Google AI Blog

The latest from Google Research

Robust Neural Machine Translation

Monday, July 29, 2019

Posted by Yong Cheng, Software Engineer, Google Research

In recent years, neural machine translation (NMT) using Transformer models has experienced tremendous success. Based on deep neural networks, NMT models are usually trained end-to-end on very large parallel corpora (input/output text pairs) in an entirely data-driven fashion and without the need to impose explicit rules of language.

Despite this huge success, NMT models can be sensitive to minor perturbations of the input, which can manifest as a variety of different errors, such as under-translation, over-translation or mistranslation. For example, given a German sentence, the state-of-the-art NMT model, Transformer, will yield a correct translation.

> *"Der Sprecher des Untersuchungsausschusses hat angekündigt, vor Gericht zu ziehen, falls sich die geladenen Zeugen weiterhin weigern sollten, eine Aussage zu machen."*
>
> (Machine translation to English: *"The spokesman of the Committee of Inquiry has announced that if the witnesses summoned continue to refuse to testify, he will be brought to court."*),

But, when we apply a subtle change to the input sentence, say from *geladenen* to the synonym *vorgeladenen*, the translation becomes very different (and in this case, incorrect):

> *"Der Sprecher des Untersuchungsausschusses hat angekündigt, vor Gericht zu ziehen, falls sich die vorgeladenen Zeugen weiterhin weigern sollten, eine Aussage zu machen."*
>
> (Machine translation to English: *"The investigative committee has announced that he will be brought to justice if the witnesses who have been invited continue to refuse to testify."*).

This lack of robustness in NMT models prevents many commercial systems from being applicable to tasks that cannot tolerate this level of instability. Therefore, learning robust

translation models is not just desirable, but is often required in many scenarios. Yet, while the robustness of neural networks has been extensively studied in the computer vision community, only a few prior studies on learning robust NMT models can be found in literature.

In "Robust Neural Machine Translation with Doubly Adversarial Inputs" (to appear at ACL 2019), we propose an approach that uses generated adversarial examples to improve the stability of machine translation models against small perturbations in the input. We learn a robust NMT model to directly overcome adversarial examples generated with knowledge of the model and with the intent of distorting the model predictions. We show that this approach improves the performance of the NMT model on standard benchmarks.

Training a Model with AdvGen

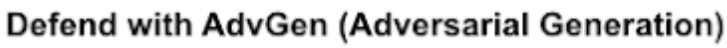
An ideal NMT model would generate similar translations for separate inputs that exhibit small differences. The idea behind our approach is to perturb a translation model with adversarial inputs in the hope of improving the model's robustness. It does this using an algorithm called Adversarial Generation (AdvGen**),** which generates plausible adversarial examples for perturbing the model and then feeds them back into the model for defensive training. While this method is inspired by the idea of generative adversarial networks (GANs), it does not rely on a discriminator network, but simply applies the adversarial example in training, effectively diversifying and extending the training set.

The first step is to perturb the model using AdvGen. We start by using Transformer to calculate the translation loss based on a source input sentence, a target input sentence and a target output sentence. Then AdvGen randomly selects some words in the source sentence, assuming a uniform distribution. Each word has an associated list of similar words, i.e., candidates that can be used for substitution, from which AdvGen selects the word that is most likely to introduce errors in Transformer output. Then, this generated adversarial sentence is fed back into Transformer, initiating the defense stage.

First, the Transformer model is applied to an input sentence (*lower left*) and, in conjunction with the target output

sentence (*above right*) and target input sentence (*middle right;* beginning with the placeholder "<sos>"), the translation loss is calculated. The AdvGen function then takes the source sentence, word selection distribution, word candidates, and the translation loss as inputs to construct an adversarial source example.

During the defend stage, the adversarial sentence is fed back into the Transformer model. Again the translation loss is calculated, but this time using the adversarial source input. Using the same method as above, AdvGen uses the target input sentence, word replacement candidates, the word selection distribution calculated by the attention matrix, and the translation loss to construct an adversarial *target* example.

