Natural Language Engineering **24** (6): 951–960. © Cambridge University Press 2018. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution, and reproduction in any medium, provided the original work is properly cited.
doi:10.1017/S1351324918000335

Emerging trends: APIs for speech and machine translation and more

KENNETH WARD CHURCH
Baidu, Sunnyvale, CA, 94089, USA

(*Received 28 August 2018; accepted 28 August 2018*)

Abstract

Lots of companies are offering lots of APIs. Reviews are not always as constructive as they could be. Some reviews encourage unproductive work on checkbox features that no one wants. It makes no sense to do the wrong thing badly. Constructive reviews should help focus priorities on what matters. Users care more about a great box opening experience than small improvements in word error rate and BLEU, popular metrics for speech and translation. In 15 minutes or less, can we teach potential users something new (and fun) that does something useful, such as how to translate PowerPoint between English and Chinese, preserving many of the features that are important to PowerPoint such as graphics and animations? See Appendix for the solution.

1 A rising tide lifts all boats

As Robert Dale pointed out in this journal recently (Dale 2018a, 2018b), a number of big and small companies offer a number of very useful APIs for a number of tasks in speech and language (and much more). His review focused mainly on text analytics, though he has also written about APIs for translation (Dale 2016). This review will also discuss APIs for translation, as well as speech, and combinations thereof.

Translation APIs make it easy to translate text from one language to another (typically in both directions). Speech APIs make it easy to convert speech to text (and vice versa). Python libraries[1] are becoming available that make it easy to switch from one vendor to another, so users will not feel locked into one vendor or another. The Python library, speech_recognition,[2] supports Google,[3] Microsoft,[4]

[1] `https://www.quora.com/What-is-a-good-speech-recognition-library-for-Python`
[2] `https://pypi.org/project/SpeechRecognition/`
[3] `https://cloud.google.com/speech-to-text/`
[4] `https://azure.microsoft.com/en-us/services/cognitive-services/speech/`

IBM[5] and Houndify,[6] among others, though there are many more that are not supported including Amazon[7] and Baidu.[8]

Many companies offer lots of APIs.[9] [10] [11] [12] It makes sense to offer lots of APIs if there is a “better together” story, where combinations of APIs produce more value than the sum of the parts. There are obvious synergies between speech and translation APIs. When I go to China, I regularly use tools that convert speech in Chinese to text in English. There are even more compelling better together opportunities involving freeware, as we will see when we discuss translation of PowerPoint in the Appendix.

Why should researchers be interested in speech and language APIs (and more generally, products)? It is good for the research community when more and more people are using our stuff. It used to be hard for me to explain what I do to a general audience, but now that everyone has used Siri-like apps, it is much easier for me to talk to friends and family about what I do.

There is always a temptation to focus on how to improve the core technology, but in *Good Opportunities for Crummy Machine Translation* (Church and Hovy 1993), Ed Hovy and I argued that we should do what we can to improve the core technology, but in addition to that, we should also think about what we can do with what we already have.

It is good for the field when everyone has enough experience with what we do to set reasonable expectations for what can be done, and what cannot be done. It is good when the public is excited about what we are doing, but too much excitement can lead to unrealistic expectations (and AI winters). Unrealistic expectations are less likely when lots of people are using our stuff every day.

In short, it is good for all of us when there is lots of usage, and even better when usage is growing rapidly. I believe usage would be larger, and would grow even faster, if there was more awareness of just how easy it is to do amazing things with the APIs that are already out there.

Why are not APIs used more than they are?

(1) Core technology still is not good enough?
(2) Marketing issues: awareness, ease-of-use, box opening?
(3) Not enough features?
(4) Lack of a killer app?
(5) Privacy/security?

Of course, it would help if the APIs worked better than they do, but in the spirit of *Good Opportunities for Crummy Machine Translation*, I believe performance is

[5] `https://www.ibm.com/watson/services/speech-to-text/`
[6] `https://www.houndify.com/`
[7] `https://aws.amazon.com/transcribe/`
[8] `http://yuyin.baidu.com/`
[9] `https://cloud.google.com/products/`
[10] `https://aws.amazon.com/`
[11] `https://azure.microsoft.com/en-us/services/cognitive-services/`
[12] `http://ai.baidu.com/`

already good enough to do lots of useful things. We need to do a better job with marketing issues, especially awareness and ease-of-use. Marketing people talk about a funnel that starts with awareness and ends with usage and sales.[13] People generally do not buy (or use) products that they are not aware of. Too many of us, even experts in the field, do not know about APIs that are out there, and how easy it is to do amazingly powerful things with we already have. People ask for what they know what to ask for.

