Transfer pivot translation in Apertium

Shardul Chiplunkar
for Google Code-In 2016

Keywords: rule-based machine translation (RBMT), pivot translation, Apertium, ScaleMT

Abtract

Rule-based machine translation (RBMT) systems built with the Apertium platform produce high quality translations, given extensive bilingual dictionaries and shallow-transfer lexical and structural rules [1]. I implemented transfer pivot translation—translation using one or more languages as intermediate ('pivot') translation targets without specially modifying the translations in the process ('transfer')—to extend the scope of possible translations with minimal additional effort. This paper discusses the details of this work and its future implementation prospects across Apertium's API, web interface, and IRC bot.

Introduction

Apertium is a free and open-source RBMT platform which can build RBMT systems from bilingual dictionaries and shallow transfer rules for lexical selection and structural transfer [1]. Apertium itself has extensively developed some such systems, such as among various Romance languages ([2] describes initial work), along with community-driven systems such as for languages of the Turkic family[1]. Apart from the morphological analyzer/generator and part-of-speech tagger that each language requires, each language pair requires a bilingual dictionary to find equivalents of lexical forms, lexical selection rules to disambiguate analyses, and shallow-transfer rules to transfer syntactical structures. While this produces high-quality translation systems, these requirements are difficult to fulfill as they must most often be written manually.

Pivot translation is a technique that can expand the scope of possible translations using one or more intermediate ('pivot') target languages between the source and final target languages. For example[2], Apertium does not have a direct translation system from English (`eng`) to Portuguese (`por`), but with pivot translation, Spanish (`spa`) and Catalan (`cat`) can be used to translate

```
eng → spa → por
eng → spa → cat → por
```

etc. as required. As Apertium is a rule-based platform, transfer pivot translation—where no special modifications are made at any intermediate step—is the easiest way to do this, as opposed to other common approaches to pivot translation which involve statistical approaches.

I implemented transfer pivot translation in Apertium's web API '`apertium-apy`'[3] which is modeled on ScaleMT [3], and in the web interface '`apertium-html-tools`'[4]. The ScaleMT infrastructure uses parallel asynchronous pipelines for each translation pair, so the challenge was to ensure that a text being translated is correctly passed through multiple such pipelines without slowing down the system. On the other hand, the challenge for the web interface was to create an intuitive interface to view and select from possible pivot translations. I also developed an interface to pivot translation for Apertium's IRC bot `begiak`.

The following sections of this paper describe, in order, details of contributions to `apertium-apy`, `apertium-html-tools`, `begiak`, and possibilities for future work.

Web API

`apertium-apy` is modeled after the ScaleMT architecture for efficiency and scalability [3]. Each translation pair, e.g. `eng → spa`, has an associated pipeline running independently of all other

[1] Outlined at `http://wiki.apertium.org/wiki/Apertium_Turkic#Publications`.

[2] This will be a running example throughout this paper.

[3] `https://github.com/goavki/apertium-apy`

[4] `https://github.com/goavki/apertium-html-tools`

pipelines. Pipelines initialize all required resources when started and maintain them thereafter. Pipelines are started and stopped on various servers by a central router based on the quantity of requests and server load. The router also provides an outward-facing REST API with endpoints for functions such as translation, morphological analysis and generation, listing available pairs, calculating translation coverage, etc.

Pivot translation requires passing the source language text into one of these pipelines, waiting for its output, then passing it into another pipeline, and so on until the text is obtained in the target language. However, as the pipelines run independently in parallel, care had to be taken to perform the entire operation asynchronously so as to not block the entire system or slow down individual pipelines with a pivot translation request. The crucial element that I wrote was a `coreduce` routine[5], which takes a list of functions and the initial arguments as its arguments and asynchronously applies each function to the result of the previous function.

I developed two API endpoints[6] to access this functionality. The first was `translateChain` which takes a list of languages, e.g. `eng|spa|por`, and a text in the source language `eng`, and gives the text after performing the `eng → spa → por` translation.

It would be convenient if `apertium-apy` automatically chose pivot translation languages if only the source and target were provided. Thus, I implemented Dijkstra's algorithm for finding shortest paths between all possible language pairs on a particular server, which are found and recorded by the server at start-up for future use. Dijkstra's algorithm was used in particular because of the possibility of using weights for each direct translation pair—currently, the weights are all equal and have no bearing on the path chosen—which is discussed in the 'Summary and Future Work' section.

The second API endpoint was `listPairs` which already had modes to list direct pairs, analyzers/generators, and taggers. I added a mode to take a source language and give all possible target languages that could be obtained through pivot translation. This mode merely accesses the list of shortest

[5]Line 251 in `translation.py` in `https://github.com/goavki/apertium-apy/pull/43/files`

[6]Documented at `http://wiki.apertium.org/wiki/User:Shardulc`.