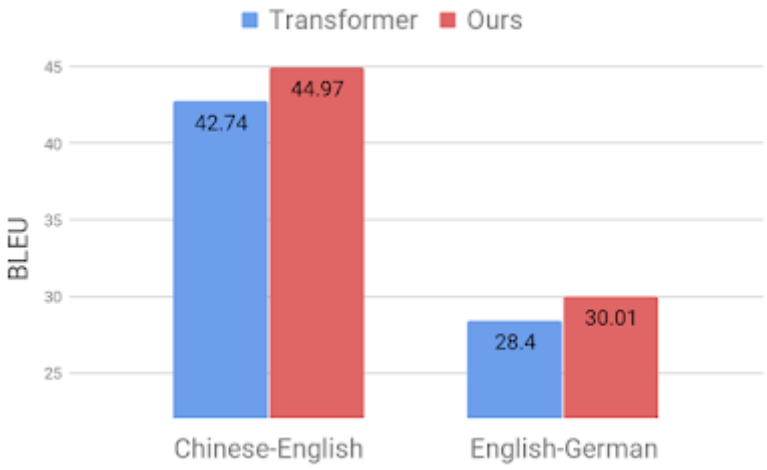


In the defense stage, the adversarial source example serves as input to the Transformer model, and the translation loss is calculated. AdvGen then uses the same method as above to generate an adversarial target example from the target input.

Finally, the adversarial sentence is fed back into Transformer and the robustness loss using the adversarial source example, the adversarial target input example and the target sentence is calculated. If the perturbation led to a significant loss, the loss is minimized so that when the model is confronted with similar perturbations, it will not repeat the same mistake. On the other hand, if the perturbation leads to a low loss, nothing happens, indicating that the model can already handle this perturbation.

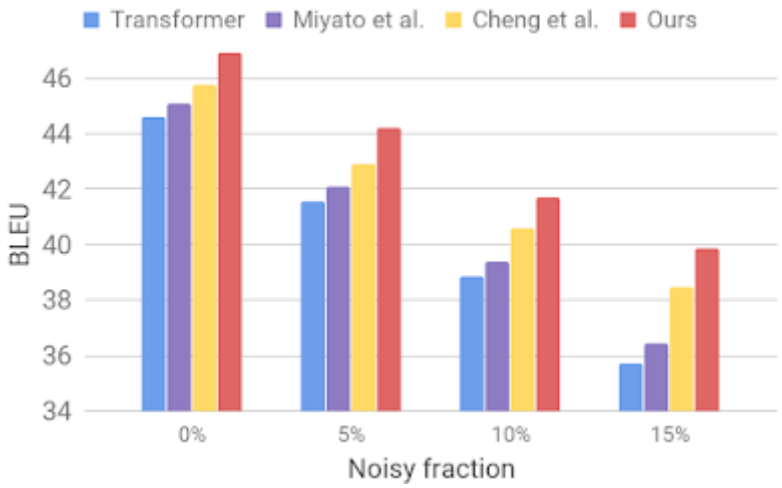
Model Performance

We demonstrate the effectiveness of our approach by applying it to the standard Chinese-English and English-German translation benchmarks. We observed a notable improvement of 2.8 and 1.6 BLEU points, respectively, compared to the competitive Transformer model, achieving a new state-of-the-art performance.



Comparison of Transformer model (Vaswani et al., 2017) on standard benchmarks.

We then evaluate our model on a noisy dataset, generated using a procedure similar to that described for AdvGen. We take an input clean dataset, such as that used on standard translation benchmarks, and randomly select words for similar word substitution. We find that our model exhibits improved robustness compared to other recent models.



Comparison of Transformer, Miyao et al. and Cheng et al. on artificial noisy inputs.

These results show that our method is able to overcome small perturbations in the input sentence and improve the generalization performance. It outperforms competitive translation models and achieves state-of-the-art translation performance on standard benchmarks. We hope our translation model will serve as a robust building block for improving many downstream tasks, especially when those are sensitive or intolerant to imperfect translation input.

Acknowledgements
This research was conducted by Yong Cheng, Lu Jiang and Wolfgang Macherey. Additional thanks go to our leadership Andrew Moore and Julia (Wenli) Zhu.

Accelerating Deep Learning Research with the

Accelerating Deep Learning Research with the Tensor2Tensor Library

Monday, June 19, 2017

Posted by Łukasz Kaiser, Senior Research Scientist, Google Brain Team

Deep Learning (DL) has enabled the rapid advancement of many useful technologies, such as machine translation, speech recognition and object detection. In the research community, one can find code open-sourced by the authors to help in replicating their results and further advancing deep learning. However, most of these DL systems use unique setups that require significant engineering effort and may only work for a specific problem or architecture, making it hard to run new experiments and compare the results.

