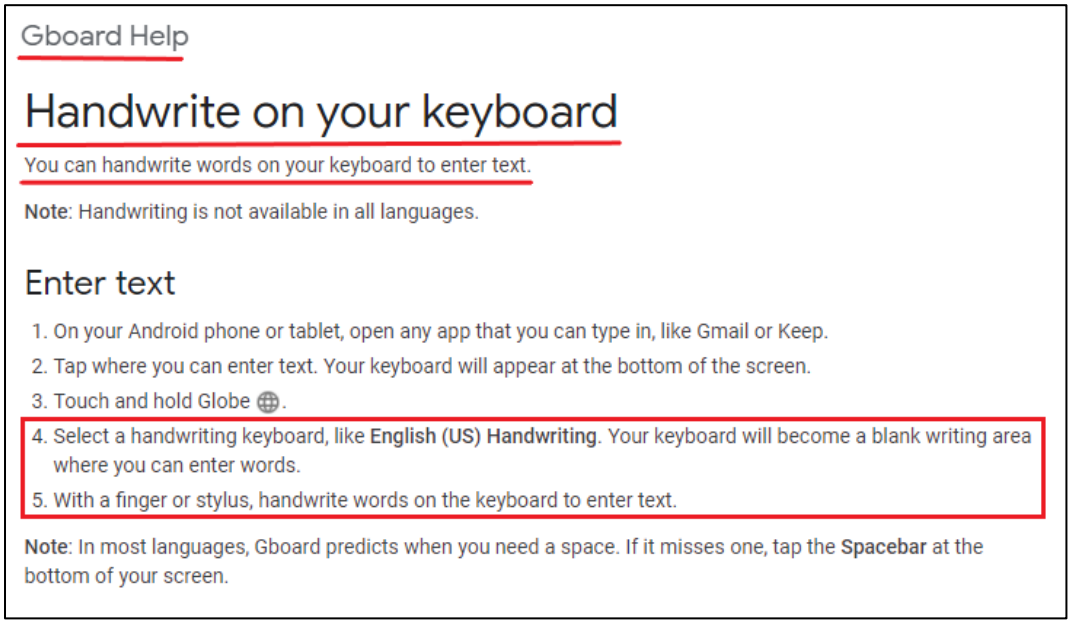
Important: Some of these steps work only on Android 7.0 and up. Learn how to check your Android version.

Note: Talk-to-text doesn't work with all languages.

Talk to write

1. On your Android phone or tablet, install Gboard .
2. Open any app that you can type with, like Gmail or Keep.
3. Tap an area where you can enter text.
4. At the top of your keyboard, touch and hold Microphone
5. When you see "Speak now," say what you want written.

Source: https://support.google.com/gboard/answer/2781851?hl=en&ref_topic=9024098



Gboard Help

Handwrite on your keyboard

You can handwrite words on your keyboard to enter text.

Note: Handwriting is not available in all languages.

Enter text

1. On your Android phone or tablet, open any app that you can type in, like Gmail or Keep.
2. Tap where you can enter text. Your keyboard will appear at the bottom of the screen.
3. Touch and hold Globe .
4. Select a handwriting keyboard, like **English (US) Handwriting**. Your keyboard will become a blank writing area where you can enter words.
5. With a finger or stylus, handwrite words on the keyboard to enter text.

Note: In most languages, Gboard predicts when you need a space. If it misses one, tap the **Spacebar** at the bottom of your screen.

Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

> Supporting features such as auto-correct, next-word prediction (predictive text) and spell-check requires the use of a machine-learning language model, such as n-gram language models, which can be used in a finite-state transduction decoder (Ouyang et al., 2017). These language models can be created based on a variety of textual sources, e.g. web crawls, external text corpora, or even wordlists (to create unigram language models). A detailed description of our standard approach to mining training data for language models across many languages can be found in Prasad et al. (2018). Since the data that we mine can be quite noisy, we apply our scalable automatic data normalization system across all languages and data sets, as described in Chua et al. (2018). Our model training algorithms are described in Allauzen et al. (2016).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 16)

> A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

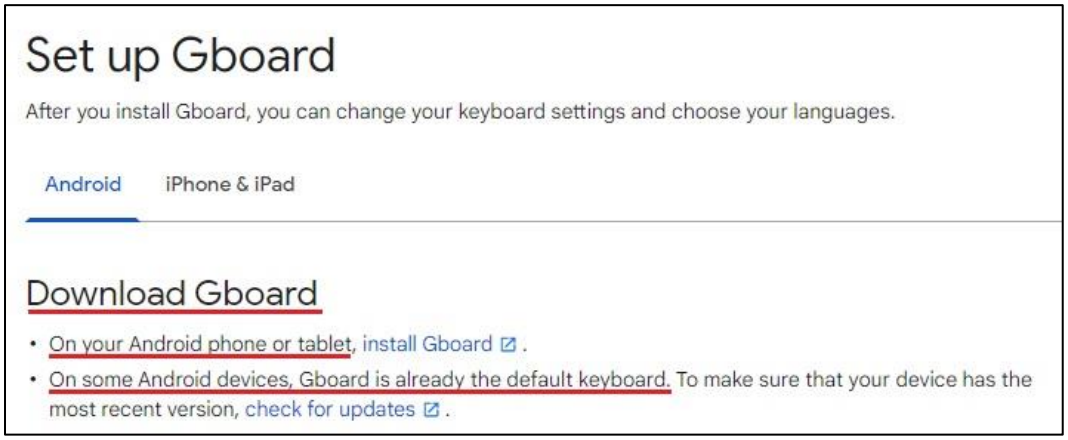
> ceding text. The n-best output labels from this state are returned. Paths containing back-off transitions to lower-orders are also considered. The primary (static) language model for the English language in Gboard is a Katz smoothed Bayesian interpolated [4] 5-gram LM containing 1.25 million n-grams, including 164,000 unigrams. Personalized user history, contacts, and email n-gram models augment the primary LM.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 1)

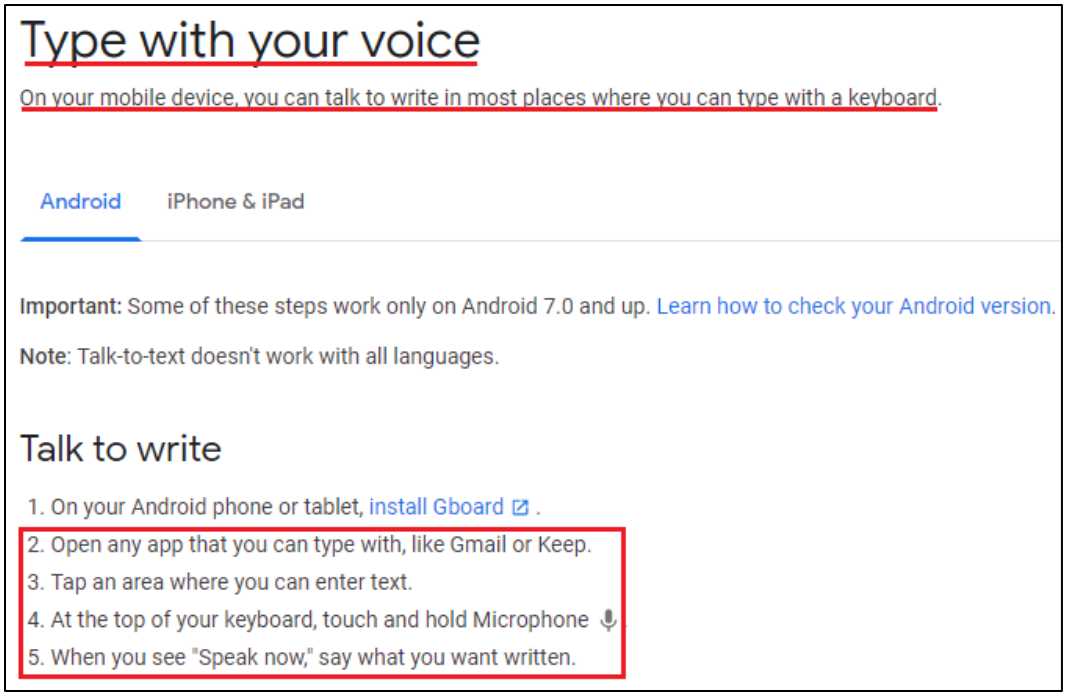
19. The method implemented by the Moto G7 Play smartphone includes performing message recognition of a first type, responsive to a first type of message input, to provide text data in accordance with both the shared language model and a first model specific to the first type of message recognition.

20. For example, the Moto G7 Play smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Moto G7 Play smartphone allows users to type a message with voice by converting the voice input into

text. It uses an on-device, all neural speech recognizer. The speech recognizer ("message recognition of a first type") uses an end-to-end Recurrent Neural Network Transducer (RNN-T) model that combines an acoustic model, a pronunciation model, and a language model. The language model used by the smartphone is an n-gram language model. The user's voice input ("first type of message input") is converted into text using the speech recognizer. The pronunciation model, the acoustic model ("first model"), and the language model ("shared language model") can generate text responsive to voice input.



Source:
https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Type with your voice

On your mobile device, you can talk to write in most places where you can type with a keyboard.

Android iPhone & iPad

Important: Some of these steps work only on Android 7.0 and up. Learn how to check your Android version.

Note: Talk-to-text doesn't work with all languages.

Talk to write

1. On your Android phone or tablet, install Gboard .
2. Open any app that you can type with, like Gmail or Keep.
3. Tap an area where you can enter text.
4. At the top of your keyboard, touch and hold Microphone
5. When you see "Speak now," say what you want written.

Source: https://support.google.com/gboard/answer/2781851?hl=en&ref_topic=9024098

The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

Today, we're happy to announce the rollout of an end-to-end, all-neural, on-device speech recognizer to power speech input in Gboard. In our recent paper, "Streaming End-to-End Speech Recognition for Mobile Devices", we present a model trained using RNN transducer (RNN-T) technology that is compact enough to reside on a phone. This means no more network latency or spottiness — the new recognizer is always available, even when you are offline. The model works at the character level, so that as you speak, it outputs words character-by-character, just as if someone was typing out what you say in real-time, and exactly as you'd expect from a keyboard dictation system.

Source: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

A Low-latency All-neural Multilingual Model

Traditional ASR systems contain separate components for acoustic, pronunciation, and language models. While there have been attempts to make some or all of the traditional ASR components multilingual [1,2,3,4], this approach can be complex and difficult to scale. E2E ASR models combine all three components into a single neural network and promise scalability and ease of parameter sharing. Recent works have extended E2E models to be multilingual [1,2], but they did not address the need for real-time speech recognition, a key requirement for applications such as the Assistant, Voice Search and GBoard dictation. For this, we turned to recent research at Google that used a Recurrent Neural Network Transducer (RNN-T) model to achieve streaming E2E ASR. The RNN-T system outputs words one character at a time, just as if someone was typing in real time, however this was not multilingual. We built upon this architecture to develop a low-latency model for *multilingual* speech recognition.

Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

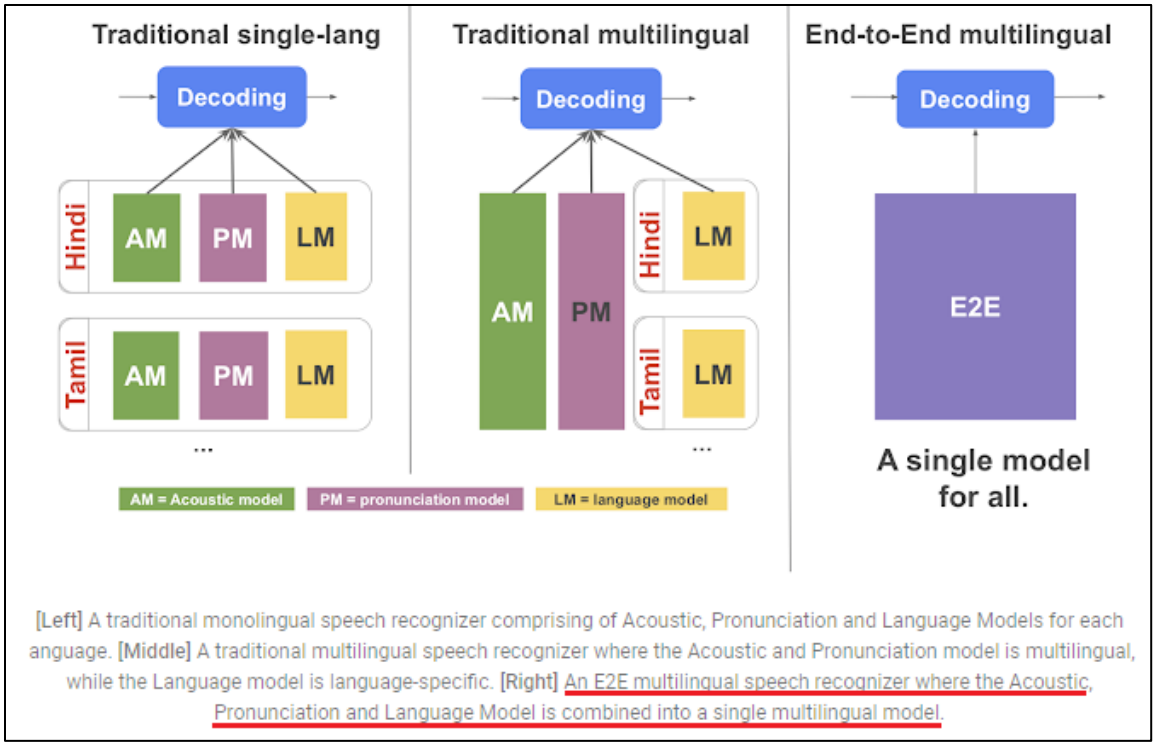
A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

The language model

Combining the acoustic and pronunciation models, we have audio coming in and words coming out. But that's not quite specific enough to provide reliable Voice Search, because you cannot just string any word together with any other word: there are word combinations that are more reasonable than others. Enter the language model, the third component of the recognition system. It calculates the frequencies of all word sequences between one to five words and thereby constrains the possible word sequences that can be formed out of the two aforementioned models to ones that are sensible combinations in language. The final search algorithm will then pick the valid word sequence that has the highest frequency of occurrence in the language.

Source: https://careers.google.com/stories/how-one-team-turned-the-dream-of-speech-recognition-into-a-reality/

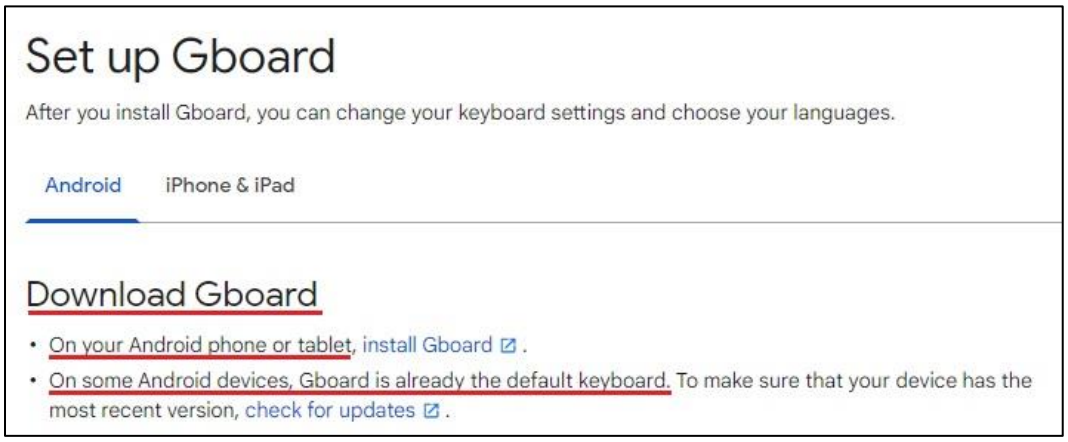


Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

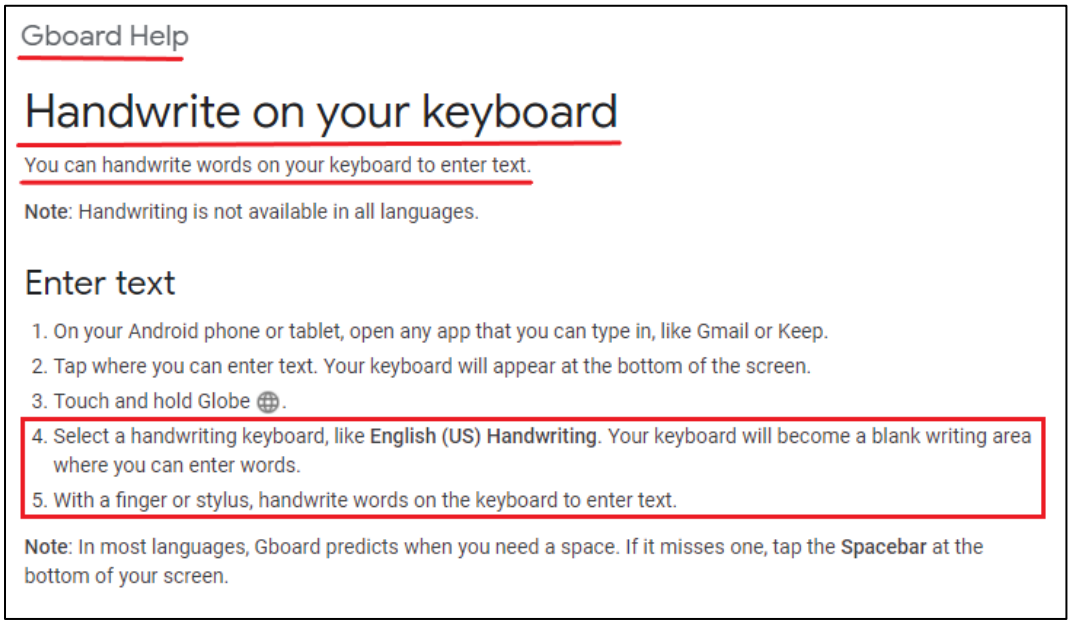
21. The method implemented by the Moto G7 Play smartphone includes performing message recognition of a second type, responsive to a second type of message input, to provide text data in accordance with both the shared language model and a second model specific to the second type of message recognition.

