Speech-Enabled Hybrid Multilingual Translation for Mobile Devices

Krasimir Angelov
University of Gothenburg
krasimir@chalmers.se

Björn Bringert
Google Inc
bringert@google.com

Aarne Ranta
University of Gothenburg
aarne@chalmers.se

Abstract

This paper presents an architecture and a prototype for speech-to-speech translation on Android devices, based on GF (Grammatical Framework). From the user's point of view, the advantage is that the system works off-line and yet has a lean size; it also gives, as a bonus, grammatical information useful for language learners. From the developer's point of view, the advantage is the open architecture that permits the customization of the system to new languages and for special purposes. Thus the architecture can be used for controlled-language-like translators that deliver very high quality, which is the traditional strength of GF. However, this paper focuses on a general-purpose system that allows arbitrary input. It covers eight languages.

1 Introduction

Many popular applications (*apps*) on mobile devices are about language. They range from general-purpose translators to tourist phrase books, dictionaries, and language learning programs. Many of the apps are commercial and based on proprietary resources and software. The mobile APIs (both Android and iOS) make it easy to build apps, and this provides an excellent way to exploit and demonstrate computational linguistics research, perhaps not used as much as it could.

GF (Grammatical Framework, (Ranta, 2011)) is a grammar formalism designed for building multilingual grammars and interfacing them with other software systems. Both multilinguality and interfacing are based on the use of an *abstract syntax*, a tree structure that captures the essence of syntax and semantics in a language-neutral way. Translation in GF is organized as *parsing* the source language input into an abstract syntax tree and then *linearizing* the tree into the target language. Here is an example of a simple question, as modelled by an abstract syntax tree and linearized to four languages, which use different syntactic structures to express the same content:

Query (What Age (Name "Madonna"))
English: *How old is Madonna?*
Finnish: *Kuinka vanha Madonna on?*
French: *Quel âge a Madonna?*
Italian: *Quanti anni ha Madonna?*

In recent years much focus in GF has been put on cloud applications (Ranta et al., 2010) and on mobile apps, for both Android (Détrez and Enache, 2010) and iOS (Djupfeldt, 2013). They all implement text-based phrasebooks, whereas Alumäe and Kaljurand (2012) have built a speech-enabled question-answering system for Estonian. An earlier speech translation system in GF is presented in Bringert (2008).

All embedded GF systems are based on a standardized run-time format of GF, called PGF (Portable Grammar Format; Angelov et al. 2009, Angelov 2011). PGF is a simple "machine language", to which the much richer GF source language is compiled by the GF grammar compiler. PGF being simple, it is relatively straightforward to write interpreters that perform parsing and linearizations with PGF grammars. The first mobile implementations were explicitly designed to work on small devices with limited resources. Thus they work fine for small grammars (with up to hundreds of rules and lexical entries per language), but they don't scale up well into open-domain grammars requiring a lexicon size of tens of thousands of lemmas. Moreover, they don't support out-of-grammar input, and have no means of choosing between alternative parse results, which in a large grammar can easily amount to thousands of trees.

A new, more efficient and robust run-time system for PGF was later written in C (Angelov, 2011). Its performance is competitive with the state of the art in grammar-based parsing (Angelov and Ljunglöf, 2014). This system uses statistical disambiguation and supports large-scale grammars, such as an English grammar covering most of the Penn Treebank. In addition, it is lean enough to be embedded as an Android application even with full-scale grammars, running even on devices as old as the Nexus One from early 2010.

Small grammars limited to natural language fragments, such as a phrasebook, are usable when equipped with predictive parsing that can suggest the next words in context. However, there is no natural device for word suggestions with speech input. The system must then require the user to learn the input language; alternatively, it can be reduced to simple keyword spotting. This can be useful in information retrieval applications, but hardly in translation. Any useful speech-enabled translator must have wide coverage, and it cannot be restricted to just translating keywords.

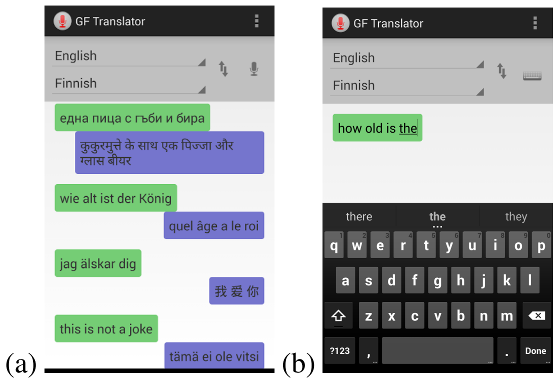
In this paper, we show a mobile system that has a wide coverage and translates both text and speech. The system is modular and could be easily adapted to traditional GF applications as well: since the PGF format is the same, one can combine any grammar with any run-time PGF interpreter.

The rest of the paper is organized as follows: Section 2 describes the system's functionalities from the user's point of view. Section 3 explains the technology from the developer's point of view. Section 4 presents some preliminary results on the usability of the system, and discusses some ways of improving it. Section 5 concludes.