Today, we are happy to release Tensor2Tensor (T2T), an open-source system for training deep learning models in TensorFlow. T2T facilitates the creation of state-of-the art models for a wide variety of ML applications, such as translation, parsing, image captioning and more, enabling the exploration of various ideas much faster than previously possible. This release also includes a library of datasets and models, including the best models from a few recent papers (*Attention Is All You Need*, *Depthwise Separable Convolutions for Neural Machine Translation* and *One Model to Learn Them All*) to help kick-start your own DL research.

Translation Model	Training time	BLEU (difference from baseline)
Transformer (T2T)	3 days on 8 GPU	**28.4 (+7.8)**
SliceNet (T2T)	6 days on 32 GPUs	26.1 (+5.5)
GNMT + Mixture of Experts	1 day on 64 GPUs	26.0 (+5.4)
ConvS2S	18 days on 1 GPU	25.1 (+4.5)
GNMT	1 day on 96 GPUs	24.6 (+4.0)
ByteNet	8 days on 32 GPUs	23.8 (+3.2)
MOSES (phrase-based baseline)	N/A	20.6 (+0.0)

BLEU scores (higher is better) on the standard WMT English-German translation task.

As an example of the kind of improvements T2T can offer, we applied the library to machine translation. As you can see in the table above, two different T2T models, SliceNet and Transformer, outperform the previous state-of-the-art, GNMT+MoE. Our best T2T model, Transformer, is 3.8 points better than the standard GNMT model, which itself was 4 points above the baseline phrase-based translation system, MOSES. Notably, with T2T you can approach previous state-of-the-art results with a single GPU in one day: a small Transformer model (not shown above) gets 24.9 BLEU after 1 day of training on a single GPU. Now everyone with a GPU can tinker with great translation models on their own: our github repo has instructions on how to do that.

Modular Multi-Task Training

The T2T library is built with familiar TensorFlow tools and defines multiple pieces needed in a deep learning system: data-sets, model architectures, optimizers, learning rate decay schemes, hyperparameters, and so on. Crucially, it enforces a standard interface between all these parts and implements current ML best practices. So you can pick any data-set, model, optimizer and set of hyperparameters, and run the training to check how it performs. We made the architecture modular, so every piece between the input data and the predicted output is a tensor-to-tensor function. If you have a new idea for the model architecture, you don't need to replace the whole setup. You can keep the embedding part and the loss and everything else, just

replace the whole setup. You can keep the embedding part and the loss and everything else, just replace the model body by your own function that takes a tensor as input and returns a tensor.

This means that T2T is flexible, with training no longer pinned to a specific model or dataset. It is so easy that even architectures like the famous LSTM sequence-to-sequence model can be defined in a few dozen lines of code. One can also train a single model on multiple tasks from different domains. Taken to the limit, you can even train a single model on all data-sets concurrently, and we are happy to report that our MultiModel, trained like this and included in T2T, yields good results on many tasks even when training jointly on ImageNet (image classification), MS COCO (image captioning), WSJ (speech recognition), WMT (translation) and the Penn Treebank parsing corpus. It is the first time a single model has been demonstrated to be able to perform all these tasks at once.

Built-in Best Practices

With this initial release, we also provide scripts to generate a number of data-sets widely used in the research community[1], a handful of models[2], a number of hyperparameter configurations, and a well-performing implementation of other important tricks of the trade. While it is hard to list them all, if you decide to run your model with T2T you'll get for free the correct padding of sequences and the corresponding cross-entropy loss, well-tuned parameters for the Adam optimizer, adaptive batching, synchronous distributed training, well-tuned data augmentation for images, label smoothing, and a number of hyper-parameter configurations that worked very well for us, including the ones mentioned above that achieve the state-of-the-art results on translation and may help you get good results too.

As an example, consider the task of parsing English sentences into their grammatical constituency trees. This problem has been studied for decades and competitive methods were developed with a lot of effort. It can be presented as a sequence-to-sequence problem and be solved with neural networks, but it used to require a lot of tuning. With T2T, it took us only a few days to add the parsing data-set generator and adjust our attention transformer model to train on this problem. To our pleasant surprise, we got very good results in only a week:

Parsing Model	F1 score (higher is better)
Transformer (T2T)	91.3
Dyer et al.	**91.7**
Zhu et al.	90.4
Socher et al.	90.4
Vinyals & Kaiser et al.	88.3

Parsing F1 scores on the standard test set, section 23 of the WSJ. We only compare here models trained discriminatively on the Penn Treebank WSJ training set, see the paper for more results.