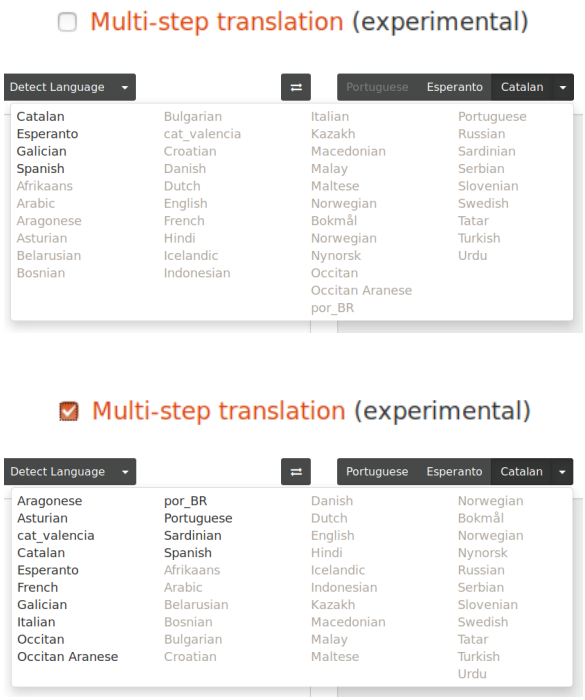

Figure 1: Target languages for `eng` with and without pivot translation

paths described in the previous paragraph. For example, Figure 1 shows that with `eng` as the source language, the only direct translation targets are `spa`, `cat`, and a couple others, but the pivot targets include `por` too.

Web Interface

Preliminary changes to `apertium-html-tools` for pivot translation are shown in Figure 1. A checkbox was added to enable pivot translation along with a link explaining what pivot translation is (termed "multi-step translation" so that the function is clearer from the name). If this checkbox was checked, the dropdown and target language selection bar, which would otherwise only enable direct targets, enabled targets that would require pivot translation to achieve. To actually perform the pivot translation, `apertium-html-tools` would use the API described previously.

However, this approach has a few shortcomings: it does not display the pivot translation path for a selected target language, or even its length for a rough measure of accuracy, and it does not allow the user to select custom pivot translation paths. I

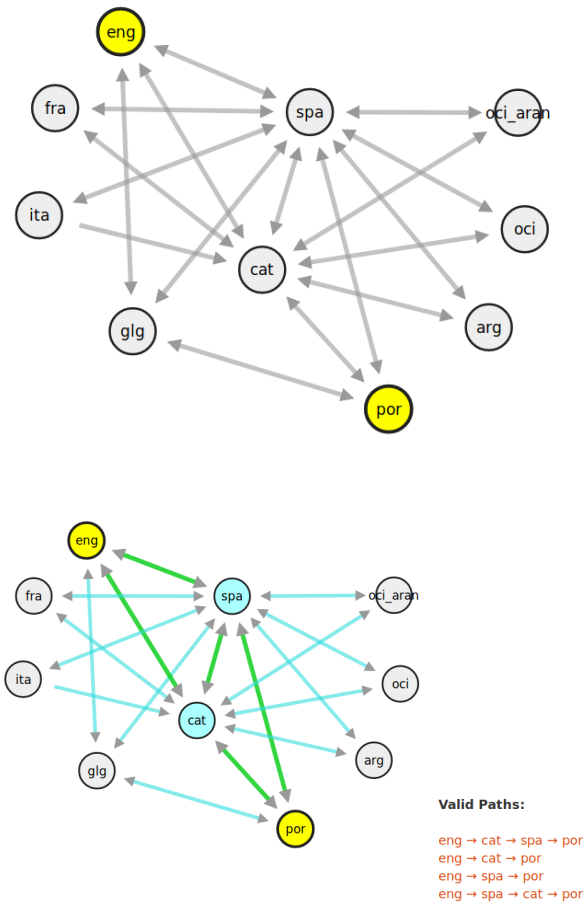

Figure 2: Unselected pivot translation graph for `eng → por` and the same graph upon selecting `cat` and `spa` nodes

built a graphical interface to remedy this which is shown in Figure 2.

The user is presented with a directed graph of possible direct translations that can be used as pivot languages for the `eng → por` translation. The user clicks to select pivot languages `cat` and `spa`, represented as nodes, which then turn blue; all direct translation pairs, represented as arrows, which are part of some pivot path involving the selected languages also turn blue. (In the figure, there is no direct translation pair that does not involve `spa` or `cat`, so there are no gray arrows.) If any paths involving only the selected languages exist, the arrows for the entire path turn green and the desired path can be selected from a 'Valid Paths' list.