22. For example, the Moto G7 Play smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Moto G7 Play smartphone provides the capability to convert a user's handwriting into text. It uses an end-to-end Recurrent Neural Network (RNN) model for handwriting recognition ("message recognition of a second type"). The language model used by the smartphone is an n-gram language model. The writing area on the device's keypad records the patterns created by the user, and the corresponding handwriting information ("second type of message input") is converted into text.

23. The handwriting model ("second model") used by the smartphone is a bi-directional version of the quasi-recurrent neural network (QRNN) model. The handwriting model, in combination with the n-gram language model ("shared language model"), can generate text responsive to handwriting input.



Source: https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

> The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

> **2 End-to-end Model Architecture**
>
> Our handwriting recognition model draws its inspiration from research aimed at building end-to-end transcription models in the context of handwriting recognition [15], optical character recognition [8], and acoustic modeling in speech recognition [40]. The model architecture is constructed from common neural network blocks, i.e. bidirectional LSTMs and fully-connected layers (Figure 2). It is trained in an end-to-end manner using the CTC loss [15].

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 3)

> **RNN-Based Handwriting Recognition in Gboard**
>
> Since then, progress in machine learning has enabled new model architectures and training methodologies, allowing us to revise our initial approach (which relied on hand-designed heuristics to cut the handwritten input into single characters) and instead build a single machine learning model that operates on the whole input and reduces error rates substantially compared to the old version. We launched those new models for all latin-script based languages in Gboard at the beginning of the year, and have published the paper "Fast Multi-language LSTM-based Online Handwriting Recognition" that explains in more detail the research behind this release. In this post, we give a high-level overview of that work.

Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

> We experimented with multiple types of RNNs, and finally settled on using a bidirectional version of quasi-recurrent neural networks (QRNN). QRNNs alternate between convolutional and recurrent layers, giving it the theoretical potential for efficient parallelization, and provide a good predictive performance while keeping the number of weights comparably small. The number of weights is directly related to the size of the model that needs to be downloaded, so the smaller the better.

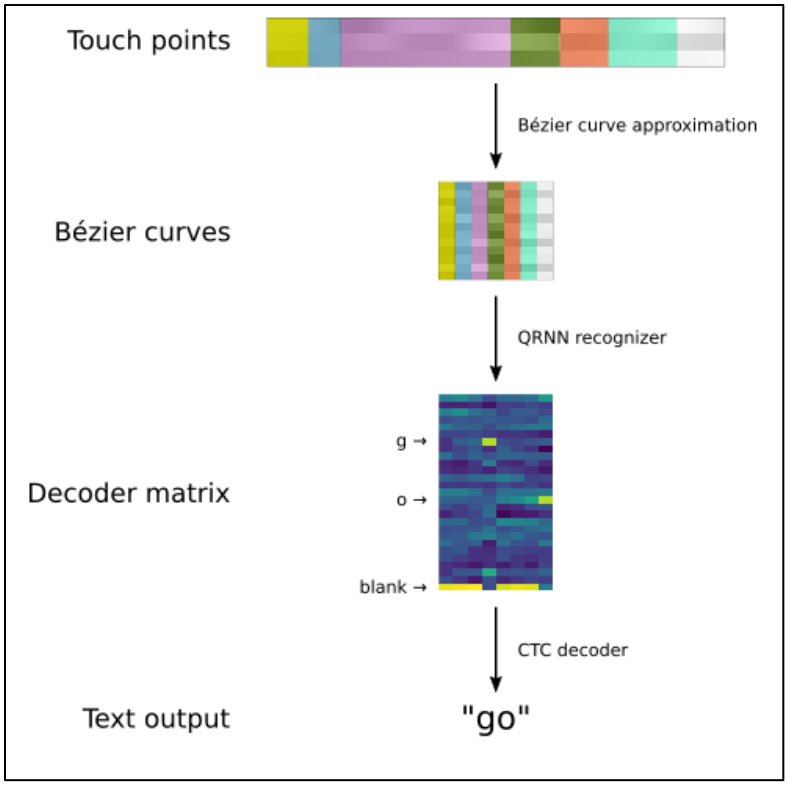
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

> In order to "decode" the curves, the recurrent neural network produces a matrix, where each column corresponds to one input curve, and each row corresponds to a letter in the alphabet. The column for a specific curve can be seen as a probability distribution over all the letters of the alphabet. However, each letter can consist of multiple curves (the *g* and *o* above, for instance, consist of four and three curves, respectively). This mismatch between the length of the output sequence from the recurrent neural network (which always matches the number of bezier curves) and the actual number of characters the input is supposed to represent is addressed by adding a special *blank* symbol to indicate no output for a particular curve, as in the Connectionist Temporal Classification (CTC) algorithm. We use a Finite State Machine Decoder to combine the outputs of the Neural Network with a character-based language model encoded as a weighted finite-state acceptor. Character sequences that are common in a language (such as "sch" in German) receive bonuses and are more likely to be output, whereas uncommon sequences are penalized. The process is visualized below.

Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

> Character Language Models: For each language we support, we build a 7-gram language model over Unicode codepoints from a large web-mined text corpus using Stupid back-off [3]. The final files are pruned to 10 million 7-grams each. Compared to our previous system [25], we found that language model size has a smaller impact on the recognition accuracy, which is likely due to the capability of recurrent neural networks to capture dependencies between consecutive characters. We therefore use smaller language models over shorter contexts.

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 6)



Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

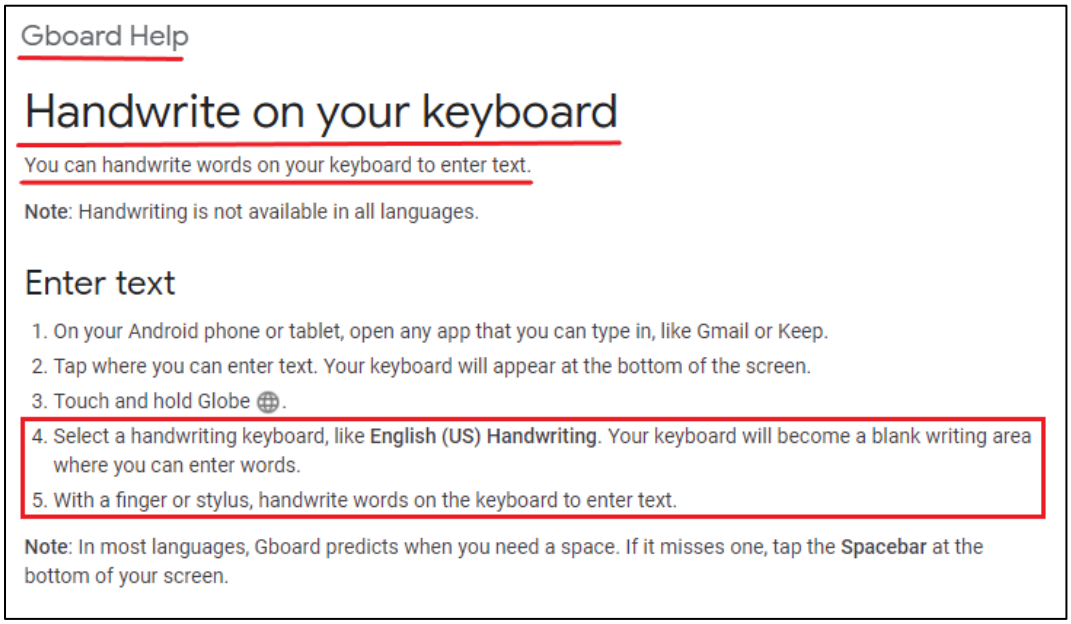
24. The method implemented by the Moto G7 Play smartphone includes training the shared language model responsive to user correction of error in message recognition of either of the first and second types, thereby improving accuracy of each of the first and second types of message recognition.

25. For example, the Moto G7 Play smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Moto G7 Play smartphone provides the capability to convert a user's voice and/or handwriting into text. It provides users with a personalized model to learn the user's preferences. In the writing area of the keypad of the device, patterns created by the user are recorded, and the corresponding handwriting information is converted into text. The data input on the virtual keyboard is utilized to learn the user's preferences, train the language model to suit the user, and update the model

locally. The smartphone builds one model for each language variety and uses on-device personalization to improve the language model by learning the user's preferences. Using an end-to-end model approach improves the predictions of both the speech and handwriting recognizers ("improving accuracy of each of the first and second types of message recognition").

26. As one example, when a user handwrites text in the writing area of the virtual keyboard, multiple suggestions can be provided to the user on the display area or interface (e.g., three suggestions to the left, center, and right). If the user selects the suggested text on the left or the right ("user correction") for the same written text multiple times, the user's selection is used to train the language model. If the user handwrites the same text at a later time, the user's previous selection is outputted as the primary suggestion (e.g., the suggested text in the center of the display area or interface).

27. Accordingly, the shared language model can be trained to be responsive to user correction of error in handwriting recognition.



Gboard Help

Handwrite on your keyboard

You can handwrite words on your keyboard to enter text.

Note: Handwriting is not available in all languages.

Enter text

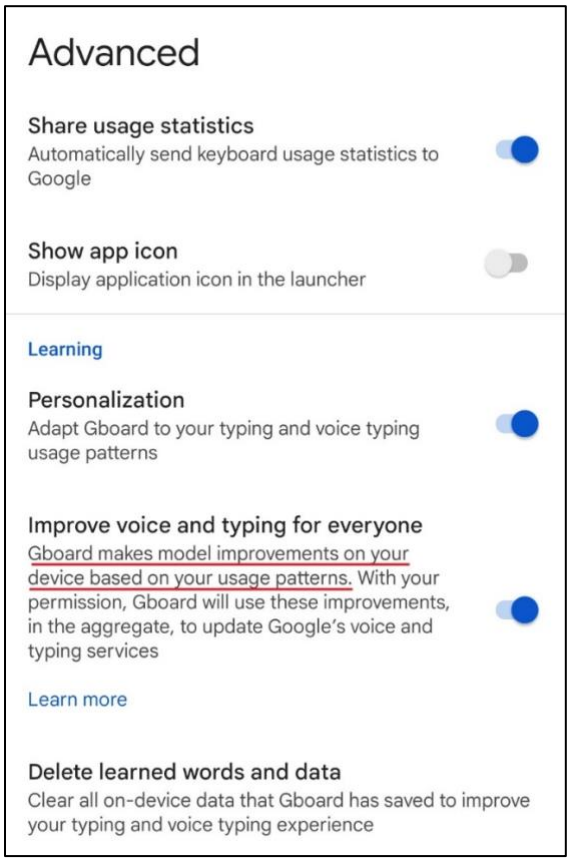
1. On your Android phone or tablet, open any app that you can type in, like Gmail or Keep.
2. Tap where you can enter text. Your keyboard will appear at the bottom of the screen.
3. Touch and hold Globe.
4. Select a handwriting keyboard, like **English (US) Handwriting**. Your keyboard will become a blank writing area where you can enter words.
5. With a finger or stylus, handwrite words on the keyboard to enter text.

Note: In most languages, Gboard predicts when you need a space. If it misses one, tap the **Spacebar** at the bottom of your screen.

Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

Mobile devices, referred to as clients, generate large volumes of personal data that can be used for training. Instead of uploading data to servers for centralized training, clients process their local data and share model updates with the server.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 3)



Source: Screenshot taken during testing.

and the CTC loss [15] (Sec. 2). We train a separate model for each *script* (Sec. 3). To support potentially many languages per script (see Table 1), language-specific language models and feature functions are used during decoding (Sec. 2.5). E.g. we have a single recognition model for Arabic script which is combined with specific language models and feature functions for our Arabic, Persian, and Urdu language recognizers. Table 1 shows the full list of scripts and languages that we currently support.

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 2)

2 End-to-end Model Architecture

Our handwriting recognition model draws its inspiration from research aimed at building end-to-end transcription models in the context of handwriting recognition [15], optical character recognition [8], and acoustic modeling in speech recognition [40]. The model architecture is constructed from common neural network blocks, i.e. bidirectional LSTMs and fully-connected layers (Figure 2). It is trained in an end-to-end manner using the CTC loss [15].

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 3)

Supporting features such as auto-correct, next-word prediction (predictive text) and spell-check requires the use of a machine-learning language model, such as n-gram language models, which can be used in a finite-state transduction decoder (Ouyang et al., 2017). These language models can be created based on a variety of textual sources, e.g. web crawls, external text corpora, or even wordlists (to create unigram language models). A detailed description of our standard approach to mining training data for language models across many languages can be found in Prasad et al. (2018). Since the data that we mine can be quite noisy, we apply our scalable automatic data normalization system across all languages and data sets, as described in Chua et al. (2018). Our model training algorithms are described in Allauzen et al. (2016).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 16)

A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

ceding text. The n-best output labels from this state are returned. Paths containing back-off transitions to lower-orders are also considered. The primary (static) language model for the English language in Gboard is a Katz smoothed Bayesian interpolated [4] 5-gram LM containing 1.25 million n-grams, including 164,000 unigrams. Personalized user history, contacts, and email n-gram models augment the primary LM.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 1)

> The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

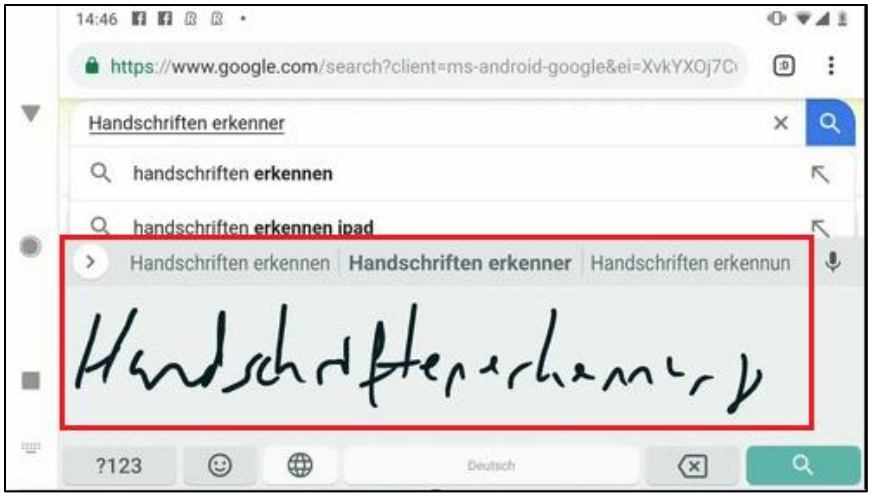
Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

> More generally, for commonly written language varieties such as English (`en`), Russian (`ru`) and Chinese (`zh`), large text corpora can be found easily across many domains. This means that the typing experience upon first use will typically be better than in languages where smaller text corpora are available, with limited domain coverage. As described above, on-device personalization can help improve pre-built generic language models as the keyboard application is used over time, by creating a personal dictionary with out-of-vocabulary words and common phrases. In our user studies, we find that such on-device personalization usually helps improve the typing experience significantly.

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 18)

> Testers have, of course, also identified areas of improvement: most commonly, users indicate that the dictionary still appears to be relatively small, presumably arising from finding too many correctly spelled words being highlighted as spelling mistakes. This makes sense, given that the training corpora we trained the language models on are typically smaller than the corpora in other languages that our testers may be familiar with. Fortunately, on-device personalization can help address this by learning words over time as the keyboard is used.