Contribute to Tensor2Tensor

In addition to exploring existing models and data-sets, you can easily define your own model and add your own data-sets to Tensor2Tensor. We believe the already included models will perform very well for many NLP tasks, so just adding your data-set might lead to interesting results. By making T2T modular, we also make it very easy to contribute your own model and see how it performs on various tasks. In this way the whole community can benefit from a library of baselines and deep learning research can accelerate. So head to our github repository, try the new models, and contribute your own!

Acknowledgements
The release of Tensor2Tensor was only possible thanks to the widespread collaboration of many engineers and researchers. We want to acknowledge here the core team who contributed (in alphabetical order): *Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, Jakob Uszkoreit, Ashish Vaswani.*

---

1 We include a number of datasets for image classification (MNIST, CIFAR-10, CIFAR-100, ImageNet), image captioning (MS COCO), translation (WMT with multiple languages including English-German and English-French), language modelling (LM1B), parsing (Penn Treebank), natural language inference (SNLI), speech recognition (TIMIT), algorithmic problems (over a dozen tasks from reversing through addition and multiplication to algebra) and we will be adding more and welcome your data-sets too.↩

2 Including LSTM sequence-to-sequence RNNs, convolutional networks also with separable convolutions (e.g., Xception), recently researched models like ByteNet or the Neural GPU, and our new state-of-the-art models mentioned in this post that we will be actively updating in the repository.↩

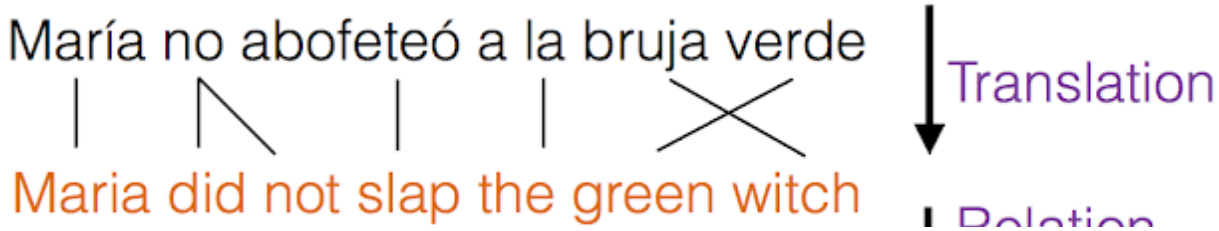
A Multilingual Corpus of Automatically Extracted Relations from Wikipedia

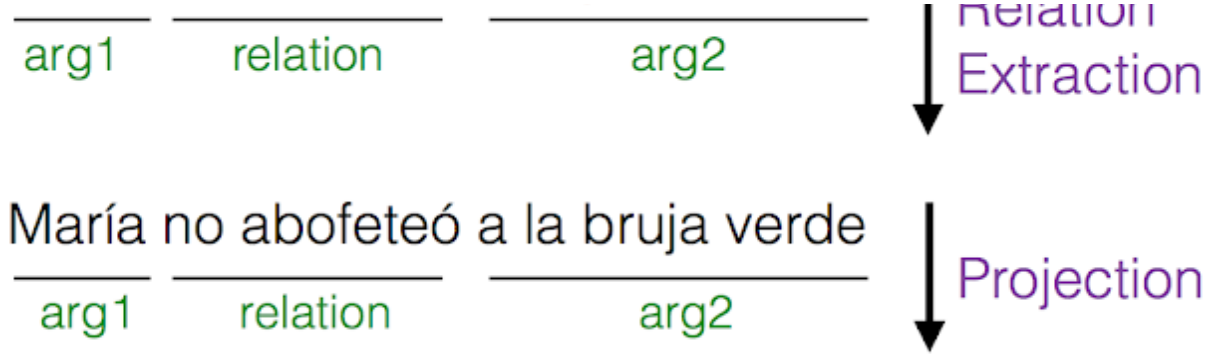
Tuesday, June 2, 2015

Posted by Shankar Kumar, Google Research Scientist and Manaal Faruqui, Carnegie Mellon University PhD candidate

In Natural Language Processing, relation extraction is the task of assigning a semantic relationship between a pair of arguments. As an example, a relationship between the phrases "*Ottawa*" and "*Canada*" is "*is the capital of*". These extracted relations could be used in a variety of applications ranging from Question Answering to building databases from unstructured text.