2 Goals for a constructive review

Too many reviews are too divisive. There is usually just a single winner, and the rest are losers.[14] It should not be a zero sum game. There are lots of great solutions out there. Everyone benefits when great solutions succeed. A rising tide lifts all boats. It is good for all of us when more people use our stuff.

Too many reviews emphasize the wrong metrics. Check box features are an awful example of how bad reviews can drive bad (unproductive) behavior. It is common for reviews to start with a long list of features (that no one wants), and assume (incorrectly) that coverage of this list is a good thing. Products naturally respond to this (inappropriate) pressure by doing the minimum they can get away with (on the features that they should not be working on). It makes no sense to do the wrong thing badly. Constructive reviews should help focus priorities on what matters.

Those of us who work on the core technology use other metrics such as word error rate (WER)[15] and BLEU[16] (popular metrics for speech and translation). These metrics may be more meaningful than checkbox features, at least to those of us who work on core technologies, but small incremental improvements in such metrics probably do not matter much to the target audience. Obviously, large differences are very important, but I worry that excessive emphasis on small differences can lead to unproductive behavior somewhat similar to checkbox features. The target audience probably does not care if this vendor is slightly ahead of that vendor, especially if there are more compelling reasons for preferring that vendor such as box opening.

Who is the target audience and what do they care about? APIs need to appeal first to the developers that are likely to build apps on top of these APIs, and second, and more importantly, to the customers that are likely to use whatever these developers produce. What do these audiences care about? They probably do not care much about checkbox features, or small incremental improvements in WER/BLEU.

I suspect these audiences care more about the kinds of things that Apple is really good at. Apple is particularly good at box opening. The first 15 minutes with an Apple product is a joy (the pain comes later). Box opening does not come easy; it takes considerable hard work and passion to get it right, as this TED talk makes clear.[17] When the box opening goes well, then the customer is prepared to listen

[13] https://en.wikipedia.org/wiki/Purchase_funnel
[14] https://www.youtube.com/watch?v=EPdyD21yVc0
[15] https://en.wikipedia.org/wiki/Word_error_rate
[16] https://en.wikipedia.org/wiki/BLEU
[17] https://www.npr.org/programs/ted-radio-hour/478560031/the-power-of-design

some more. That gives you a chance to make a case for your product. What is it good for?

IMHO, a constructive review of APIs should start with a great box opening experience, something like the short (one-page) program in the Appendix. This program can be digested in 15 minutes or less, and does something surprisingly useful, translating PowerPoint files from one language to another, preserving many of the features that are important to PowerPoint such as graphics and animations.

Many people feel a sense of accomplishment after working through a short one-page program that teaches them something new and does something useful. If that can be done in 15 minutes or less, that was a good use of their time. After a successful box opening experience like that, the target audience is prepared to listen some more. What else can be done with these APIs?

While the Appendix is not very complicated, it ought to be possible to make the box opening experience even better. As mentioned above, Python libraries[18] are becoming available that hide much of the complexity. To run the program in the Appendix locally, one needs to obtain a few secrets (appid and secretKey) from a page that is written in Chinese.[19] The Appendix also requires installation of a python library with: pip install python-pptx.[20] For users that want to use the PowerPoint translation service without going through the pain of installing it locally, the service is available at `http://translatepptx.com/`.

3 A few simple (but impactful) use cases

Users probably do not know just how easy it is to do simple (but impactful) things like the following:

(1) Translate documents in popular formats (e.g., PDF, HTML, Microsoft Office) from one language to another.
(2) Accessibility features for hearing and visually impaired.
(3) Accessibility tools for non-Chinese speakers.
(4) Nanny cams and syllable counting.

You can help with awareness by asking your students to do things like this as homework exercises.