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 19)



Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

28. Buffalo Patents has been damaged as a result of the infringing conduct by Motorola alleged above. Thus, Motorola is liable to Buffalo Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

29. Buffalo Patents is only asserting method claims for the ’405 Patent, so 35 U.S.C. § 287(a) does not apply.

COUNT II

DIRECT INFRINGEMENT OF U.S. PATENT NO. 8,204,737

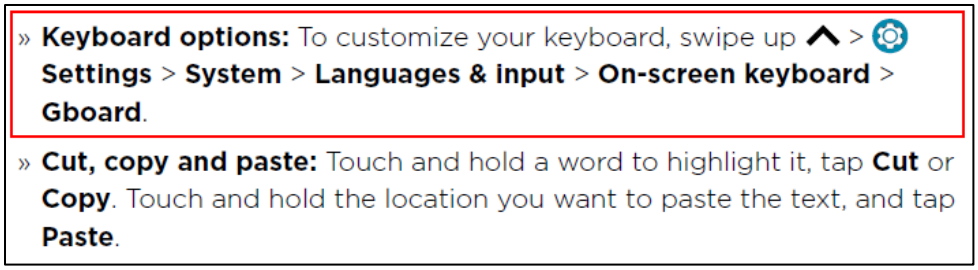
30. On June 19, 2012, the '737 Patent was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Message Recognition Using Shared Language Model."

31. Buffalo Patents is the owner of the '737 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '737 Patent against infringers, and to collect damages for all relevant times.

32. Motorola made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Motorola G Stylus 5G smartphone and other products[3] that include the Gboard application or similar virtual keyboard technology ("accused products"):

Source: https://www.motorola.com/us/smartphones-moto-g-stylus-5g/p?skuId=619

[3] *See* Appendix.



Source: https://motorola-global-portal.custhelp.com/app/answers/detail/a_id/160188/~/moto-g-stylus-5g (Page 27)

Source: https://motorola-global-portal.custhelp.com/app/answers/detail/a_id/160188/~/moto-g-stylus-5g (Page 61)

33. By doing so, Motorola has directly infringed (literally and/or under the doctrine of equivalents) at least Claims 1 and 13 of the '737 Patent. Motorola's infringement in this regard is ongoing.

34. The Motorola G Stylus 5G smartphone is an exemplary accused product.

35. The Motorola G Stylus 5G smartphone includes a system for generating text responsive to voice and handwriting input, which includes a tablet surface, wherein the system is configured to receive freehand input based at least in part on manipulation of a stylus relative to the tablet surface.

36. For example, the Motorola G Stylus 5G smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Motorola G Stylus 5G smartphone allows users to handwrite on the virtual keyboard ("receive freehand input") by providing an empty space upon which users can handwrite text using their

finger or a stylus ("based at least in part on manipulation of a stylus relative to the tablet surface").

moto g stylus 5G gives you the ultra-fast speed and built-in stylus you want. Downloads in seconds?† No problem. Jot notes and control apps with pinpoint precision? With the **moto g stylus 5G**, you got it.

Source: https://www.motorola.com/us/smartphones-moto-g-stylus-5g/p?skuId=619

By now it's pretty clear that Gboard is one of the most popular keyboard apps for mobile devices, but even if we knew that the fact that it's got over 1 billion downloads on Google Play Store is still an impressive achievement.

Source: https://www.phonearena.com/news/Google-Gboard-keyboard-app-1-billion-downloads_id108061

Set up Gboard

After you install Gboard, you can change your keyboard settings and choose your languages.

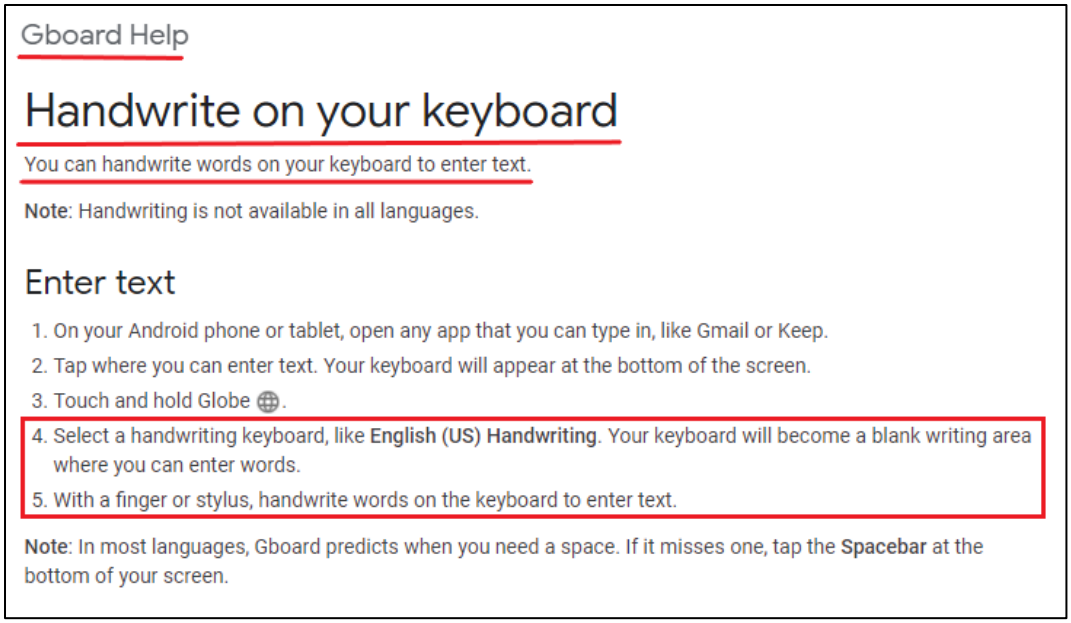
Android iPhone & iPad

Download Gboard

- On your Android phone or tablet, install Gboard .
- On some Android devices, Gboard is already the default keyboard. To make sure that your device has the most recent version, check for updates .

Source:

https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Gboard Help

Handwrite on your keyboard

You can handwrite words on your keyboard to enter text.

Note: Handwriting is not available in all languages.

Enter text

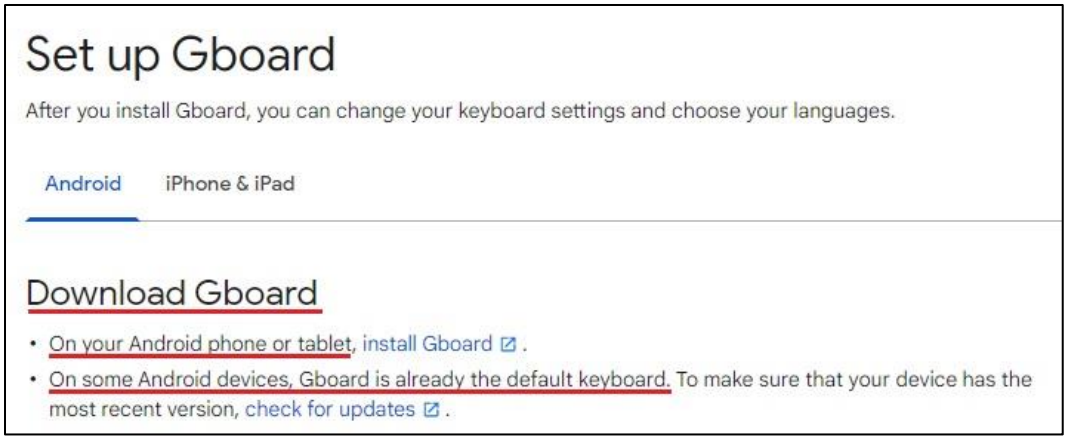
1. On your Android phone or tablet, open any app that you can type in, like Gmail or Keep.
2. Tap where you can enter text. Your keyboard will appear at the bottom of the screen.
3. Touch and hold Globe.
4. Select a handwriting keyboard, like **English (US) Handwriting**. Your keyboard will become a blank writing area where you can enter words.
5. With a finger or stylus, handwrite words on the keyboard to enter text.

Note: In most languages, Gboard predicts when you need a space. If it misses one, tap the **Spacebar** at the bottom of your screen.

Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

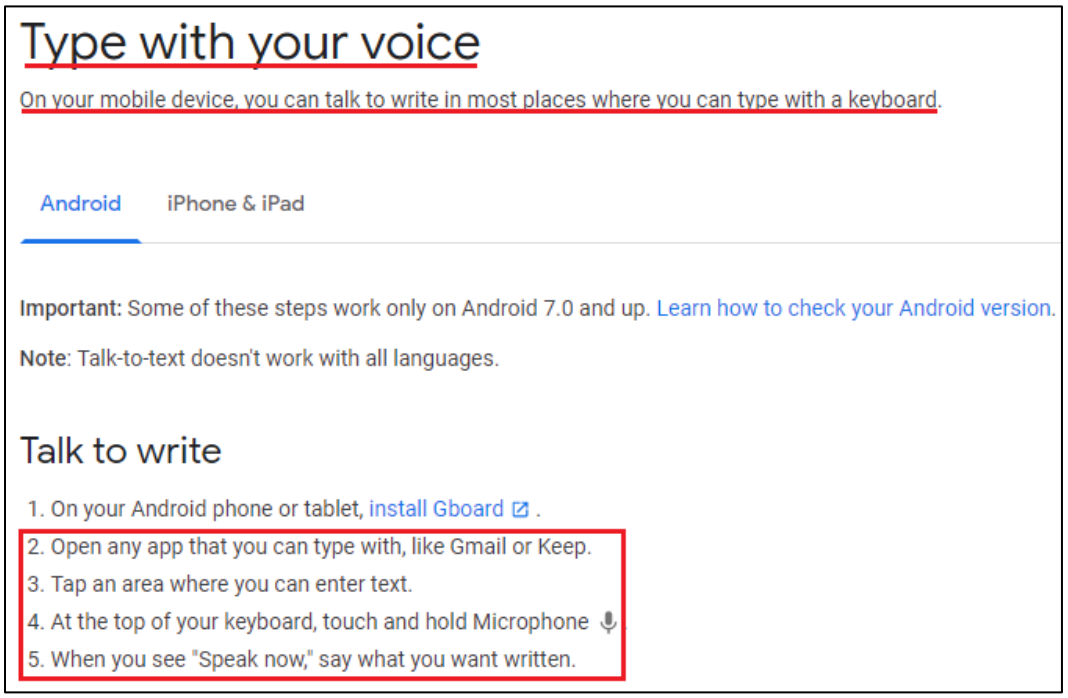
37. The Motorola G Stylus 5G smartphone includes a system for generating text responsive to voice and handwriting input, which includes a microphone for receiving voice input.

38. For example, the Motorola G Stylus 5G smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Motorola G Stylus 5G smartphone allows users to type with their voice. When a user holds the microphone button and talks, the device receives the audio input through the microphone ("receiving voice input") and converts it to text.



Source:

https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Source: https://support.google.com/gboard/answer/2781851?hl=en&ref_topic=9024098

39. The Motorola G Stylus 5G smartphone includes a system for generating text responsive to voice and handwriting input, wherein the system is configured to select a particular user message model from a plurality of user message models.

40. For example, the Motorola G Stylus 5G smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the

Motorola G Stylus 5G smartphone provides users with a personalized model to learn the user's preferences. As one example, users can select a profile ("select a particular user message model") from available profiles to be used as a virtual keyboard model. Examples of available profiles include personal and work profiles ("plurality of user message models").

> The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

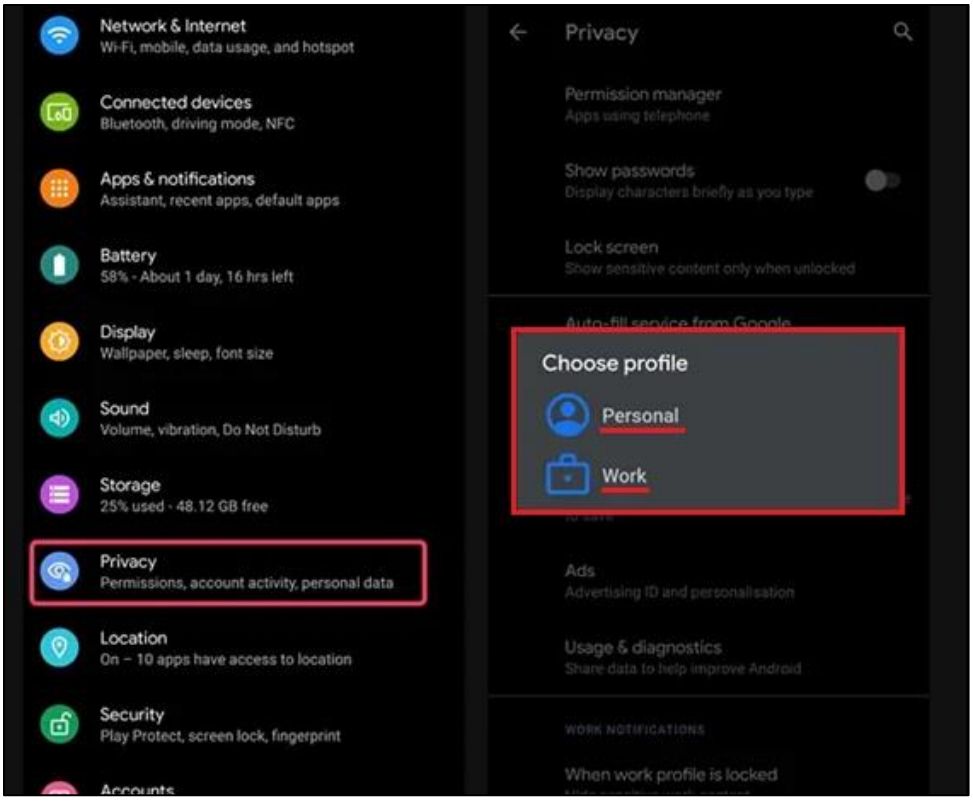
Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

> More generally, for commonly written language varieties such as English (`en`), Russian (`ru`) and Chinese (`zh`), large text corpora can be found easily across many domains. This means that the typing experience upon first use will typically be better than in languages where smaller text corpora are available, with limited domain coverage. As described above, on-device personalization can help improve pre-built generic language models as the keyboard application is used over time, by creating a personal dictionary with out-of-vocabulary words and common phrases. In our user studies, we find that such on-device personalization usually helps improve the typing experience significantly.

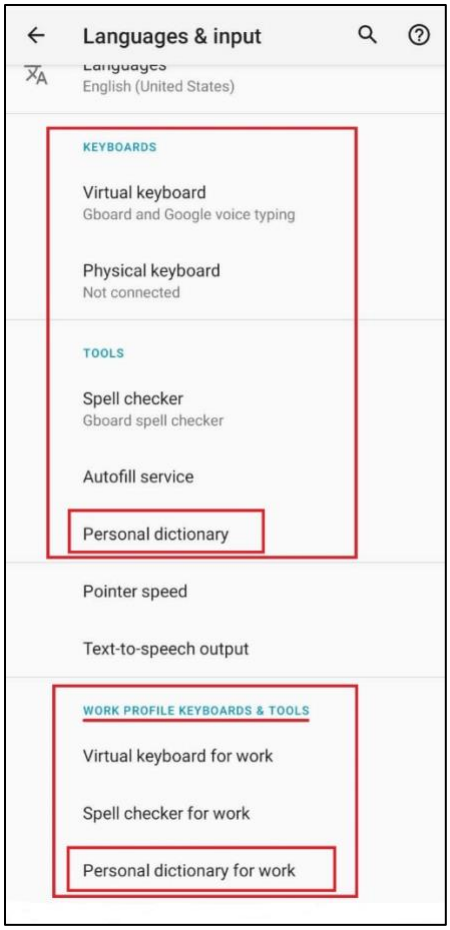
Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 18)

> A **work profile** can be set up on an Android device to separate work apps and data from personal apps and data. With a work profile you can securely and privately use the same device for work and personal purposes—your organization manages your work apps and data while your personal apps, data, and usage remain private.

Source: https://support.google.com/work/android/answer/6191949



Source: https://beebom.com/android-10-simplified-work-profiles/



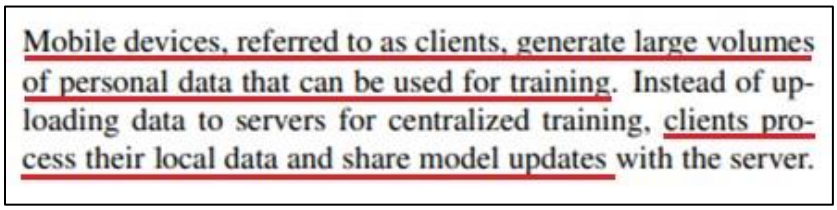
Source: Screenshot taken during testing.

41. The Motorola G Stylus 5G smartphone includes a system for generating text responsive to voice and handwriting input, wherein the system is configured to adjust a language model of a local message model based at least in part on the particular user message model, wherein the local message model includes the language model, a handwriting model, and an acoustic model.