A proper quantitative evaluation of the translation quality has to wait till another occasion, and will be more properly done in a context that addresses hybrid GF-based translation as a research topic. Early attempts in this area have not yet converged into a stable methodology, but we believe that setting translation in the context of a practical use case, as here, can help identify what issues to focus on.

2 Functionalities

The app starts with the last-used language pair preselected for input and output. It waits for speech input, which is invoked by touching the microphone icon. Once the input is finished, it appears in text on the left side of the screen. Its translation appears below it, on the right, and is also rendered as speech (Figure 1 (a)).



една пица с гъби и бира
कुकुरमुत्ते के साथ एक पिज्जा और ग्लास बीयर
wie alt ist der König
quel âge a le roi
jag älskar dig
我 爱 你
this is not a joke
tämä ei ole vitsi

how old is the

Figure 1: Translation between various languages with (a) speech (b) text input.

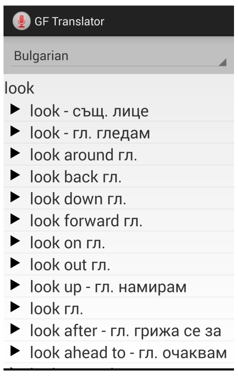
The source and target languages are selected by the two drop-down lists on the top of the screen. The icon with two arrows to the right of the language selectors allows the two languages to be swapped quickly.

The speech recognition and text-to-speech (TTS) is done using public Android APIs. On most devices, these make use of Google's speech recognizer and synthesizer, which are available in both online and offline versions. The offline engines tend to have a reduced choice of languages and reduced quality compared to the online engines, but don't require an internet connection.

Alternatively, the user can select the keyboard mode. The microphone icon is then changed to a keyboard icon, which opens a software keyboard and shows a text field for entering a new phrase. Once the phrase is translated, it is shown on the screen but also sent to TTS (Figure 1 (b)).

If the input consists of a single lexical unit, the user can open a dictionary description for the word. The resulting screen shows the base form of the word, followed by a list of possible translations. The target language is shown on the top of the screen and it can be changed to see the translations in the other languages (Figure 2 (a)). Touching one of the translations opens a full-form inflection table together with other grammatical information about the word, such as gender and verb valency (Figure 2 (b)).

Finally, the translator also works as an input mode for other apps such as SMS. It provides a soft keyboard, which is similar to the standard Android keyboard, except that it has two more keys allowing the entered phrase to be translated in-place from inside any other application.

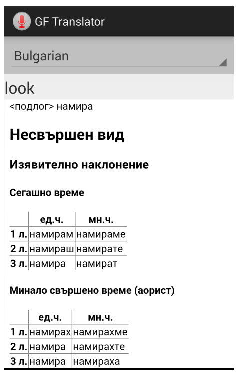
(a) 

GF Translator

Bulgarian

look

- look - същ. лице
- look - гл. гледам
- look around гл.
- look back гл.
- look down гл.
- look forward гл.
- look on гл.
- look out гл.
- look up - гл. намирам
- look гл.
- look after - гл. грижа се за
- look ahead to - гл. очаквам

(b) 

GF Translator

Bulgarian

look

<подлог> намира

Несвършен вид

Изявително наклонение

Сегашно време

	ед.ч.	**мн.ч.**
1 л.	намирам	намираме
2 л.	намираш	намирате
3 л.	намира	намират

Минало свършено време (аорист)

	ед.ч.	**мн.ч.**
1 л.	намирах	намирахме
2 л.	намира	намирахте
3 л.	намира	намираха

Figure 2: (a) Results of dictionary lookup. (b) Valency and the inflection table for a Bulgarian verb.

3 Technology

3.1 Run-time processing

The core of the system is the C runtime for PGF (Angelov, 2011). The runtime is compiled to native code with the Android NDK and is called via foreign function interface from the user interface, which is implemented in Java.

The main challenge in using the runtime on mobile devices is that even the latest models are still several times slower that a modern laptop. For instance, just loading the grammars for English and Bulgarian, on a mobile device initially took about 28 seconds, while the same task is a negligible operation on a normal computer. We spent considerable time on optimizing the grammar loader and the translator in general. Now the same grammar, when loaded sequentially, takes only about 5-6 seconds. Furthermore, we made the grammar loader parallel, i.e. it loads each language in parallel. The user interface runs in yet another thread, so while the grammar is loading, the user can already start typing or uttering a sentence. In addition, we made it possible to load only those languages that are actually used, i.e. only two at a time instead of all eight at once.

Parsing is a challenge in itself. As the grammars grow bigger, there tends to be more and more need for disambiguation. This is performed by a statistical model, where each abstract syntax tree node has weight. We used the method of Angelov and Ljunglöf (2014) to find the best tree.

Moreover, since any sound grammar is likely to fail on some input, there is need for robustness. This has been solved by chunking the input into maximal parsable bits. As a result, the translations are not always grammatically correct, because dependencies between chunks, such as agreement, get lost. This kind of errors are familiar to anyone who has used a statistical system such as Google translate. In the GF system it is easy to avoid them, provided the parse is complete.