3.1 Accessibility

Many of my colleagues at Baidu were very impressed when they saw an accessibility tool that Dimitri Kanevsky developed for his own personal use (personal communication). Dimitri happens to be deaf. When we worked together at IBM, he would walk around with a tablet that transcribed audio so he could read what he could not hear. In those days, the transcription was performed by a human service (in the

[18] `https://www.quora.com/What-is-a-good-speech-recognition-library-for-Python`
[19] `https://api.fanyi.baidu.com/api/trans/product/index`
[20] `https://github.com/scanny/python-pptx`

cloud), but when he moved to Google, that option was no longer available, so he built an alternative solution, replacing the human service with Google's APIs.

Dimitri's tablet may not sound like much when I describe it like I just did, but we were very impressed when we saw it in action, even though we should have known just how easy it is to do things like this. The tablet made it possible for Dimitri to walk around in our noisy (reverberant) lunch room and join in on the conversation like everyone else.

One problem with Dimitri's solution is that it looked as if he was multitasking. His solution worked so well that not everyone at the lunch table realized that he could not hear what they were saying. I was worried that some people might have thought that he was texting someone else during lunch. At some point, I felt it necessary to explain what was going on with the tablet.

Multitasking is like drunk driving. It is easy to think that you are more effective than you are. Unlike texting while driving, texting during lunch probably will not kill you, but it can be annoying.

It is amazing that Dimitri's solution worked as well as it did. I remember when speech recognition just barely worked. Perhaps that is being generous. Demos often went badly wrong. I once gave an after dinner talk titled, 'How to cook a demo.'

We all know lots of tricks for giving demos. Do this, and do not do that. Recording conditions are super important. Some speakers work better than others. Do not let just anyone use the microphone, and make sure the microphone is close to the speaker.

Over time, with lots of hard work (and lots of data collection), speech APIs are getting better and better with many of these well-known challenges. It was remarkable how well Dimitri's tablet worked, even for speakers with accents from China and elsewhere, and even when they were pretty far away from his tablet.

That said, Dimitri's tablet was not perfect. I could not help but notice that the WER was better when we were talking than when Dimitri was talking. He has a Russian accent, but his hearing impairment is probably a bigger challenge for speech recognition. Actually, I am impressed that it worked as well as it did on his speech given that the training corpus probably does not have much speech from speakers with hearing impairments. In any case, it is not that important for this use case, because Dimitri knows what he said. WER matters a lot when we are talking, but not so much when he is talking.

Whatever you measure, you get. We tend to focus on mindless metrics like WER, but it is important to tune the metric to the use case. In this case, the metric should weigh our speech more than his speech. It does not matter that much how well it works when he is talking because he knows what he said.

3.2 Accessibility tools for non-Chinese speakers

In my new position at Baidu, I find myself dependent on tools that are not that different from Dimitri's tablet. I go to lots of meetings, as many of us do. Some of these meetings are in Chinese (both the oral discussion as well as PowerPoint slides).

I have become dependent on a number of translation products/demos. As mentioned above, input can come from many sources including microphones and documents in a variety of popular formats. Translation capabilities have been integrated into many of the apps that I use including web browsers, email readers, social media platforms and more. There are phone-based translation apps that take input from all sorts of places including the keyboard, the microphone and the camera. I sometimes use the camera feature to translate PowerPoint slides, but I find it hard to take a picture of the slide and pay attention to the talk at the same time. I prefer the solution in the Appendix because it avoids multitasking, but that solution requires access to an electronic copy of the slides, which is not always an option.

Translation technology works remarkably well, especially in sunny day scenarios. The technology works better for formal presentations than informal break-out sessions. Break-out sessions introduce a number of challenges: less formal speech, less controlled recording conditions and more diverse speakers.

Recently, Robin Li, the CEO of Baidu, gave a talk to about 200 of us in Chinese. The technology worked great for his talk, but they turned off the translation technology at the end of his talk, making it all too clear to me just how dependent I was on the translation technology. After they turned off the the translation technology, I am less clear about what happened, but I am guessing that the chair of the session asked the audience for questions. When no one seemed eager to ask the first question, the chair handed the microphone to me and 'volunteered' me to ask the first question. It felt a bit like that line from Dirty Harry: 'Do you feel lucky, punk?"[21] There was some nervous laughter at that point; it was a rather high-stakes public evaluation of their translation technology (using a novel metric they were not prepared for). Did their technology work well enough that I could ask a sensible question?