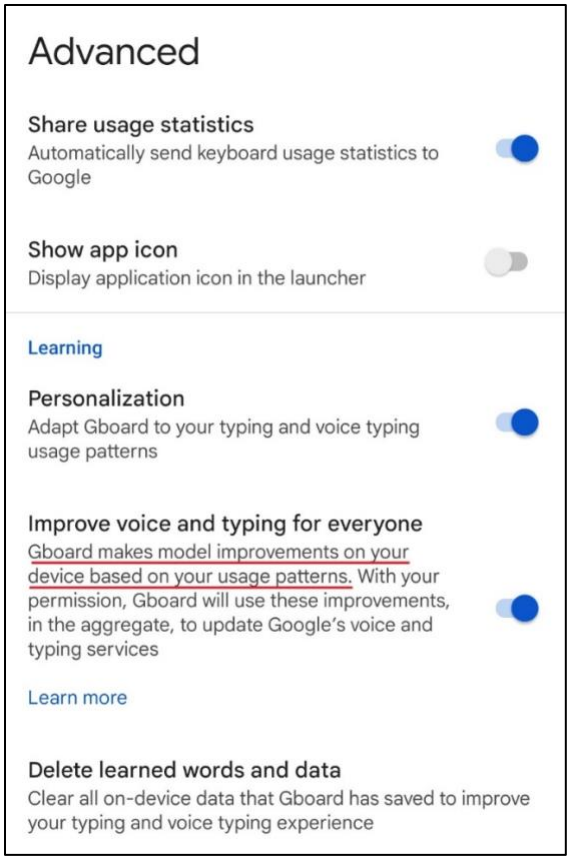
42. For example, the virtual keyboard application on the Motorola G Stylus 5G smartphone provides users with a personalized model to learn the user's preferences. As one example, users can select a profile to be used as a virtual keyboard model from available profiles, such as personal and work profiles. After selecting a profile, the data input by the user (via the virtual keyboard) is utilized to learn the user's preferences, train the language model of the

selected profile, and update the model locally. The smartphone builds one model for each language variety and uses on-device personalization to improve the language model of a selected profile by learning the user's preferences.

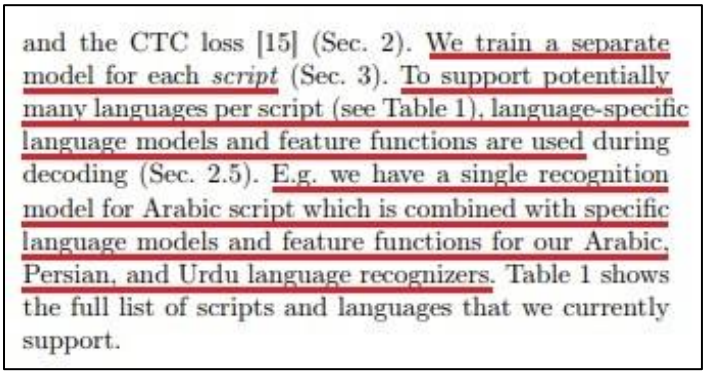
43. As one example, an end-to-end model architecture is used for the handwriting model and acoustic model. End-to-end models contain multiple functioning models tied together to form a single neural network that trains and improves itself. The end-to-end model ("local message model") contains an n-gram language model ("language model") personalized to the user, along with acoustic and handwriting models. Based on the input received from the user, the message correction data, and the selected profile of the user ("particular user message model"), the n-gram language model is trained and corrects itself to improve accuracy of prediction ("adjust a language model").


Mobile devices, referred to as clients, generate large volumes of personal data that can be used for training. Instead of uploading data to servers for centralized training, clients process their local data and share model updates with the server.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 3)



Source: Screenshot taken during testing.



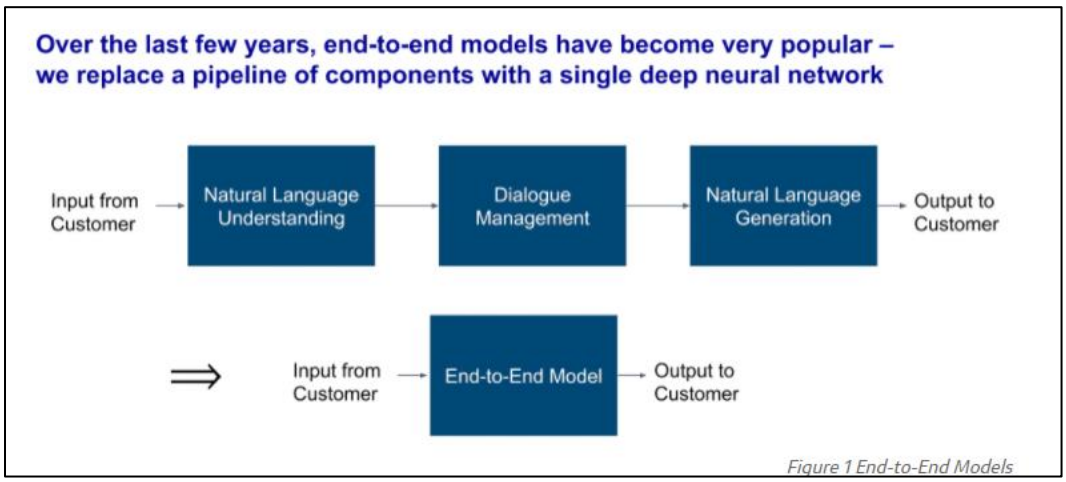
and the CTC loss [15] (Sec. 2). We train a separate model for each *script* (Sec. 3). To support potentially many languages per script (see Table 1), language-specific language models and feature functions are used during decoding (Sec. 2.5). E.g. we have a single recognition model for Arabic script which is combined with specific language models and feature functions for our Arabic, Persian, and Urdu language recognizers. Table 1 shows the full list of scripts and languages that we currently support.

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 2)

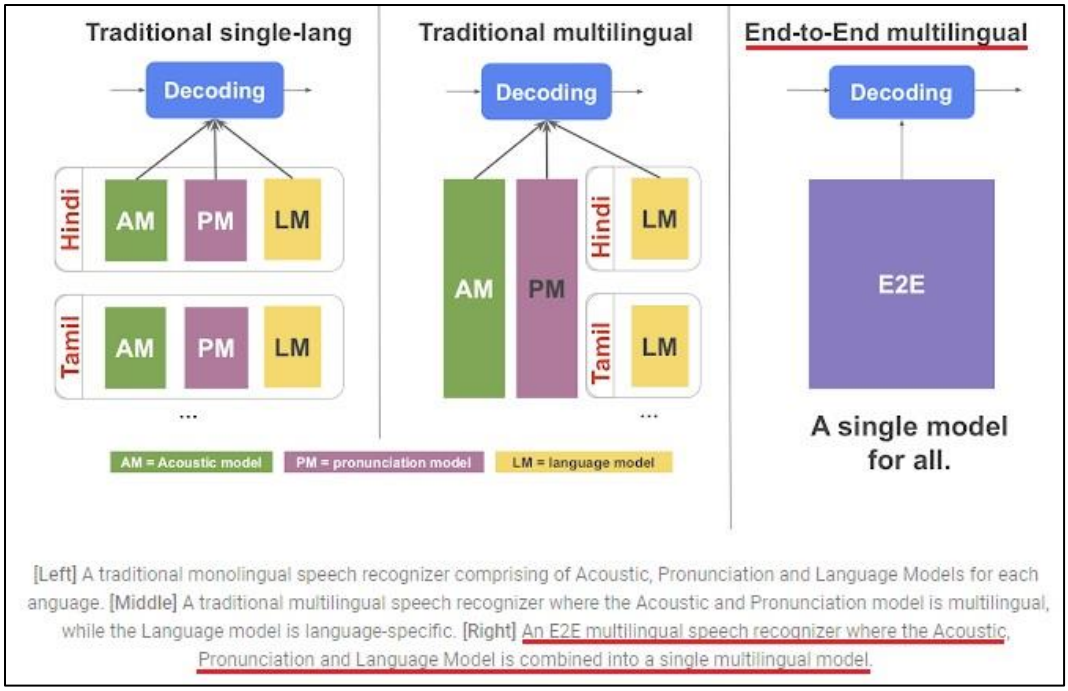
2 End-to-end Model Architecture

Our handwriting recognition model draws its inspiration from research aimed at building end-to-end transcription models in the context of handwriting recognition [15], optical character recognition [8], and acoustic modeling in speech recognition [40]. The model architecture is constructed from common neural network blocks, i.e. bidirectional LSTMs and fully-connected layers (Figure 2). It is trained in an end-to-end manner using the CTC loss [15].

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 3)



Source: https://www.capitalone.com/tech/machine-learning/pros-and-cons-of-end-to-end-models/



[Left] A traditional monolingual speech recognizer comprising of Acoustic, Pronunciation and Language Models for each anguage. [Middle] A traditional multilingual speech recognizer where the Acoustic and Pronunciation model is multilingual, while the Language model is language-specific. [Right] An E2E multilingual speech recognizer where the Acoustic, Pronunciation and Language Model is combined into a single multilingual model.

Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

Supporting features such as auto-correct, next-word prediction (predictive text) and spell-check requires the use of a machine-learning language model, such as n-gram language models, which can be used in a finite-state transduction decoder (Ouyang et al., 2017). These language models can be created based on a variety of textual sources, e.g. web crawls, external text corpora, or even wordlists (to create unigram language models). A detailed description of our standard approach to mining training data for language models across many languages can be found in Prasad et al. (2018). Since the data that we mine can be quite noisy, we apply our scalable automatic data normalization system across all languages and data sets, as described in Chua et al. (2018). Our model training algorithms are described in Allauzen et al. (2016).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 16)

A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

> ceding text. The n-best output labels from this state are returned. Paths containing back-off transitions to lower-orders are also considered. The primary (static) language model for the English language in Gboard is a Katz smoothed Bayesian interpolated [4] 5-gram LM containing 1.25 million n-grams, including 164,000 unigrams. Personalized user history, contacts, and email n-gram models augment the primary LM.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 1)

> The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

> More generally, for commonly written language varieties such as English (`en`), Russian (`ru`) and Chinese (`zh`), large text corpora can be found easily across many domains. This means that the typing experience upon first use will typically be better than in languages where smaller text corpora are available, with limited domain coverage. As described above, on-device personalization can help improve pre-built generic language models as the keyboard application is used over time, by creating a personal dictionary with out-of-vocabulary words and common phrases. In our user studies, we find that such on-device personalization usually helps improve the typing experience significantly.

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 18)

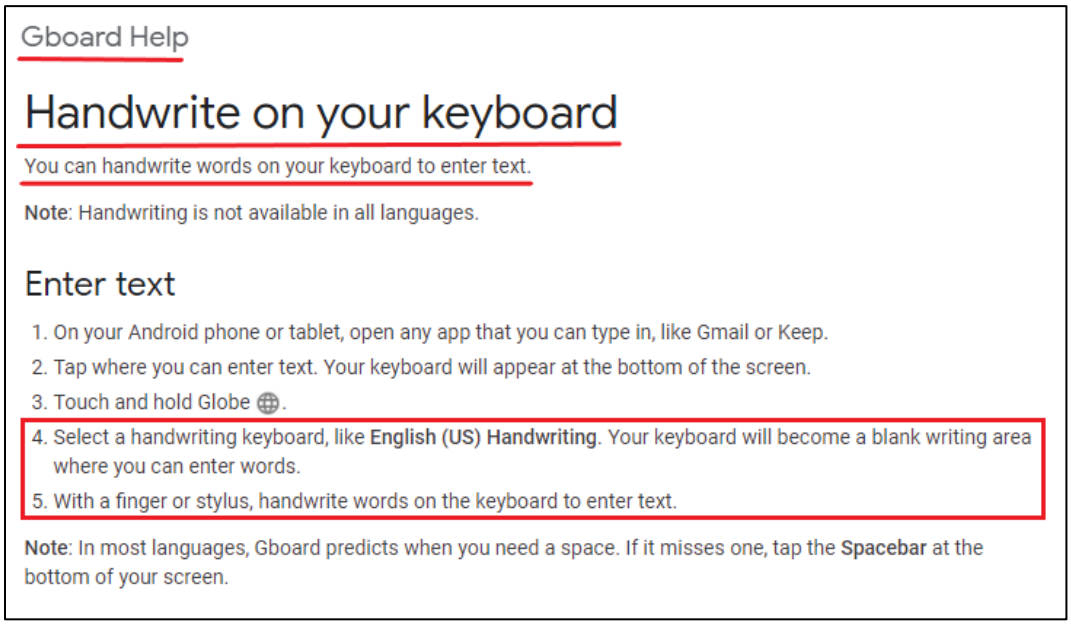
> Testers have, of course, also identified areas of improvement: most commonly, users indicate that the dictionary still appears to be relatively small, presumably arising from finding too many correctly spelled words being highlighted as spelling mistakes. This makes sense, given that the training corpora we trained the language models on are typically smaller than the corpora in other languages that our testers may be familiar with. Fortunately, on-device personalization can help address this by learning words over time as the keyboard is used.

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 19)

44. The Motorola G Stylus 5G smartphone includes a system for generating text responsive to voice and handwriting input, wherein the system is configured to generate text data

based at least in part on the freehand input, the adjusted language model, and the handwriting model.

45. For example, the virtual keyboard application on the Motorola G Stylus 5G smartphone provides a feature for converting handwriting to text. It uses an end-to-end model based on the Recurrent Neural Network (RNN), which combines handwriting and language models for handwriting recognition. The language model used by the smartphone is an n-gram language model. In the writing area of the keypad of the device, patterns created by the user are recorded, and the corresponding handwriting information ("freehand input") is converted into text using a handwriting model ("handwriting model"), which is a bi-directional version of the quasi-recurrent neural network (QRNN) model. The user-personalized language model ("adjusted language model") is used to generate text ("generate text data") responsive to handwriting input ("based at least in part on the freehand input").



Gboard Help

Handwrite on your keyboard

You can handwrite words on your keyboard to enter text.

Note: Handwriting is not available in all languages.

Enter text

1. On your Android phone or tablet, open any app that you can type in, like Gmail or Keep.
2. Tap where you can enter text. Your keyboard will appear at the bottom of the screen.
3. Touch and hold Globe.
4. Select a handwriting keyboard, like **English (US) Handwriting**. Your keyboard will become a blank writing area where you can enter words.
5. With a finger or stylus, handwrite words on the keyboard to enter text.

Note: In most languages, Gboard predicts when you need a space. If it misses one, tap the **Spacebar** at the bottom of your screen.

Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

2 End-to-end Model Architecture

Our handwriting recognition model draws its inspiration from research aimed at building end-to-end transcription models in the context of handwriting recognition [15], optical character recognition [8], and acoustic modeling in speech recognition [40]. The model architecture is constructed from common neural network blocks, i.e. bidirectional LSTMs and fully-connected layers (Figure 2). It is trained in an end-to-end manner using the CTC loss [15].

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 3)

RNN-Based Handwriting Recognition in Gboard

Since then, progress in machine learning has enabled new model architectures and training methodologies, allowing us to revise our initial approach (which relied on hand-designed heuristics to cut the handwritten input into single characters) and instead build a single machine learning model that operates on the whole input and reduces error rates substantially compared to the old version. We launched those new models for all latin-script based languages in Gboard at the beginning of the year, and have published the paper "Fast Multi-language LSTM-based Online Handwriting Recognition" that explains in more detail the research behind this release. In this post, we give a high-level overview of that work.

Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

We experimented with multiple types of RNNs, and finally settled on using a bidirectional version of quasi-recurrent neural networks (QRNN). QRNNs alternate between convolutional and recurrent layers, giving it the theoretical potential for efficient parallelization, and provide a good predictive performance while keeping the number of weights comparably small. The number of weights is directly related to the size of the model that needs to be downloaded, so the smaller the better.