While relation extraction systems work accurately for English and a few other languages, where tools for syntactic analysis such as parsers, part-of-speech taggers and named entity analyzers are readily available, there is relatively little work in developing such systems for most of the world's languages where linguistic analysis tools do not yet exist. Fortunately, because we do have translation systems between English and many other languages (such as Google Translate), we can translate text from a non-English language to English, perform relation extraction and project these relations back to the foreign language.





Relation extraction in a Spanish sentence using the cross-lingual relation extraction pipeline.

In Multilingual Open Relation Extraction Using Cross-lingual Projection, that will appear at the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT 2015), we use this idea of cross-lingual projection to develop an algorithm that extracts open-domain relation tuples, i.e. where an arbitrary phrase can describe the relation between the arguments, in multiple languages from Wikipedia. In this work, we also evaluated the performance of extracted relations using human annotations in French, Hindi and Russian.

Since there is no such publicly available corpus of multilingual relations, we are releasing a dataset of automatically extracted relations from the Wikipedia corpus in 61 languages, along with the manually annotated relations in 3 languages (French, Hindi and Russian). It is our hope that our data will help researchers working on natural language processing and encourage novel applications in a wide variety of languages.

We wish to thank Bruno Cartoni, Vitaly Nikolaev, Hidetoshi Shimokawa, Kishore Papineni, John Giannandrea and their teams for making this data release possible. This dataset is licensed by Google Inc. under the Creative Commons Attribution-ShareAlike 3.0 License.

Google Handwriting Input in 82 languages on your Android mobile device

Wednesday, April 15, 2015

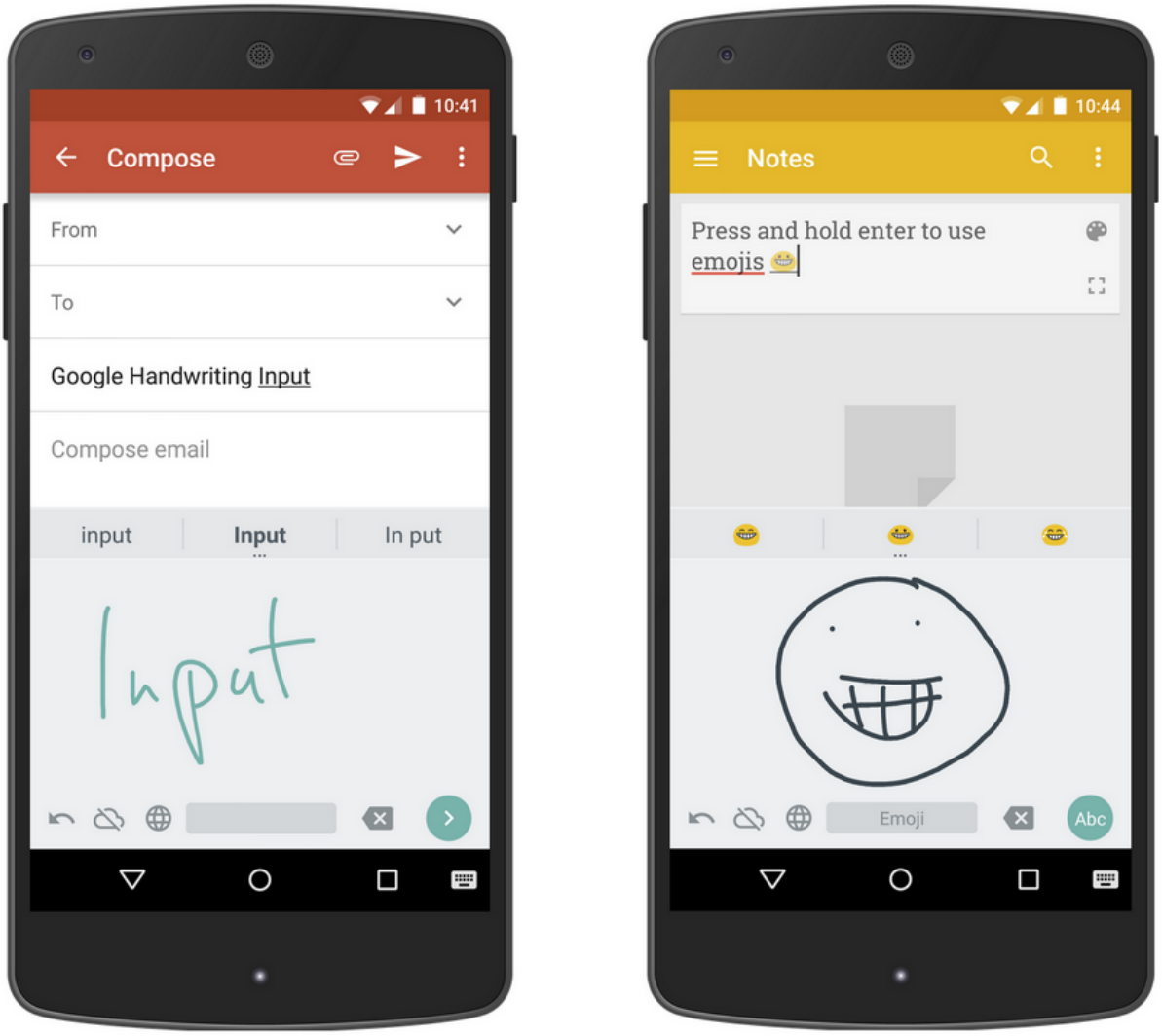
Posted by Thomas Deselaers, Daniel Keysers, Henry Rowley, Li-Lun Wang, Victor Cărbune, Ashok Popat, Dhyanesh Narayanan, Handwriting Team, Google Research