Actually, that evaluation was too easy. Following Doddington *et al.* (1998), it is well known that speech technology works better for some speakers (so called sheep) than others (goats). It has been my experience, working for a number of different speech groups over the years, that speech technology tends to work well for the boss, and especially well for bosses that give a lot of demos. Thus, I am not surprised that Robin is a 'sheep'. He is not only a boss who gives lots of demos, but he also happens to speak unusually clearly in both Chinese and English. That evaluation would not have gone as well during the break out sessions, where there were more 'goats'.

3.3 Nanny cams and syllable counting

It is becoming increasingly feasible to record everything that a young child hears. Nanny cams were originally designed to make it possible to see how your kid is doing when you cannot be there, but they also make it possible to do much more

[21] `https://www.youtube.com/watch?v=8Xjr2hnOHiM`

than that. It may be possible in the not too distant future for a nanny cam to create a web page of your baby's first words like `https://imgur.com/a/KwZ6C`.

It used to be a challenge to record this much audio (and video). Deb Roy popularized the idea of digitizing the first few years of a child's life. His Ted Talk[22] has 2.4M views. When the Human Speechome Project[23] started, it was necessary to install a machine room in Deb Roy's basement. These days, it is relatively easy to capture the audio with consumer electonics, though one should not expect off-the-shelf APIs to work well with speech from babies.

Although it may be a while before the technology is ready to create a log of your baby's first words, there are probably easier tasks such as counting syllables that can already be done with what we already have. It is widely believed that the amount of speech that a child hears early in life is a leading indicator of future success (Weisleder and Fernald 2013). Parents might benefit from a phone app or a smart speaker that counts syllables like Fitbit counts steps. Just as the Fitbit nags you to take more steps, so too, the syllable counting app could nag parents to talk to their children.

4 Concluding remarks: APIs and open versus closed architectures

APIs make it possible for small startup companies to offer small pieces of larger solutions. There used to be some start up companies working on spelling correction, but most users did not want a standalone solution for spelling correction. These start up companies had to find a way to integrate their spelling correction solution into a larger solution such as Microsoft Office. That was not so easy to do in those days. These days, with apps and APIs, the ecosystem is more open than it used to be, making it more likely that a start up company can succeed with a small piece of a larger solution.

APIs can be viewed as an extreme position on the continuum between open architectures and closed architectures. APIs are about as open as open can be. Windows and Android are more typical examples of open architectures; Apple tends to be more closed. Open architectures favor large ecosystems with lots of third parties creating value (and chaos). Closed architectures are more controlled and less chaotic (for better and for worse). I have deep respect for Apple's views on many things, as should be clear from the comments above about box opening. That said, I am biased in favor of the open position and would love to see the extreme open position of APIs succeed.

The debate over security/privacy of logs is somewhat similar to the debate over open and closed architectures. It has been suggested that Apple's privacy policy has downsides (Apple's smart speaker is the least-smart one on the market), but 'right now, that end of the scale is looking like the smart place to be'.[24] It will be interesting to see how this debate plays out over time. This trade-off between privacy and smartness is well described in this history of Siri.[25]

[22] `https://www.ted.com/talks/deb_roy_the_birth_of_a_word`
[23] `https://www.media.mit.edu/cogmac/projects/hsp.html`
[24] `https://9to5mac.com/2018/03/30/apple-privacy-policy/`
[25] `https://www.youtube.com/watch?v=4ryQTkDWmBg`

When I want to think about how things are likely to stand up to the test of time, I often find it useful to look back at how these questions played out a few years ago. Here is a video from 2011 with 1.5 M views of a user having fun with Siri soon after it first came out.[26] Siri's silly jokes did especially well at the time, at least during the box opening experience. But it was clear, even in the video, that those jokes would get old very quickly.

After a successful box opening experience like that, the target audience is prepared to listen some more. What can you do with Siri? Unfortunately, the field still does not have a good answer to this question.

The 2011 video started out with nearly 8 minutes of silly jokes. Then she gets down to business and starts to describe some things that Siri can do that might be useful (though most of them are remarkably uncompelling). At 8:45, she demos one of those use cases by saying, 'call home'. She then gets completely flustered when Siri asks for clarification by reading her home number to all 1.5 M viewers of the video.