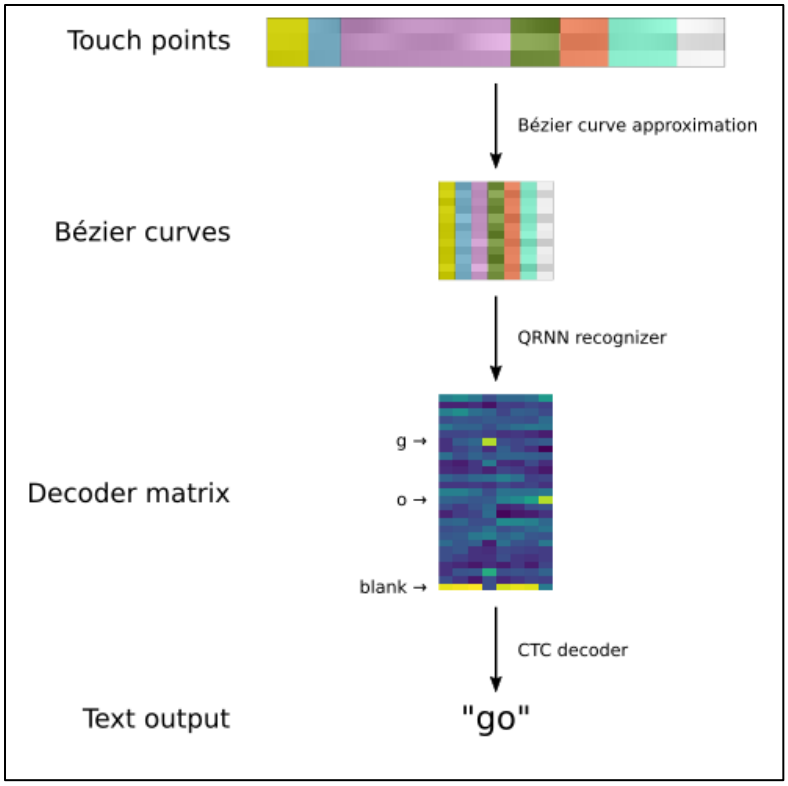
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

In order to "decode" the curves, the recurrent neural network produces a matrix, where each column corresponds to one input curve, and each row corresponds to a letter in the alphabet. The column for a specific curve can be seen as a probability distribution over all the letters of the alphabet. However, each letter can consist of multiple curves (the *g* and *o* above, for instance, consist of four and three curves, respectively). This mismatch between the length of the output sequence from the recurrent neural network (which always matches the number of bezier curves) and the actual number of characters the input is supposed to represent is addressed by adding a special *blank* symbol to indicate no output for a particular curve, as in the Connectionist Temporal Classification (CTC) algorithm. We use a Finite State Machine Decoder to combine the outputs of the Neural Network with a character-based language model encoded as a weighted finite-state acceptor. Character sequences that are common in a language (such as "sch" in German) receive bonuses and are more likely to be output, whereas uncommon sequences are penalized. The process is visualized below.

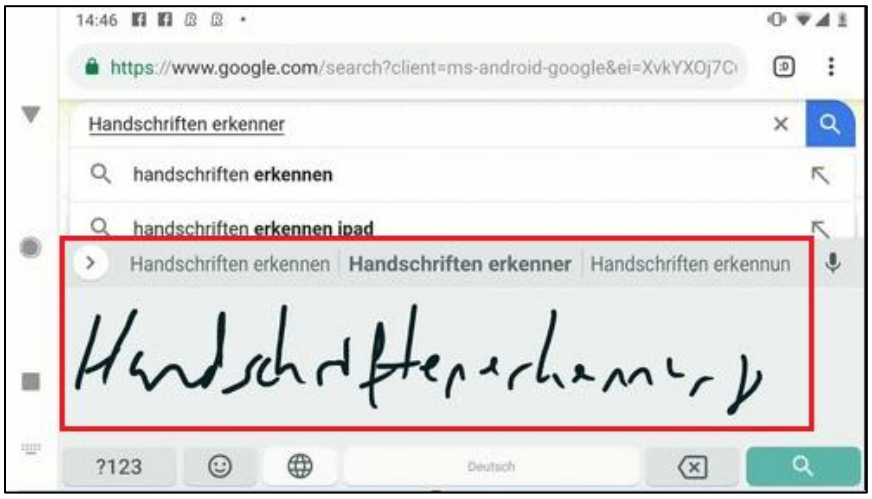
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

Character Language Models: For each language we support, we build a 7-gram language model over Unicode codepoints from a large web-mined text corpus using Stupid back-off [3]. The final files are pruned to 10 million 7-grams each. Compared to our previous system [25], we found that language model size has a smaller impact on the recognition accuracy, which is likely due to the capability of recurrent neural networks to capture dependencies between consecutive characters. We therefore use smaller language models over shorter contexts.

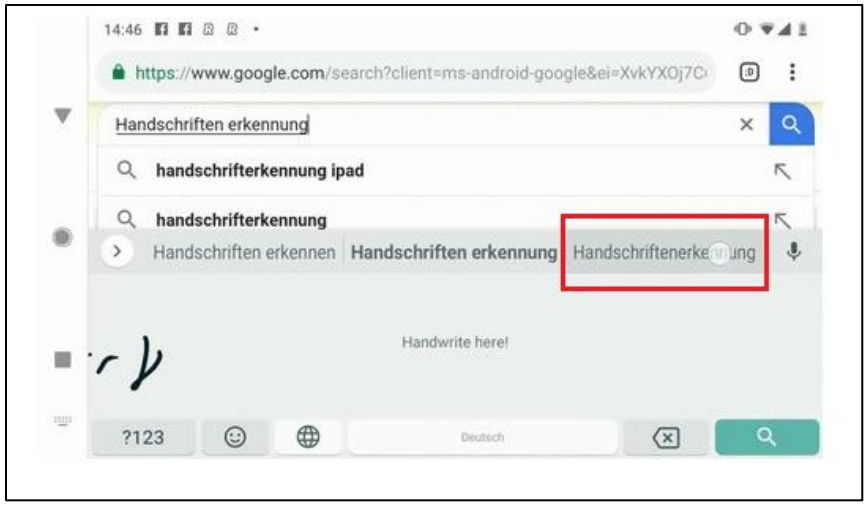
Source: https://arxiv.org/pdf/1902.10525.pdf (Page 6)



Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html



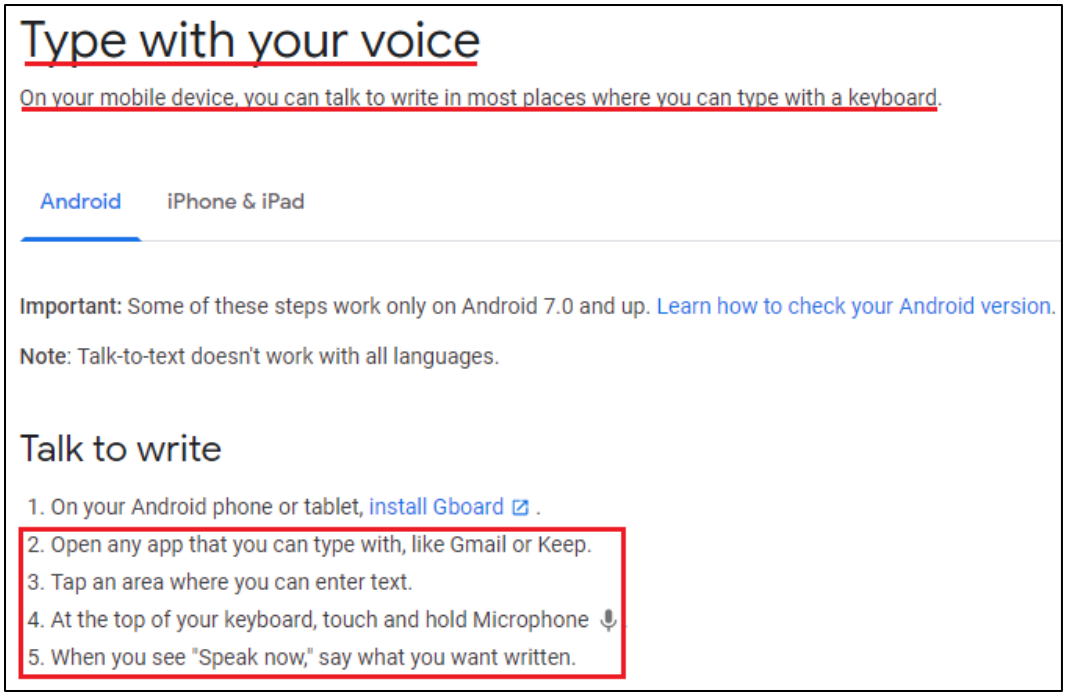
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html



Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

46. The Motorola G Stylus 5G smartphone includes a system for generating text responsive to voice and handwriting input, wherein the system is configured to generate text data based at least in part on the voice input, the adjusted language model, and the acoustic model.

47. For example, the virtual keyboard application on the Motorola G Stylus 5G smartphone uses an on-device, all neural speech recognizer. The speech recognizer uses an end-to-end RNN-T model that combines an acoustic model, pronunciation model, and a language model. The language model is an n-gram language model. The user's voice input is converted into text using the speech recognizer. The pronunciation model, acoustic model ("acoustic model"), and the user-personalized language model ("adjusted language model") can generate text responsive to voice input.



Type with your voice

On your mobile device, you can talk to write in most places where you can type with a keyboard.

Android iPhone & iPad

Important: Some of these steps work only on Android 7.0 and up. Learn how to check your Android version.

Note: Talk-to-text doesn't work with all languages.

Talk to write

1. On your Android phone or tablet, install Gboard .
2. Open any app that you can type with, like Gmail or Keep.
3. Tap an area where you can enter text.
4. At the top of your keyboard, touch and hold Microphone
5. When you see "Speak now," say what you want written.

Source: https://support.google.com/gboard/answer/2781851?hl=en&ref_topic=9024098

The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

Today, we're happy to announce the rollout of an end-to-end, all-neural, on-device speech recognizer to power speech input in Gboard. In our recent paper, "Streaming End-to-End Speech Recognition for Mobile Devices", we present a model trained using RNN transducer (RNN-T) technology that is compact enough to reside on a phone. This means no more network latency or spottiness — the new recognizer is always available, even when you are offline. The model works at the character level, so that as you speak, it outputs words character-by-character, just as if someone was typing out what you say in real-time, and exactly as you'd expect from a keyboard dictation system.

Source: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

A Low-latency All-neural Multilingual Model

Traditional ASR systems contain separate components for acoustic, pronunciation, and language models. While there have been attempts to make some or all of the traditional ASR components multilingual [1,2,3,4], this approach can be complex and difficult to scale. E2E ASR models combine all three components into a single neural network and promise scalability and ease of parameter sharing. Recent works have extended E2E models to be multilingual [1,2], but they did not address the need for real-time speech recognition, a key requirement for applications such as the Assistant, Voice Search and GBoard dictation. For this, we turned to recent research at Google that used a Recurrent Neural Network Transducer (RNN-T) model to achieve streaming E2E ASR. The RNN-T system outputs words one character at a time, just as if someone was typing in real time, however this was not multilingual. We built upon this architecture to develop a low-latency model for *multilingual* speech recognition.

Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

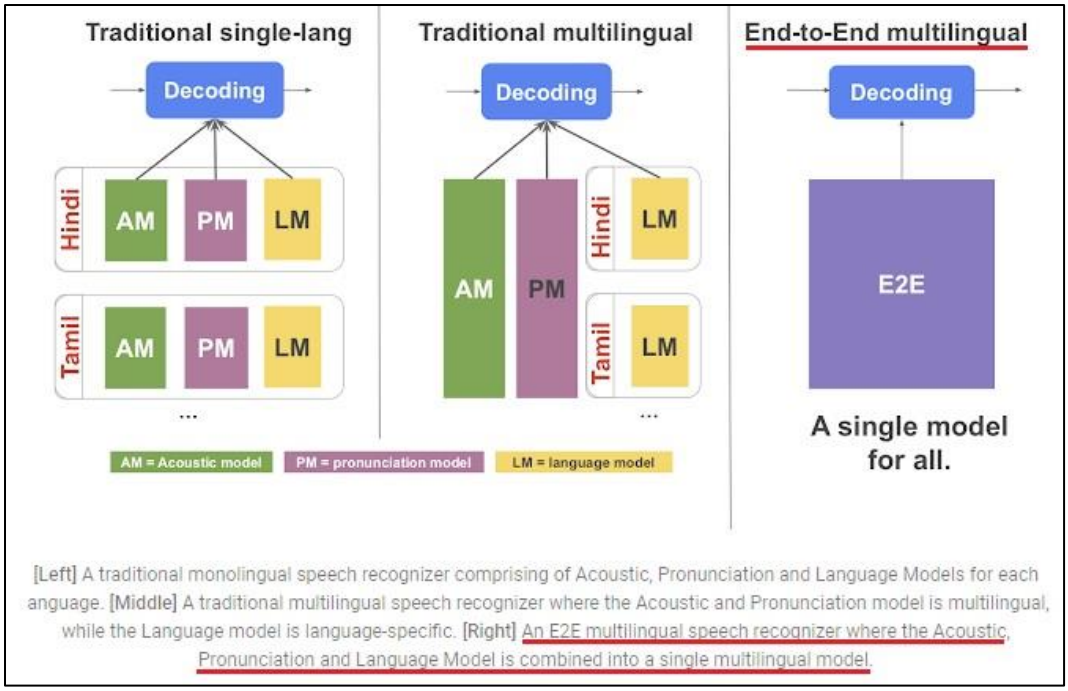
A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

The language model

Combining the acoustic and pronunciation models, we have audio coming in and words coming out. But that's not quite specific enough to provide reliable Voice Search, because you cannot just string any word together with any other word: there are word combinations that are more reasonable than others. Enter the language model, the third component of the recognition system. It calculates the frequencies of all word sequences between one to five words and thereby constrains the possible word sequences that can be formed out of the two aforementioned models to ones that are sensible combinations in language. The final search algorithm will then pick the valid word sequence that has the highest frequency of occurrence in the language.

Source: https://careers.google.com/stories/how-one-team-turned-the-dream-of-speech-recognition-into-a-reality/



Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

48. The Motorola G Stylus 5G smartphone implements a method that includes the step of receiving, at a device, freehand input based at least in part on manipulation of a stylus.

49. For example, the Motorola G Stylus 5G smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Motorola G Stylus 5G smartphone allows users to handwrite on the virtual keyboard ("receiving, at a device, freehand input") by providing an empty space upon which users can handwrite text using their finger or a stylus ("based at least in part on manipulation of a stylus").

moto g stylus 5G gives you the ultra-fast speed and built-in stylus you want. Downloads in seconds?† No problem. Jot notes and control apps with pinpoint precision? With the **moto g stylus 5G**, you got it.

Source: https://www.motorola.com/us/smartphones-moto-g-stylus-5g/p?skuId=619

By now it's pretty clear that Gboard is one of the most popular keyboard apps for mobile devices, but even if we knew that the fact that it's got over 1 billion downloads on Google Play Store is still an impressive achievement.

Source: https://www.phonearena.com/news/Google-Gboard-keyboard-app-1-billion-downloads_id108061

Set up Gboard

After you install Gboard, you can change your keyboard settings and choose your languages.

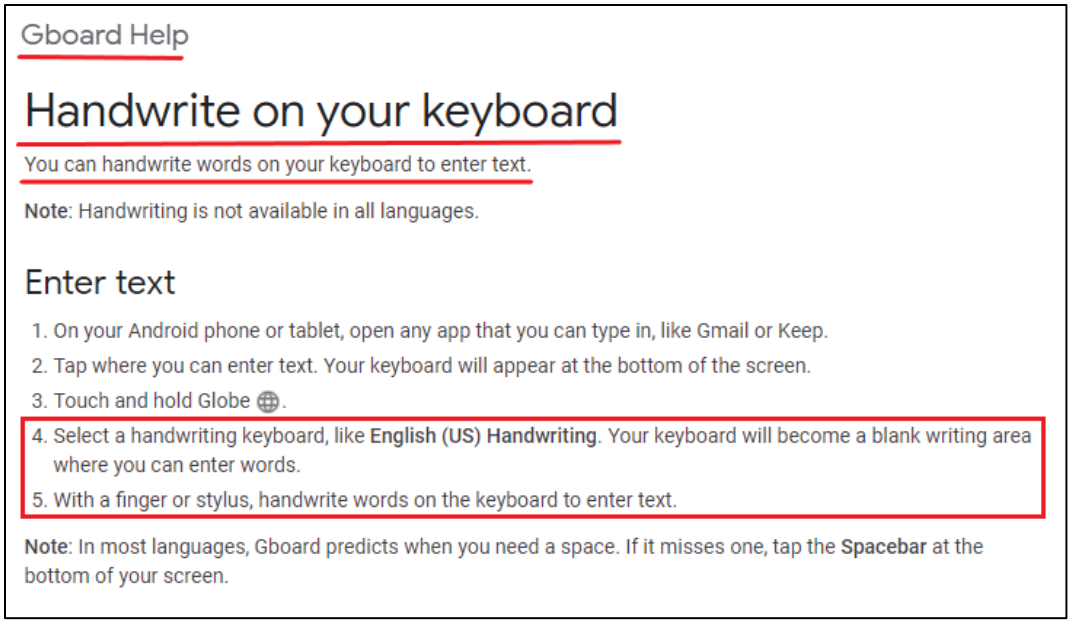
Android iPhone & iPad

Download Gboard

- On your Android phone or tablet, install Gboard .
- On some Android devices, Gboard is already the default keyboard. To make sure that your device has the most recent version, check for updates .

Source:

https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Gboard Help

Handwrite on your keyboard

You can handwrite words on your keyboard to enter text.

Note: Handwriting is not available in all languages.