*(**Update 2020-12-10:** In 2019 we launched RNN-based handwriting recognition in Gboard. Moving forward, we will continue to focus our development efforts on Gboard, and have removed the Google Handwriting Input app from the Google Play store. We encourage you to download Gboard instead, and follow these instructions to enable handwriting input.)*

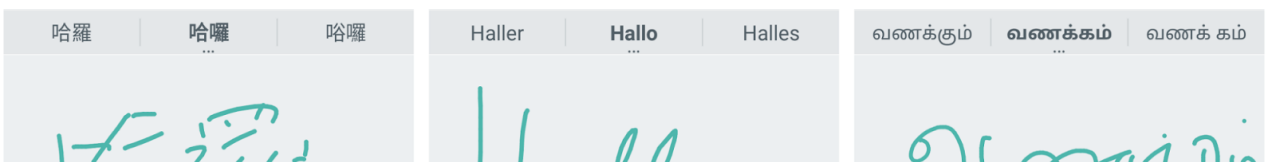
Entering text on mobile devices is still considered inconvenient by many; touchscreen keyboards, although much improved over the years, require a lot of attention to hit the right buttons. Voice input is an option, but there are situations where it is not feasible, such as in a noisy environment or during a meeting. Using handwriting as an input method can allow for

natural and intuitive input method for text entry which complements typing and speech input methods. However, until recently there have been many languages where enabling this functionality presented significant challenges.

Today we launched Google Handwriting Input, which lets users handwrite text on their Android mobile device as an additional input method for *any* Android app. Google Handwriting Input supports 82 languages in 20 distinct scripts, and works with both printed and cursive writing input with or without a stylus. Beyond text input, it also provides a fun way to enter hundreds of emojis by drawing them (simply press and hold the 'enter' button to switch modes). Google Handwriting Input works with or without an Internet connection.



By building on large-scale language modeling, robust multi-language OCR, and incorporating large-scale neural-networks and approximate nearest neighbor search for character classification, Google Handwriting Input supports languages that can be challenging to type on a virtual keyboard. For example, keyboards for ideographic languages (such as Chinese) are often based on a particular dialect of the language, but if a user does not know that dialect, they may be hard to use. Additionally, keyboards for complex script languages (like many South Asian languages) are less standardized and may be unfamiliar. Even for languages where virtual keyboards are more widely used (like English or Spanish), some users find that handwriting is more intuitive, faster, and generally more comfortable.



Writing 'Hello' in Chinese, German, and Tamil.

Google Handwriting Input is the result of many years of research at Google. Initially, cloud based handwriting recognition supported the Translate Apps on Android and iOS, Mobile Search, and Google Input Tools (in Chrome, ChromeOS, Gmail and Docs, translate.google.com, and the Docs symbol picker). However, other products required recognizers to run directly on an Android device without an Internet connection. So we worked to make recognition models smaller and faster for use in Android handwriting input methods for Simplified and Traditional Chinese, Cantonese, and Hindi, as well as multi-language support in Gesture Search. Google Handwriting Input combines these efforts, allowing recognition both on-device and in the cloud (by tapping on the cloud icon) in any Android app.