The graph shown is already complex and needs to be movable and spread out to be of use. Thus, it is built using the force-directed graph provisions of the powerful JavaScript visualization library `D3.js`[7]. This means that the nodes repel each other and the arrows tend to maintain their length, while the source and target language nodes can be moved around by the user as wanted, making the whole interface dynamic and intuitive. Care was taken to ensure that the interface functions as expected on mobile devices as well.

IRC Bot

Apertium's IRC bot `begiak` could already obtain direct translations using `apertium-apy`, so adding provisions for pivot translations—of course, performed on the API side—was not a difficult matter. However, I discovered a large number of malfunctioning or broken API-accessing commands in the process, and rewrote almost the entire module[8] so that it could provide access to most, if not all, `apertium-apy` functions.

In order to aid future development, I added documentation for the new code and bot commands, and also created extensive unit tests[9] for the module.

Future Work

An important possibility for future development is using the quality of direct translations to decide pivot translation paths. Absolute quality evaluations for a direct pair are not available and may not be practical, but estimates would make a great difference for pivot translation. These estimates could be used as weights in the implementation of Dijkstra's algorithm mentioned in the 'Web API' section, could be served over the API, and could be used to provide a visual representation of the quality of direct translations and pivot paths in the graphical interface in the 'Web Interface' section. It is also possible that the weights are adjusted depending on the text to be translated after determining translation coverage for that specific text, but this appears to be very inefficient.

[7] `https://d3js.org/`

[8] `https://github.com/goavki/phenny/blob/master/modules/apy.py`

[9] `https://github.com/goavki/phenny/blob/master/modules/test/test_apy.py`

The lack of such quality evaluations makes pivot translation an experimental feature which would not have as much utility to average users as the other services Apertium provides. Thus, pivot translation has only been enabled on the Apertium Turkic website[10] and not on the main Apertium website[11], which may be the object of future work.

Finally, due to minor bugs and a lack of rigorous documentation and testing, the graphical interface is not currently part of any Apertium websites. The inclusion of the interface would benefit the Apertium Turkic website which has enabled pivot translation but suffers from the shortcomings of the preliminary approach described in the 'Web Interface' section.

Overall, pivot translation is a promising prospect for Apertium, expanding the scope of possible translation services. While translation quality has not been objectively evaluated, some direct translation pairs are of high enough accuracy that one-pivot and occasionally two-pivot translation produces good results. Further development might eliminate the need for pivot translation, or—an alternative which has been suggested early in Apertium's history [4] and more-or-less followed since then—improve existing direct pairs through community involvement to the point where pivot translation merely provides bridges between groups of languages with high-quality translations among themselves.

Summary and Acknowledgments

I integrated transfer pivot translation functionality across Apertium's infrastructure, comprising development of the web API, the web interface, and the IRC bot. For these contributions, I was selected by Apertium mentors as one of two Grand Prize Winners in Google Code-In 2016, an open-source software development contest for high-schoolers running from December 2016 to January 2017. (A total of 34 Grand Prize Winners were selected by 17 organizations in the contest, from over 1,300 participants across the world.)

I express my heartfelt gratitude towards Jonathan Washington, Sushain Cherivirala, Kevin Brubeck Unhammer, Joonas Kylmälä, Wei En, Tino Didriksen, and many others on the `#apertium` IRC channel and on GitHub for their guidance and feedback. I am also grateful to the Google Code-In team for organizing the Google Code-In contest, providing the essential resources that introduced me to Apertium and to open-source development. I look forward to contributing to Apertium in the future.

[10] `http://turkic.apertium.org/`

[11] `https://www.apertium.org/`

References

[1] Tyers, Francis M. et. al. "Free/open-source resources in the Apertium platform for machine translation research and development". *The Prague Bulletin of Mathematical Linguistics*, no. 93, Jan. 2010, pp. 67–76.

[2] Corbí-Bellot, Antonio M. et. al. "An open-source shallow-transfer machine translation engine for the romance languages of Spain". *Proceedings of the European Association for Machine Translation*, 10th Annual Conference, Budapest, Hungary, 30–31 May 2005, pp. 79–86.

[3] Sánchez-Cartagena, Víctor M. and Pérez-Ortiz, Juan Antonio. "ScaleMT: a Free/Open-Source Framework for Building Scalable Machine Translation Web Services". *The Prague Bulletin of Mathematical Linguistics*, no. 93, Jan. 2010, pp. 97–106.

[4] Armentano-Oller, Carme, et. al. "An open-source shallow-transfer machine translation toolbox: consequences of its release and availability". *Proceedings of Open-Source Machine Translation (OSMaTran)*, Machine Translation Summit X, Phuket, Thailand, 12–16 Sep. 2005, pp. 23–30.