Enter text

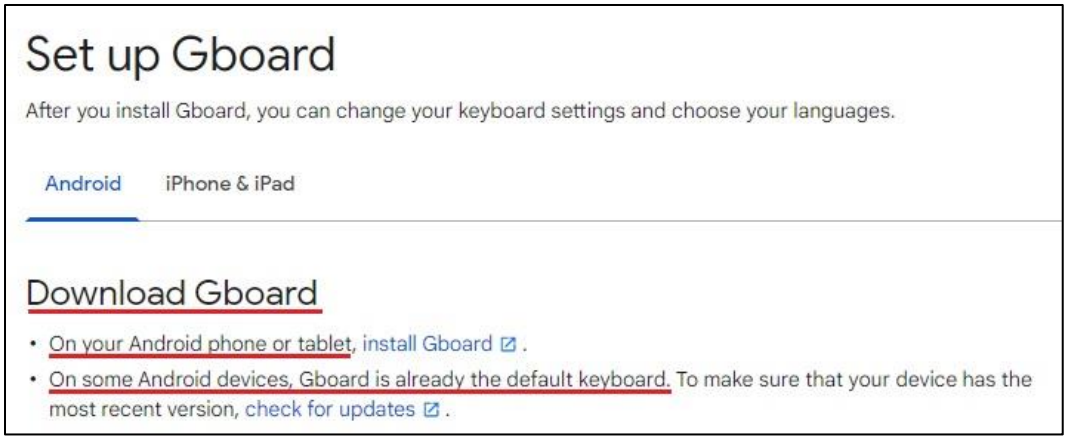
1. On your Android phone or tablet, open any app that you can type in, like Gmail or Keep.
2. Tap where you can enter text. Your keyboard will appear at the bottom of the screen.
3. Touch and hold Globe.
4. Select a handwriting keyboard, like **English (US) Handwriting**. Your keyboard will become a blank writing area where you can enter words.
5. With a finger or stylus, handwrite words on the keyboard to enter text.

Note: In most languages, Gboard predicts when you need a space. If it misses one, tap the **Spacebar** at the bottom of your screen.

Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

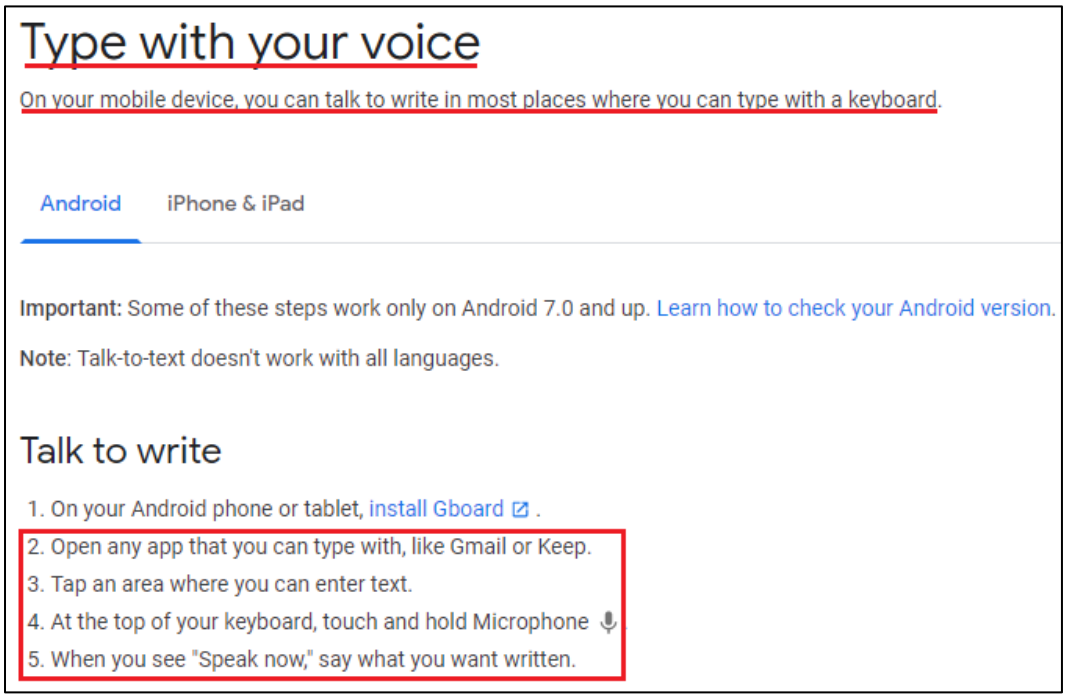
50. The method implemented by the Motorola G Stylus 5G smartphone includes the step of receiving, at the device, audio input based at least in part on information received at a microphone.

51. For example, the Motorola G Stylus 5G smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the Motorola G Stylus 5G smartphone allows users to type with their voice ("information received at a microphone"). When a user holds the microphone button and talks, the device receives the audio input ("receiving, at the device, audio input") and converts it to text.



Source:

https://support.google.com/gboard/answer/6380730?hl=en&co=GENIE.Platform%3DAndroid



Source: https://support.google.com/gboard/answer/2781851?hl=en&ref_topic=9024098

52. The method implemented by the Motorola G Stylus 5G smartphone includes the step of selecting, via the device, a selected user message model from a plurality of user message models.

53. For example, the Motorola G Stylus 5G smartphone includes Gboard, which is the smartphone's default virtual keyboard application. The virtual keyboard application on the

Motorola G Stylus 5G smartphone provides users with a personalized model to learn the user's preferences. As one example, users can select a profile ("selected user message model") from available profiles to be used as a virtual keyboard model. Examples of available profiles include personal and work profiles ("plurality of user message models").

> The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

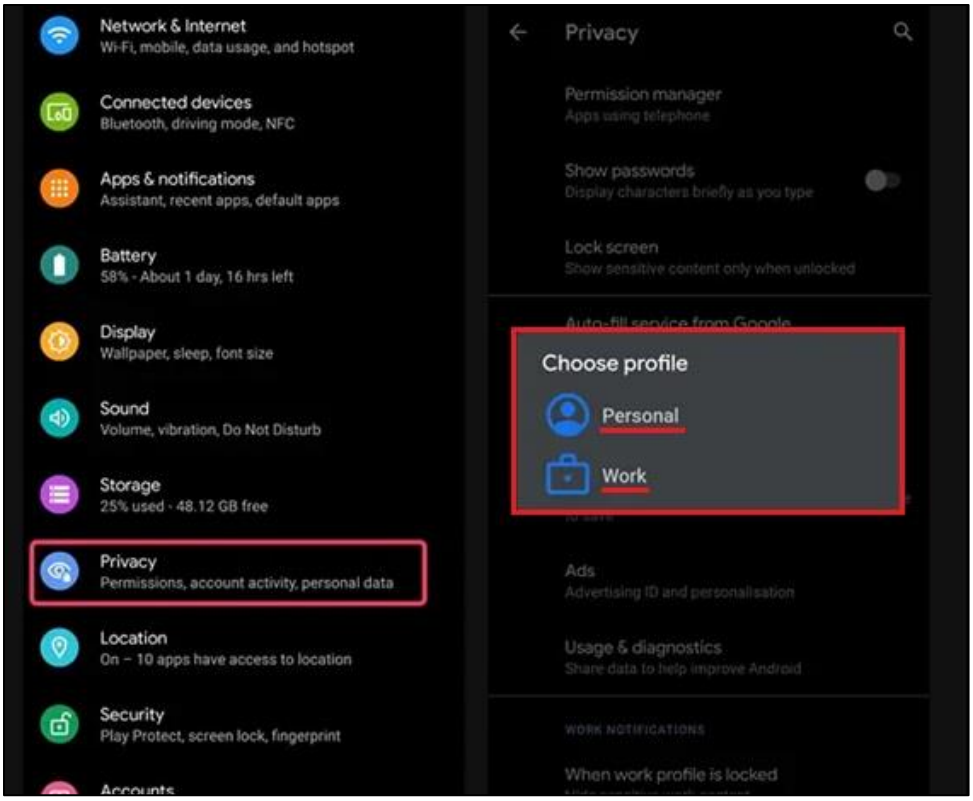
Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

> More generally, for commonly written language varieties such as English (`en`), Russian (`ru`) and Chinese (`zh`), large text corpora can be found easily across many domains. This means that the typing experience upon first use will typically be better than in languages where smaller text corpora are available, with limited domain coverage. As described above, on-device personalization can help improve pre-built generic language models as the keyboard application is used over time, by creating a personal dictionary with out-of-vocabulary words and common phrases. In our user studies, we find that such on-device personalization usually helps improve the typing experience significantly.

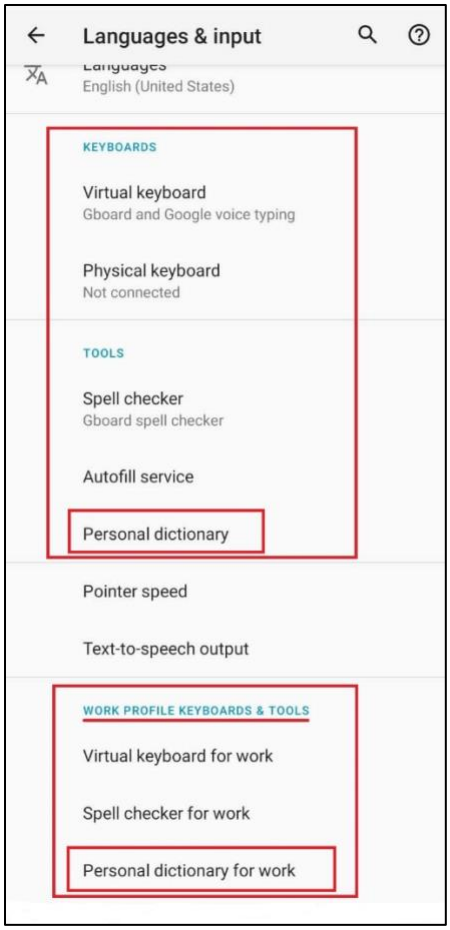
Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 18)

> A **work profile** can be set up on an Android device to separate work apps and data from personal apps and data. With a work profile you can securely and privately use the same device for work and personal purposes—your organization manages your work apps and data while your personal apps, data, and usage remain private.

Source: https://support.google.com/work/android/answer/6191949



Source: https://beebom.com/android-10-simplified-work-profiles/



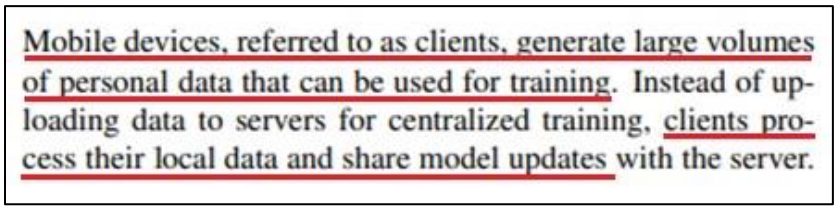
Source: Screenshot taken during testing.

54. The method implemented by the Motorola G Stylus 5G smartphone includes the step of adjusting, via the device, a language model of a local message model, wherein the local message model includes the language model, a handwriting model, and an acoustic model, wherein the adjusting the language model is based at least in part on the selected user message model.

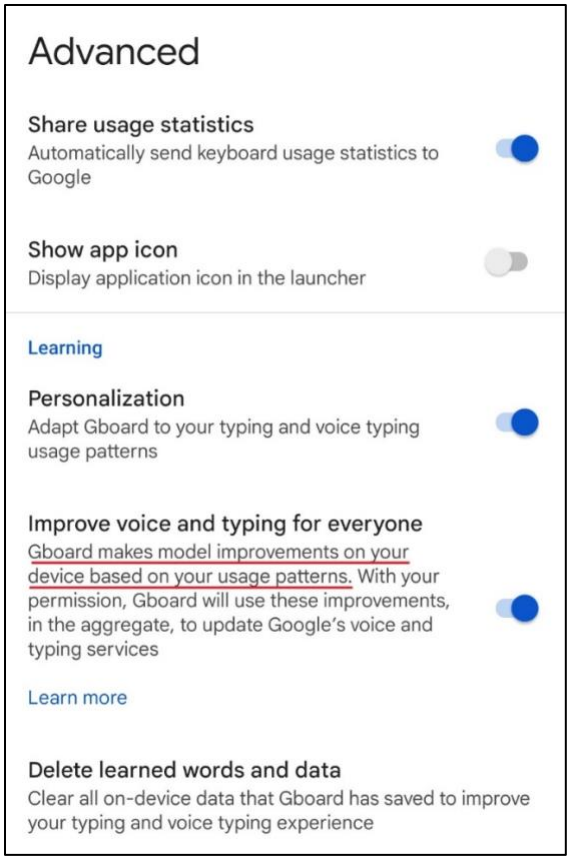
55. For example, the virtual keyboard application on the Motorola G Stylus 5G smartphone provides users with a personalized model to learn the user's preferences. As one example, users can select a profile to be used as a virtual keyboard model from available profiles, such as personal and work profiles ("plurality of user message models"). After selecting a profile, the data input by the user (via the virtual keyboard) is utilized to learn the user's

preferences, train the language model of the selected profile, and update the model locally. The smartphone builds one model for each language variety and uses on-device personalization to improve the language model of a selected profile by learning the user's preferences.

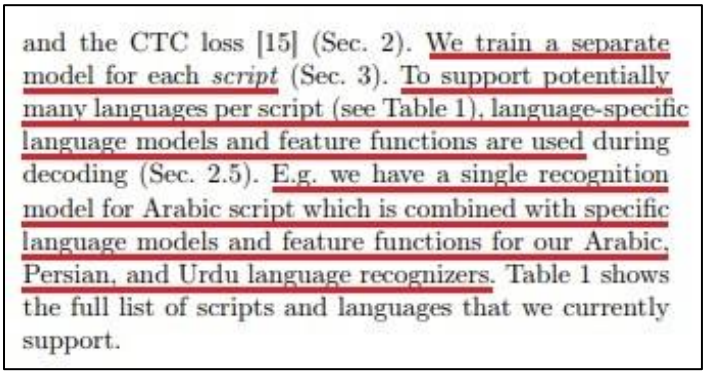
56. As one example, an end-to-end model architecture is used for the handwriting model and acoustic model. End-to-end models contain multiple functioning models tied together to form a single neural network that trains and improves itself. The end-to-end model ("local message model") contains an n-gram language model ("language model") personalized to the user, along with acoustic and handwriting models ("local message model includes the language model, a handwriting model, and an acoustic model"). Based on the input received from the user, the message correction data, and the selected profile of the user ("selected user message model"), the n-gram language model is trained and corrects itself to improve accuracy of prediction ("adjusting the language model").


Mobile devices, referred to as clients, generate large volumes of personal data that can be used for training. Instead of uploading data to servers for centralized training, clients process their local data and share model updates with the server.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 3)



Source: Screenshot taken during testing.



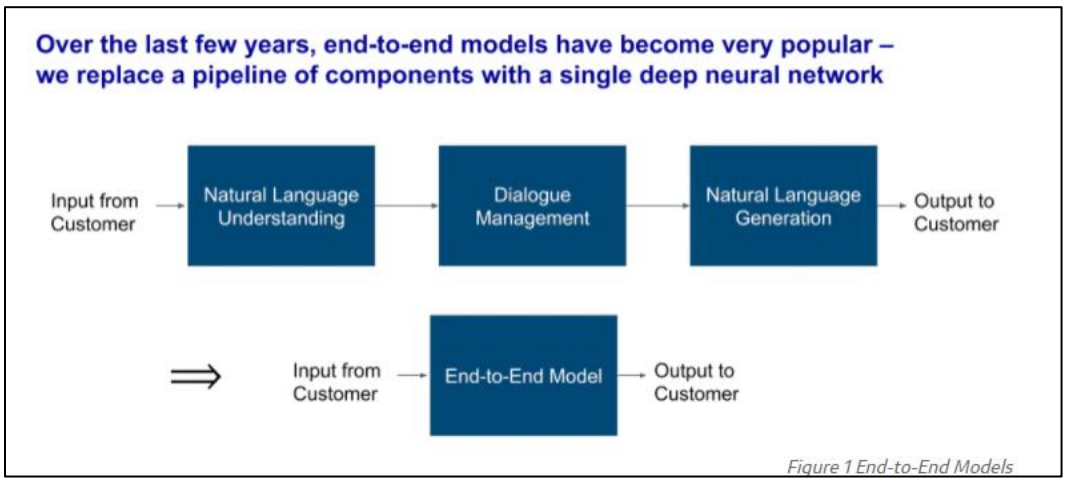
and the CTC loss [15] (Sec. 2). We train a separate model for each *script* (Sec. 3). To support potentially many languages per script (see Table 1), language-specific language models and feature functions are used during decoding (Sec. 2.5). E.g. we have a single recognition model for Arabic script which is combined with specific language models and feature functions for our Arabic, Persian, and Urdu language recognizers. Table 1 shows the full list of scripts and languages that we currently support.