You can install Google Handwriting Input from the Play Store here. More information and FAQs can be found here.

Making Blockly Universally Accessible

Tuesday, April 1, 2014

Posted by Neil Fraser, Chief Interplanetary Liaison

We work hard to make our products accessible to people everywhere, in every culture. Today we're expanding our outreach efforts to support a traditionally underserved community -- those who call themselves "tlhIngan."

Google's Blockly programming environment is used in K-12 classrooms around the world to teach programming. But the world is not enough. Students on Qo'noS have had difficulty learning to code because most of the teaching tools aren't available in their native language. Additionally, many existing tools are too fragile for their pedagogical approach. As a result, Klingons have found it challenging to enter computer science. This is reflected in the fact that less than 2% of Google engineers are Klingon.

Today we launch a full translation of Blockly in Klingon. It incorporates Klingon cultural norms to facilitate learning in this unique population:

- Blockly has no syntax errors. This reduces frustration, and reduces the number of computers thrown through bulkheads.
- Variables are untyped. Type errors can too easily be perceived as a challenge to the honor of a student's family (and we've seen where that ends).
- Debugging and bug reports have been omitted, our research indicates that in the event of a bug, they prefer the entire program to just blow up.

Get a little keyboard dirt under your fingernails. Learn that although ghargh is delicious, code structure should not resemble it. And above all, be proud that tlhIngan maH. Qapla'!

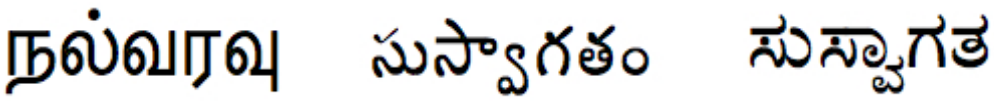
You can try out the demo here or get involved here.

Google Translate welcomes you to the Indic web

Tuesday, June 21, 2011

Posted by Ashish Venugopal, Research Scientist

(Cross-posted on the Translate Blog and the Official Google Blog)



স্বাগতম ಸ್ವಾಗತ

Beginning today, you can explore the linguistic diversity of the Indian sub-continent with Google Translate, which now supports five new experimental alpha languages: Bengali, Gujarati, Kannada, Tamil and Telugu. In India and Bangladesh alone, more than 500 million people speak these five languages. Since 2009, we've launched a total of 11 alpha languages, bringing the current number of languages supported by Google Translate to 63.

Indic languages differ from English in many ways, presenting several exciting challenges when developing their respective translation systems. Indian languages often use the Subject Object Verb (SOV) ordering to form sentences, unlike English, which uses Subject Verb Object (SVO) ordering. This difference in sentence structure makes it harder to produce fluent translations; the more words that need to be reordered, the more chance there is to make mistakes when moving them. Tamil, Telugu and Kannada are also highly agglutinative, meaning a single word often includes affixes that represent additional meaning, like tense or number. Fortunately, our research to improve Japanese (an SOV language) translation helped us with the word order challenge, while our work translating languages like German, Turkish and Russian provided insight into the agglutination problem.

You can expect translations for these new alpha languages to be less fluent and include many more untranslated words than some of our more mature languages—like Spanish or Chinese—which have much more of the web content that powers our statistical machine translation approach. Despite these challenges, we release alpha languages when we believe that they help people better access the multilingual web. If you notice incorrect or missing translations for any of our languages, please correct us; we enjoy learning from our mistakes and your feedback helps us graduate new languages from alpha status. If you're a translator, you'll also be able to take advantage of our machine translated output when using the Google Translator Toolkit.

Since these languages each have their own unique scripts, we've enabled a transliterated input method for those of you without Indian language keyboards. For example, if you type in the word "nandri," it will generate the Tamil word நன்றி (see what it means). To see all these beautiful scripts in action, you'll need to install fonts* for each language.

We hope that the launch of these new alpha languages will help you better understand the Indic web and encourage the publication of new content in Indic languages, taking us five alpha steps closer to a web without language barriers.

*Download the fonts for each language: Tamil, Telugu, Bengali, Gujarati and Kannada.