Apple's closed architecture has a number of obvious appeals (and challenges). This becomes especially clear with box opening and privacy. Apple starts out with an amazing box opening experience, but struggles with the obvious next two questions:

- Killer Apps: How do we find compelling use cases for what we do?
- Privacy/Security: How do we prevent bad things from happening?

The 'killer app' question remains a challenge today, as evidenced by this more recent video,[27] promising 50 'useful' commands in 5 minutes, though others might not find these commands all that compelling, and some might consider them to be checkbox features. We still do not really know what voice assistants are good for, but I think we will have a better chance of figuring that out if we can encourage more people to propose more suggestions in an open framework.

As for privacy, I appreciate the fact that Apple tries harder than most to address privacy concerns, but even so, privacy is even more elusive today than it was in 2011, as indicated by this more recent (entertaining) video with 3.6 M views, asking Siri and Alexa some awkward questions about privacy.[28]

In summary, while the jury is still out on the open versus closed debate, I would love to see the open side succeed. I especially like the extreme openness of APIs. Open architectures have the advantage of empowering massive ecosystems. The field needs answers to the killer app and privacy questions. Empowering a massive ecosystem is the most promising way forward to answer these questions.

References

Church, K., and Hovy, E. 1993. Good applications for crummy machine translation. *Machine Translation* **8**: 239–258.

[26] `https://www.youtube.com/watch?v=cP2ILuPodPs`
[27] `https://www.youtube.com/watch?v=dJjVMZ1ACIs`
[28] `https://www.youtube.com/watch?v=tBd7LTmWaqU`

Dale, R. 2016. How to make money in the translation business. *Natural Language Engineering* **22**(2): 321–325. doi:10.1017/S1351324916000012

Dale, R. 2018. Text analytics APIs, Part 1: The bigger players. *Natural Language Engineering* **24**(2): 317–324.

Dale, R. 2018. Text analytics APIs, Part 2: The smaller players. *Natural Language Engineering* **24**(5): 797–803.

Doddington, G., Liggett, W., Martin, A., Przybocki, M., and Reynolds, D. 1998. Sheep, goats, lambs and wolves: A statistical analysis of speaker performance in the NIST 1998 speaker recognition evaluation. National Inst of Standards and Technology Gaithersburg Md. arXiv:1808.01371v2.

Weisleder A., and Fernald, A. 2013. Talking to children matters: Early language experience strengthens processing and builds vocabulary. *Psychological science* **24**(11): 2143–2152. doi:10.1177/0956797613488145.

Appendix: A short program to translate PowerPoint

```
# This service is available at http://translatepptx.com
# usage: python3 translate_pptx.py <input> <output> <source> <target>
# where <input> and <output> are pptx files
# and <source> and <target> are either: en (English) or zh (Chinese)

from pptx import Presentation
import sys, http.client, hashlib, random, json, string
from hashlib import md5
import urllib.request, urllib.parse, urllib.error

ppt = Presentation(sys.argv[1])
# request appid and secretKey
# from https://api.fanyi.baidu.com/api/trans/product/index
appid = 'xxxxxxxxxxxxxxxxx'
secretKey = 'yyyyyyyyyyyyyyyyyyyy'
sourceLang = sys.argv[3]
targetLang = sys.argv[4]
salt = random.randint(32768, 65536)
httpClient=http.client.HTTPConnection('api.fanyi.baidu.com')

def translate(q):
    m1 = hashlib.md5()
    m1.update((appid+q+str(salt)+secretKey).encode('utf-8'))
    sign = m1.hexdigest()
    url='/api/trans/vip/translate?appid=%s&q=%s&from=%s&to=
%s&salt=%s&sign=%s' %(
    appid, urllib.parse.quote(q), sourceLang, targetLang,
    str(salt), sign)
    try:
        httpClient.request('GET', url)
        response = httpClient.getresponse()
        r = str(response.read())[2:-1]
        j = json.loads(r)
        return j['trans_result'][-1]['dst'].encode('ascii').
        decode('unicode-escape')
    except Exception as e:
        return '***Exception (%s): %s' % (e,q)

for slide in ppt.slides:
    for shape in slide.shapes:
        if shape.has_text_frame:
            for paragraph in shape.text_frame.paragraphs:
                for run in paragraph.runs:
                    run.text = translate(run.text)

ppt.save(sys.argv[2])
```