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 2)

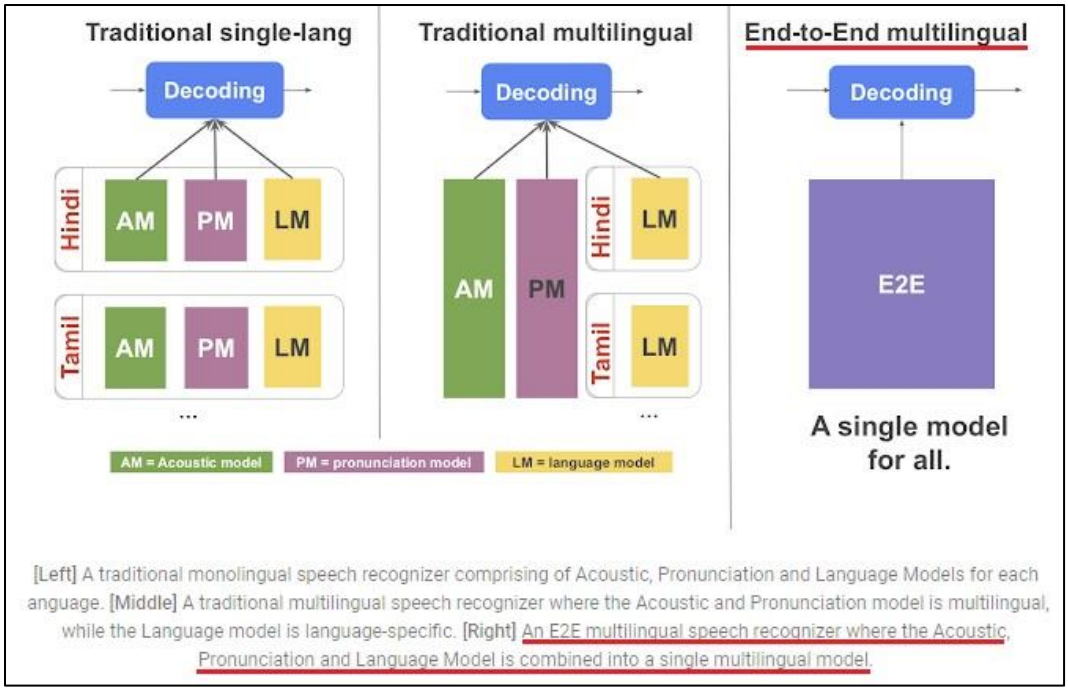
2 End-to-end Model Architecture

Our handwriting recognition model draws its inspiration from research aimed at building end-to-end transcription models in the context of handwriting recognition [15], optical character recognition [8], and acoustic modeling in speech recognition [40]. The model architecture is constructed from common neural network blocks, i.e. bidirectional LSTMs and fully-connected layers (Figure 2). It is trained in an end-to-end manner using the CTC loss [15].

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 3)



Source: https://www.capitalone.com/tech/machine-learning/pros-and-cons-of-end-to-end-models/



[Left] A traditional monolingual speech recognizer comprising of Acoustic, Pronunciation and Language Models for each anguage. [Middle] A traditional multilingual speech recognizer where the Acoustic and Pronunciation model is multilingual, while the Language model is language-specific. [Right] An E2E multilingual speech recognizer where the Acoustic, Pronunciation and Language Model is combined into a single multilingual model.

Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

Supporting features such as auto-correct, next-word prediction (predictive text) and spell-check requires the use of a machine-learning language model, such as n-gram language models, which can be used in a finite-state transduction decoder (Ouyang et al., 2017). These language models can be created based on a variety of textual sources, e.g. web crawls, external text corpora, or even wordlists (to create unigram language models). A detailed description of our standard approach to mining training data for language models across many languages can be found in Prasad et al. (2018). Since the data that we mine can be quite noisy, we apply our scalable automatic data normalization system across all languages and data sets, as described in Chua et al. (2018). Our model training algorithms are described in Allauzen et al. (2016).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 16)

A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

ceding text. The n-best output labels from this state are returned. Paths containing back-off transitions to lower-orders are also considered. The primary (static) language model for the English language in Gboard is a Katz smoothed Bayesian interpolated [4] 5-gram LM containing 1.25 million n-grams, including 164,000 unigrams. Personalized user history, contacts, and email n-gram models augment the primary LM.

Source: https://arxiv.org/pdf/1811.03604.pdf (Page 1)

The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

More generally, for commonly written language varieties such as English (`en`), Russian (`ru`) and Chinese (`zh`), large text corpora can be found easily across many domains. This means that the typing experience upon first use will typically be better than in languages where smaller text corpora are available, with limited domain coverage. As described above, on-device personalization can help improve pre-built generic language models as the keyboard application is used over time, by creating a personal dictionary with out-of-vocabulary words and common phrases. In our user studies, we find that such on-device personalization usually helps improve the typing experience significantly.

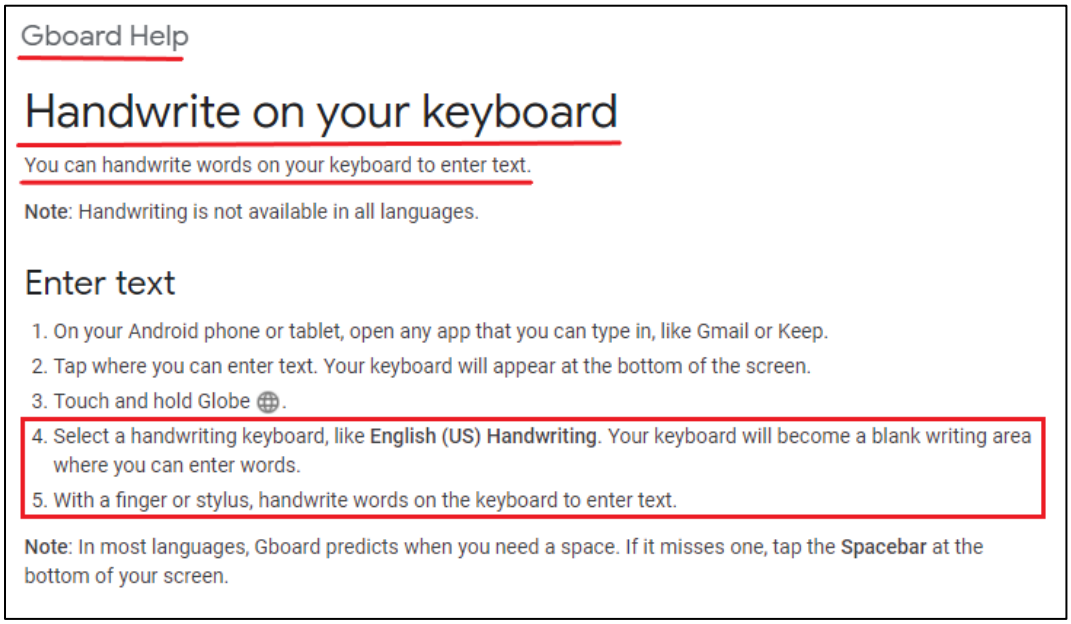
Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 18)

Testers have, of course, also identified areas of improvement: most commonly, users indicate that the dictionary still appears to be relatively small, presumably arising from finding too many correctly spelled words being highlighted as spelling mistakes. This makes sense, given that the training corpora we trained the language models on are typically smaller than the corpora in other languages that our testers may be familiar with. Fortunately, on-device personalization can help address this by learning words over time as the keyboard is used.

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 19)

57. The method implemented by the Motorola G Stylus 5G smartphone includes the step of generating, by the device, converted handwriting text data based at least in part on the freehand input, the adjusted language model, and the handwriting model.

58. For example, the virtual keyboard application on the Motorola G Stylus 5G smartphone provides a feature for converting handwriting to text. It uses an end-to-end model based on the Recurrent Neural Network (RNN), which combines handwriting and language models for handwriting recognition. The language model used by the smartphone is an n-gram language model. In the writing area of the keypad of the device, patterns created by the user are recorded, and the corresponding handwriting information ("freehand input") is converted into text using a handwriting model ("handwriting model"), which is a bi-directional version of the quasi-recurrent neural network (QRNN) model. The user-personalized language model ("adjusted language model") is used to generate text responsive to handwriting input ("converted handwriting text data").



Gboard Help

Handwrite on your keyboard

You can handwrite words on your keyboard to enter text.

Note: Handwriting is not available in all languages.

Enter text

1. On your Android phone or tablet, open any app that you can type in, like Gmail or Keep.
2. Tap where you can enter text. Your keyboard will appear at the bottom of the screen.
3. Touch and hold Globe.
4. Select a handwriting keyboard, like **English (US) Handwriting**. Your keyboard will become a blank writing area where you can enter words.
5. With a finger or stylus, handwrite words on the keyboard to enter text.

Note: In most languages, Gboard predicts when you need a space. If it misses one, tap the **Spacebar** at the bottom of your screen.

Source: https://support.google.com/gboard/answer/9108773?hl=en&ref_topic=9024098

The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

2 End-to-end Model Architecture

Our handwriting recognition model draws its inspiration from research aimed at building end-to-end transcription models in the context of handwriting recognition [15], optical character recognition [8], and acoustic modeling in speech recognition [40]. The model architecture is constructed from common neural network blocks, i.e. bidirectional LSTMs and fully-connected layers (Figure 2). It is trained in an end-to-end manner using the CTC loss [15].

Source: https://arxiv.org/pdf/1902.10525.pdf (Page 3)

RNN-Based Handwriting Recognition in Gboard

Since then, progress in machine learning has enabled new model architectures and training methodologies, allowing us to revise our initial approach (which relied on hand-designed heuristics to cut the handwritten input into single characters) and instead build a single machine learning model that operates on the whole input and reduces error rates substantially compared to the old version. We launched those new models for all latin-script based languages in Gboard at the beginning of the year, and have published the paper "Fast Multi-language LSTM-based Online Handwriting Recognition" that explains in more detail the research behind this release. In this post, we give a high-level overview of that work.

Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

We experimented with multiple types of RNNs, and finally settled on using a bidirectional version of quasi-recurrent neural networks (QRNN). QRNNs alternate between convolutional and recurrent layers, giving it the theoretical potential for efficient parallelization, and provide a good predictive performance while keeping the number of weights comparably small. The number of weights is directly related to the size of the model that needs to be downloaded, so the smaller the better.

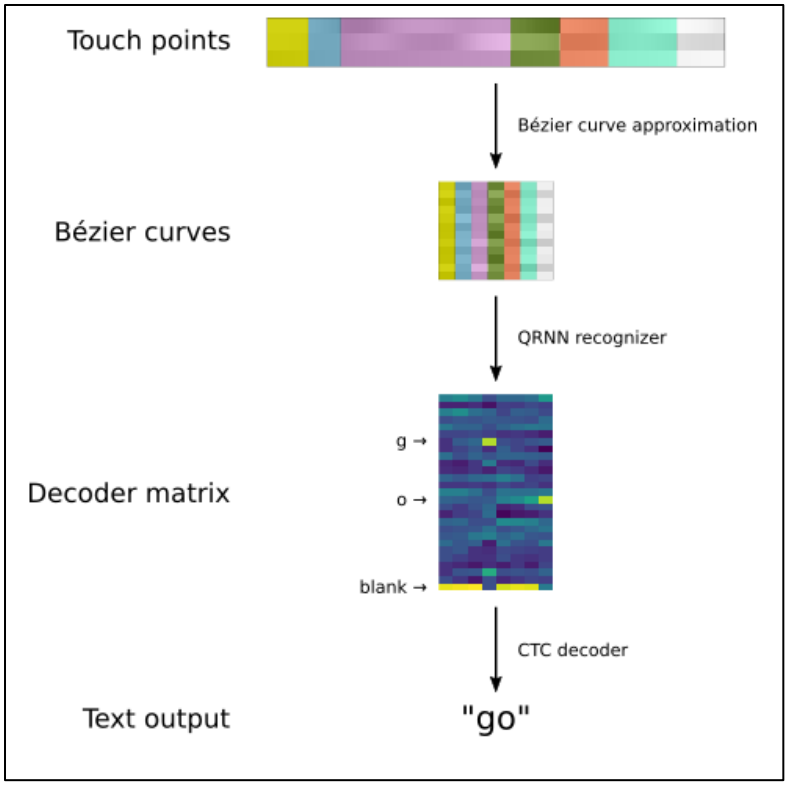
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

In order to "decode" the curves, the recurrent neural network produces a matrix, where each column corresponds to one input curve, and each row corresponds to a letter in the alphabet. The column for a specific curve can be seen as a probability distribution over all the letters of the alphabet. However, each letter can consist of multiple curves (the *g* and *o* above, for instance, consist of four and three curves, respectively). This mismatch between the length of the output sequence from the recurrent neural network (which always matches the number of bezier curves) and the actual number of characters the input is supposed to represent is addressed by adding a special *blank* symbol to indicate no output for a particular curve, as in the Connectionist Temporal Classification (CTC) algorithm. We use a Finite State Machine Decoder to combine the outputs of the Neural Network with a character-based language model encoded as a weighted finite-state acceptor. Character sequences that are common in a language (such as "sch" in German) receive bonuses and are more likely to be output, whereas uncommon sequences are penalized. The process is visualized below.

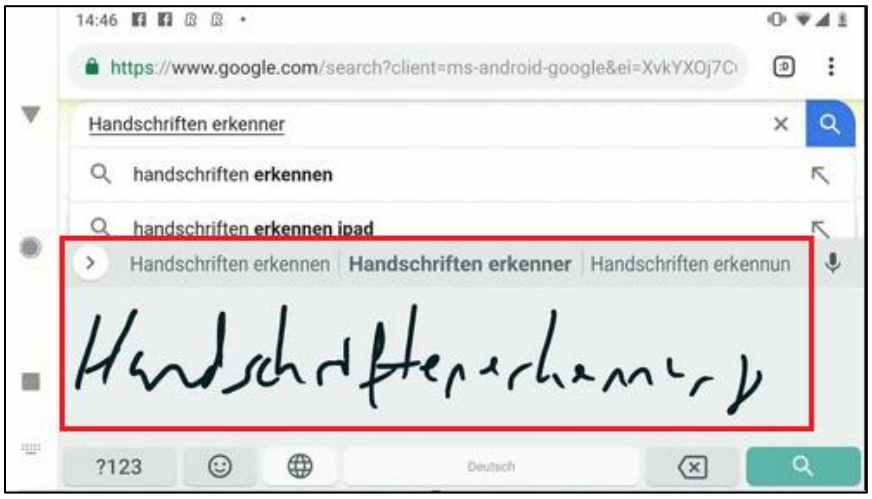
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

Character Language Models: For each language we support, we build a 7-gram language model over Unicode codepoints from a large web-mined text corpus using Stupid back-off [3]. The final files are pruned to 10 million 7-grams each. Compared to our previous system [25], we found that language model size has a smaller impact on the recognition accuracy, which is likely due to the capability of recurrent neural networks to capture dependencies between consecutive characters. We therefore use smaller language models over shorter contexts.