Bulgarian	26664	French	19570
Chinese	17050	German	9992
English	65009	Hindi	33841
Finnish	57036	Swedish	24550

Table 1: Lexical coverage (lemmas)

3.2 The language component

The language-specific component of the app is the PGF grammar, which contains both the grammars proper and the probabilistic model of the abstract syntax. The app can be adaptad to a different PGF grammar by changing a few lines of the source code. Hence any grammar written in GF is readily usable as the language component of an app. But here we focus on the large-scale grammar meant for robust translation.

The core of the grammar is the GF Resource Grammar Library (Ranta, 2009), which currently covers 29 languages. Of these, 8 have been extended with more syntax rules (about 20% in addition to the standard library) and a larger lexicon. Table 1 shows the list of languages together with the size of the lexicon for each of them. The abstract syntax is based on English lemmas and some split word senses of them. The other languages, having fewer words than English, are thus incomplete. Unknown words are rendered by either showing them in English (if included in the English lexicon) or just returning them verbatim (typical for named entities).

The lexicon has been bootstrapped from various freely available sources, such as linked WordNets and the Wiktionary. Parts of the lexicon have been checked or completely written manually.

4 First results

The most striking advantage of the translation app is its lean size: currently just 18Mb for the whole set of 8 languages, allowing translation for 56 language pairs. This can be compared with the size of about 200Mb for just one language pair in Google's translation app used off-line. The Apertium off-line app is between these two, using around 2MB per language pair.

The speed is still an issue. While the app now loads smoothly on modern hardware (such as Nexus 5 phones), translation is usually much slower than in Google and Apertium apps. The speed depends heavily on the complexity of the source language, with Finnish and French the worst ones, and on sentence length. Only with short sentences (under ten words) from Bulgarian, Chinese, English, and Swedish, does the translator deliver satisfactory speed. On the other hand, long sentences entered via speech are likely to contain speech recognition errors, which makes their translation pointless anyway.

Translating single words is based on a simpler algorithm (dictionary lookup) and is therefore immediate; together with the grammatical information displayed, this makes single word translation into the most mature feature of the app so far.

The translation quality and coverage are reasonable in phrasebook-like short and simple sentences. The app has exploited some idiomatic constructions of the earlier GF phrasebook (Détrez and Enache, 2010), so that it can correctly switch the syntactic structure and translate e.g. *how old are you* to French as *quel âge as-tu*. In many other cases, the results are unidiomatic word-to-word translations but still grammatical. For instance, *hur mycket är klockan*, which should give *what is the time*, returns *how mighty is the bell*. Such short idioms are typically correct in Google's translation app, and collecting them into the GF resources will be an important future task.

On the plus side, grammar-based translation is more predictable than statistical. Thus (currently) when using Google translate from Swedish to English, both *min far är svensk* and its negation *min far är inte svensk* come out as the positive sentence *my father is Swedish*. With grammar-based translation, such semantic errors can be avoided.

5 Conclusion

We have presented a platform for mobile translation apps based on GF grammars, statistical disambiguation, and chunking-based robustness, enhanced by Android's off-the-shelf speech input and output. The platform is demonstrated by a system that translates fairly open text between 8 languages, with reasonable performance for short sentences but slow parsing for longer ones, with moreover lower quality due to more parse errors.

The processing modules, user interface, and the language resources are available as open source software and thereby usable for the community for building other systems with similar functionalities. As the app is a front end to a grammatical language resource, it can also be used for other language-aware tasks such as learning apps; this is illustrated in the demo app by the display of inflection tables. The app and its sources are available via http://www.grammaticalframework.org.

References

Tanel Alumäe and Kaarel Kaljurand. 2012. Open and extendable speech recognition application architecture for mobile environments. *The Third International Workshop on Spoken Languages Technologies for Under-resourced Languages (SLTU 2012), Cape Town, South Africa*.

Krasimir Angelov and Peter Ljunglöf. 2014. Fast statistical parsing with parallel multiple context-free grammars. In *European Chapter of the Association for Computational Linguistics*, Gothenburg.

Krasimir Angelov, Björn Bringert, and Aarne Ranta. 2009. PGF: A Portable Run-Time Format for Type-Theoretical Grammars. *Journal of Logic, Language and Information*, 19(2), pp. 201–228.

Krasimir Angelov. 2011. *The Mechanics of the Grammatical Framework*. Ph.D. thesis, Chalmers University of Technology.

Björn Bringert. 2008. Speech translation with Grammatical Framework. In *Coling 2008: Proceedings of the workshop on Speech Processing for Safety Critical Translation and Pervasive Applications*, pages 5–8, Manchester, UK, August. Coling 2008 Organizing Committee.

Grégoire Détrez and Ramona Enache. 2010. A framework for multilingual applications on the android platform. In *Swedish Language Technology Conference*.

Emil Djupfeldt. 2013. Grammatical framework on the iphone using a C++ PGF parser. Technical report, Chalmers Univerity of Technology.

Aarne Ranta, Krasimir Angelov, and Thomas Hallgren. 2010. Tools for multilingual grammar-based translation on the web. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 66–71, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aarne Ranta. 2009. The GF resource grammar library. *Linguistic Issues in Language Technology*.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford. ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).