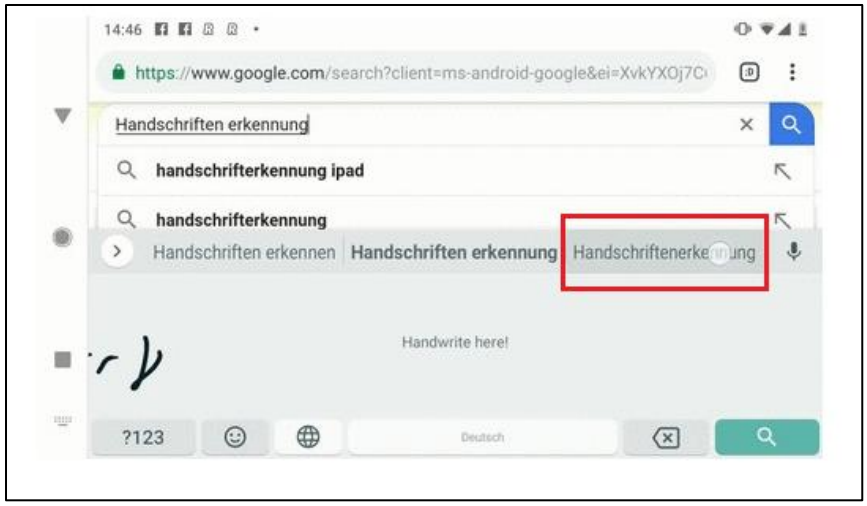
Source: https://arxiv.org/pdf/1902.10525.pdf (Page 6)



Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html



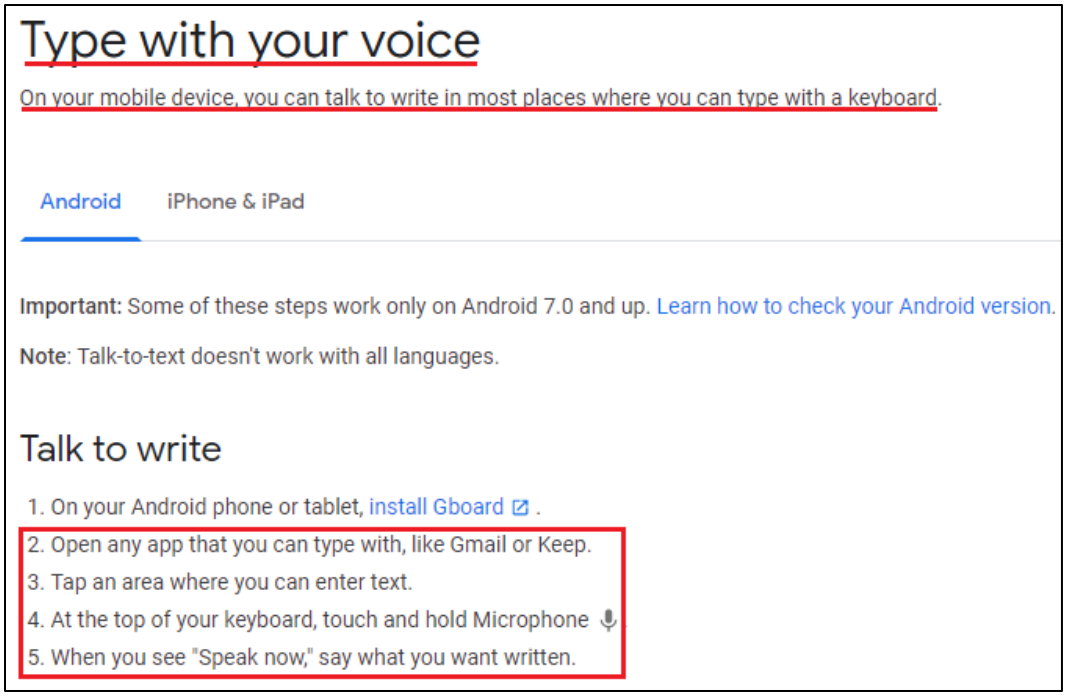
Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html



Source: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html

59. The method implemented by the Motorola G Stylus 5G smartphone includes the step of generating, by the device, converted speech text data based at least in part on the audio input, the adjusted language model, and the acoustic model.

60. For example, the virtual keyboard application on the Motorola G Stylus 5G smartphone uses an on-device, all neural speech recognizer. The speech recognizer uses an end-to-end RNN-T model that combines an acoustic model, pronunciation model, and a language model. The language model is an n-gram language model. The user's voice input ("audio input") is converted into text using the speech recognizer. The pronunciation model, acoustic model ("acoustic model"), and the user-personalized language model ("adjusted language model") can generate text responsive to voice input ("generating, by the device, converted speech text data").



Type with your voice

On your mobile device, you can talk to write in most places where you can type with a keyboard.

Android iPhone & iPad

Important: Some of these steps work only on Android 7.0 and up. Learn how to check your Android version.

Note: Talk-to-text doesn't work with all languages.

Talk to write

1. On your Android phone or tablet, install Gboard .
2. Open any app that you can type with, like Gmail or Keep.
3. Tap an area where you can enter text.
4. At the top of your keyboard, touch and hold Microphone
5. When you see "Speak now," say what you want written.

Source: https://support.google.com/gboard/answer/2781851?hl=en&ref_topic=9024098

The approach we have adopted is to build one model covering all sub-varieties for each language variety, where we tune the auto-correction parameters to be significantly more lenient, and then to rely on on-device personalization to learn the user's individual preferences. Similar approaches can be adopted for many languages the world over, including colloquial Arabic varieties, which also offer a wide range of internal linguistic variation with unclear boundaries between varieties (Abdul-Mageed et al., 2018).

Source: https://arxiv.org/ftp/arxiv/papers/1912/1912.01218.pdf (Page 17)

Today, we're happy to announce the rollout of an end-to-end, all-neural, on-device speech recognizer to power speech input in Gboard. In our recent paper, "Streaming End-to-End Speech Recognition for Mobile Devices", we present a model trained using RNN transducer (RNN-T) technology that is compact enough to reside on a phone. This means no more network latency or spottiness — the new recognizer is always available, even when you are offline. The model works at the character level, so that as you speak, it outputs words character-by-character, just as if someone was typing out what you say in real-time, and exactly as you'd expect from a keyboard dictation system.

Source: https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

A Low-latency All-neural Multilingual Model

Traditional ASR systems contain separate components for acoustic, pronunciation, and language models. While there have been attempts to make some or all of the traditional ASR components multilingual [1,2,3,4], this approach can be complex and difficult to scale. E2E ASR models combine all three components into a single neural network and promise scalability and ease of parameter sharing. Recent works have extended E2E models to be multilingual [1,2], but they did not address the need for real-time speech recognition, a key requirement for applications such as the Assistant, Voice Search and GBoard dictation. For this, we turned to recent research at Google that used a Recurrent Neural Network Transducer (RNN-T) model to achieve streaming E2E ASR. The RNN-T system outputs words one character at a time, just as if someone was typing in real time, however this was not multilingual. We built upon this architecture to develop a low-latency model for *multilingual* speech recognition.

Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

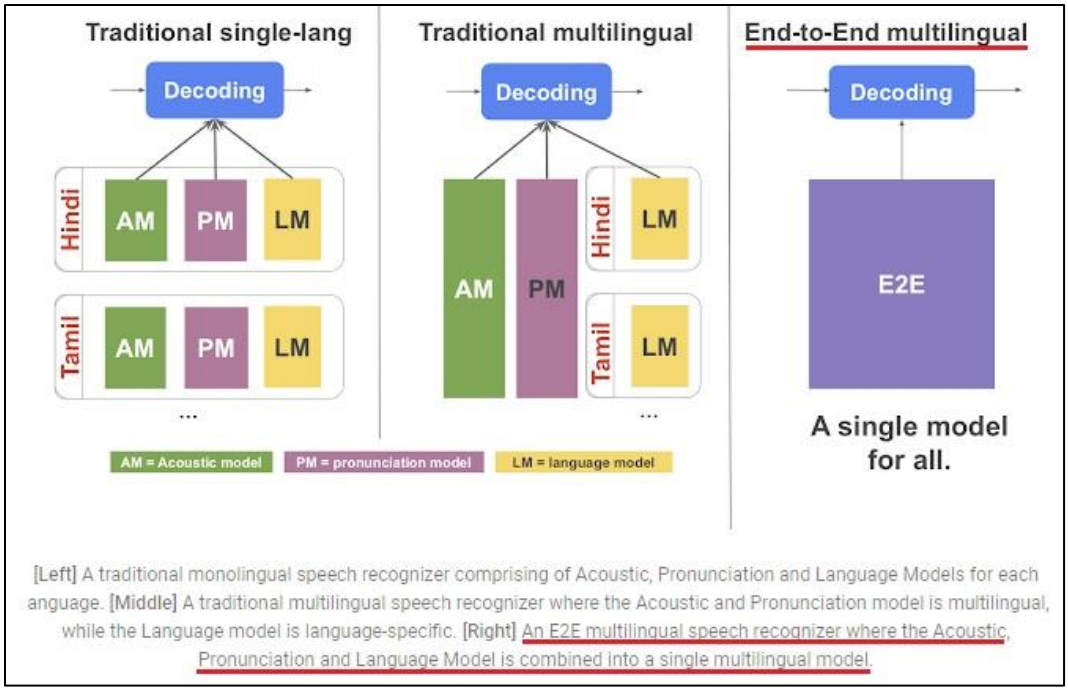
A probabilistic n-gram transducer is used to represent the language model for the keyboard. A state in the model represents an (up to) n-1 word context and an arc leaving that state is labeled with a successor word together with its probability of following that context (estimated from textual data). These, together with the spatial model that gives the likelihoods of sequences of key touches (discrete tap entries or continuous gestures in glide typing), are combined and explored with a beam search.

Source: https://ai.googleblog.com/2017/05/the-machine-intelligence-behind-gboard.html

The language model

Combining the acoustic and pronunciation models, we have audio coming in and words coming out. But that's not quite specific enough to provide reliable Voice Search, because you cannot just string any word together with any other word: there are word combinations that are more reasonable than others. Enter the language model, the third component of the recognition system. It calculates the frequencies of all word sequences between one to five words and thereby constrains the possible word sequences that can be formed out of the two aforementioned models to ones that are sensible combinations in language. The final search algorithm will then pick the valid word sequence that has the highest frequency of occurrence in the language.

Source: https://careers.google.com/stories/how-one-team-turned-the-dream-of-speech-recognition-into-a-reality/



Source: https://ai.googleblog.com/2019/09/large-scale-multilingual-speech.html

61. Motorola has had knowledge of the '737 Patent at least as of the date when it was notified of the filing of this action, and as early as July 27, 2022, when Motorola received a letter notifying it of the '737 Patent.

62. Buffalo Patents has been damaged as a result of the infringing conduct by Motorola alleged above. Thus, Motorola is liable to Buffalo Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

63. Buffalo Patents has neither made nor sold unmarked articles that practice the '737 Patent, and is entitled to collect pre-filing damages for the full period allowed by law for infringement of the '737 Patent.

ADDITIONAL ALLEGATIONS REGARDING INFRINGEMENT AND PERSONAL JURISDICTION

64. Motorola has also indirectly infringed the ’737 Patent by inducing others to directly infringe the ’737 Patent.

65. Motorola has induced the end users and/or Motorola’s customers to directly infringe (literally and/or under the doctrine of equivalents) the ’737 Patent by using the accused products.

66. Motorola took active steps, directly and/or through contractual relationships with others, with the specific intent to cause them to use the accused products in a manner that infringes one or more claims of the ’737 Patent, including, for example, Claims 1 and 13 of the ’737 Patent.

67. Such steps by Motorola included, among other things, advising or directing customers, end users, and others (including third party testing and certification organizations) to use the accused products in an infringing manner; advertising and promoting the use of the accused products in an infringing manner; and/or distributing instructions that guide users to use the accused products in an infringing manner.

68. Motorola performed these steps, which constitute joint and/or induced infringement, with the knowledge of the ’737 Patent, and with the knowledge that the induced acts constitute infringement.

69. Motorola was and is aware that the normal and customary use of the accused products by Motorola’s customers would infringe the ’737 Patent. Motorola’s inducement is ongoing.

70. Motorola has also induced its affiliates, or third party manufacturers, shippers, distributors, retailers, or other persons acting on its or its affiliates’ behalf, to directly infringe

(literally and/or under the doctrine of equivalents) the ’737 Patent by importing, selling or offering to sell the accused products.

71. Motorola has a significant role in placing the accused products in the stream of commerce with the expectation and knowledge that they will be purchased by consumers in Illinois and elsewhere in the United States.

72. Motorola purposefully directs or controls the making of accused products and their shipment to the United States, using established distribution channels, for sale in Illinois and elsewhere within the United States.

73. Motorola purposefully directs or controls the sale of the accused products into established United States distribution channels, including sales to nationwide retailers. Motorola’s established United States distribution channels include one or more United States based affiliates.

74. Motorola purposefully directs or controls the sale of the accused products online and in nationwide retailers such as Walmart and Best Buy, including for sale in Illinois and elsewhere in the United States, and expects and intends that the accused products will be so sold.

75. Motorola purposefully places the accused products—whether by itself or through subsidiaries, affiliates, or third parties—into an international supply chain, knowing that the accused products will be sold in the United States, including Illinois. Therefore, Motorola also facilitates the sale of the accused products in Illinois.

76. Motorola took active steps, directly and/or through contractual relationships with others, with the specific intent to cause such persons to import, sell, or offer to sell the accused products in a manner that infringes one or more claims of the ’737 Patent.

77. Such steps by Motorola included, among other things, making or selling the accused products outside of the United States for importation into or sale in the United States, or knowing that such importation or sale would occur; and directing, facilitating, or influencing its affiliates, or third party manufacturers, shippers, distributors, retailers, or other persons acting on its or its affiliates' behalf, to import, sell, or offer to sell the accused products in an infringing manner.

78. Motorola performed these steps, which constitute induced infringement, with the knowledge of the '737 Patent, and with the knowledge that the induced acts would constitute infringement.

79. Motorola performed such steps in order to profit from the eventual sale of the accused products in the United States.

80. Motorola's inducement is ongoing.

81. Motorola has also indirectly infringed by contributing to the infringement of the '737 Patent. Motorola has contributed to the direct infringement of the '737 Patent by the end user of the accused products.

82. The accused products have special features that are specially designed to be used in an infringing way and that have no substantial uses other than ones that infringe the '737 Patent, including, for example, Claims 1 and 13 of the '737 Patent.

83. The special features include, for example, hardware and software components implementing a virtual keyboard interface that is capable of receiving freehand input using a stylus and audio input from a user, and capable of generating text data using multiple language models based, in part, on an adjusted language model, used in a manner that infringes the'737 Patent.

84. These special features constitute a material part of the invention of one or more of the claims of the ’737 Patent, and are not staple articles of commerce suitable for substantial non-infringing use.

85. Motorola’s contributory infringement is ongoing.

86. Motorola has had actual knowledge of the ’737 Patent at least as early as July 27, 2022, when Motorola received a letter notifying it of the ’737 Patent, and/or as of the date when it was notified of the filing of this action. Since at least that time, Motorola has known the scope of the claims of the ’737 Patent; the products that practice the ’737 Patent; and that Buffalo Patents is the owner of the ’737 Patent.

87. By the time of trial, Motorola will have known and intended (since receiving such notice) that its continued actions would infringe and actively induce and contribute to the infringement of one or more claims of the ’737 Patent.

88. Furthermore, Motorola has a policy or practice of not reviewing the patents of others (including instructing its employees to not review the patents of others), and thus has been willfully blind of Buffalo Patents’ patent rights. *See, e.g.*, M. Lemley, “Ignoring Patents,” 2008 Mich. St. L. Rev. 19 (2008).

89. Motorola’s actions are at least objectively reckless as to the risk of infringing valid patents, and this objective risk was either known or should have been known by Motorola. Motorola has knowledge of the ’737 Patent.

90. Motorola’s customers have infringed the ’737 Patent. Motorola encouraged its customers’ infringement.

91. Motorola’s direct and indirect infringement of the ’737 Patent, and its direct infringement of the ’405 Patent and the ’737 Patent has been, and/or continues to be willful,

intentional, deliberate, and/or in conscious disregard of Buffalo Patents' rights under the patents-in-suit.

92. Buffalo Patents has been damaged as a result of Motorola's infringing conduct alleged above. Thus, Motorola is liable to Buffalo Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

JURY DEMAND

Buffalo Patents hereby requests a trial by jury on all issues so triable by right.

PRAYER FOR RELIEF

Buffalo Patents requests that the Court find in its favor and against Motorola, and that the Court grant Buffalo Patents the following relief:

a. Judgment that one or more claims of the '405 Patent and the '737 Patent have been infringed, either literally and/or under the doctrine of equivalents, by Motorola and/or all others acting in concert therewith;

b. A permanent injunction enjoining Motorola and its officers, directors, agents, servants, affiliates, employees, divisions, branches, subsidiaries, parents, and all others acting in concert therewith from infringement of the '737 Patent; or, in the alternative, an award of a reasonable ongoing royalty for future infringement of the '737 Patent by such entities;

c. Judgment that Motorola account for and pay to Buffalo Patents all damages to and costs incurred by Buffalo Patents because of Motorola's infringing activities and other conduct complained of herein, including an award of all increased damages to which Buffalo Patents is entitled under 35 U.S.C. § 284;

d. That Buffalo Patents be granted pre-judgment and post-judgment interest on the damages caused by Motorola's infringing activities and other conduct complained of herein;

e. That this Court declare this an exceptional case and award Buffalo Patents its reasonable attorney's fees and costs in accordance with 35 U.S.C. § 285; and

f. That Buffalo Patents be granted such other and further relief as the Court may deem just and proper under the circumstances.

Dated: August 25, 2022

Respectfully submitted,

/s/ Timothy J. Haller

Timothy J. Haller (Local Counsel)
HALLER LAW PLLC
230 E Delaware Pl, Ste 5E
Chicago, IL 60611
Phone: (630) 336-4283
haller@haller-iplaw.com

Matthew J. Antonelli (*Pro Hac Vice Pending*)
Zachariah S. Harrington (*Pro Hac Vice Pending*)
Larry D. Thompson, Jr. (*Pro Hac Vice Pending*)
Rehan M. Safiullah (*Pro Hac Vice Pending*)
ANTONELLI, HARRINGTON & THOMPSON LLP
4306 Yoakum Blvd., Ste. 450
Houston, TX 77006
(713) 581-3000
matt@ahtlawfirm.com
zac@ahtlawfirm.com
larry@ahtlawfirm.com
rehan@ahtlawfirm.com

Attorneys for Plaintiff
Buffalo Patents, LLC