Automating high-quality translations for Mobile Apps

Thesis subtitle

Master Thesis

Wanzenried & Stefan
from
Bern BE, Switzerland

Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

Month and Year

Prof. Dr. Philippe Cudr-Mauroux, eXascale Infolab, University of Fribourg, Switzerland, Supervisor
Roman Prokofyev, eXascale Infolab, University of Fribourg, Switzerland, Assistant

Abstract

Every day, hundreds of mobile applications are added to stores such as Google Play or AppStore. Many of them are intended to be used internationally, and thus require translation of the interface. At the same time, many more mobile apps already available for download in these stores. Leveraging the translation bases of existing applications, we could immediately provide high-quality translations for new apps without need to go though a human-translation process.

This project aims to extract and parse translations of existing applications to see if they can be used to translate new ones. A prototype application allows to translate existing apps from a given source to a target language. Translations are generated with different algorithms in statistical machine translation.

Contents

1 Introduction

Android and iPhone smartphones are available in countries all over the world. If an app is intended to be used internationally, translating it into multiple languages can be important to attract more users. If people understand the app better, they may leave higher ratings. However, translating an app requires additional effort. From the developers perspective, all strings must be isolated in order to replace them with different values for each supported language. Furthermore, translating all text itself is a difficult task which usually requires the work of a professional translator. // TODO

This thesis focus on automatic translations of Android apps from English to French and German. The main idea is to collect a lot of existing apps from the Google Playstore[1] and extract their translations. Are we able to provide high quality translations with the help of statistical machine translation (SMT) systems? In order to produce good translations, SMT system must consist of a big vocabulary. It is therefore assumed that apps belonging to the same categories (e.g. finance, audio, sport etc.) share similar translations. SMT is the dominant approach in automating translations and used by services like Google Translate[2] and Bing Translator[3] (Microsoft). In this thesis, we use different SMT algorithms to generate translations. A prototype web application is built which serves as a playground to translate strings and compare the results. Finally, the results of different translation systems are compared and the translation quality is evaluated with the BLEU metric.

[1] `https://play.google.com/store`
[2] `https://translate.google.com`
[3] `https://www.bing.com/translator`

2 Problem Analysis

// TODO

2.1 Android apps and internationalization

Android uses a key-value based approach to store translations and other resources in XML files[1]. For each supported language, one XML file exists, containing the translated values to the given keys. In the source code, a string is no longer hardcoded but referenced by a translation key.

// TODO

2.2 Challenges

- Quality of existing translations? - Where and how to obtain large amount of apps - Hardware for SMT systems. Training may need a lot of time - How many data is needed to produce good results? - How can we measure good translations?

[1] http://developer.android.com/guide/topics/resources/localization.html

3 Data Analysis

3.1 Apps

A total of 698 free Android apps were collected from various categories[1], ensuring a good mix of different translations. Most of them were chosen randomly beside the top apps, mostly from Google, that ensure good translation quality of different languages. Figure 3.1 shows the top ten apps according to the number of available english translations. Note that the counts indicate the number of effective translations after eliminating uninteresting data during pre-processing the corpus (see. chapter XY). Most of the apps have similar counts for all the three languages. Surprisingly, some apps provide more translations for german and french than english. Different counts can happen because of:

1. Not all strings were translated to all languages. Some english words don't need to be translated, e.g. E-Mail.
2. Some translations were abandoned during pre-processing.
3. The XML files containing the translations are out of sync.

34 apps did not isolate their strings in XML files (we assume that they are hardcoded in the programming code). This leaves a effective number of 664 apps where we could extract translations. However, 241 apps were not multilingual and therefore not useful for building translation systems. Unfortunately, apps are not marked as multi-language in the app store, so we only know if there are translations available after downloading and extracting the data.

[1] Click on Categories to see a list of available categories at: `https://play.google.com/store/apps?hl=en`

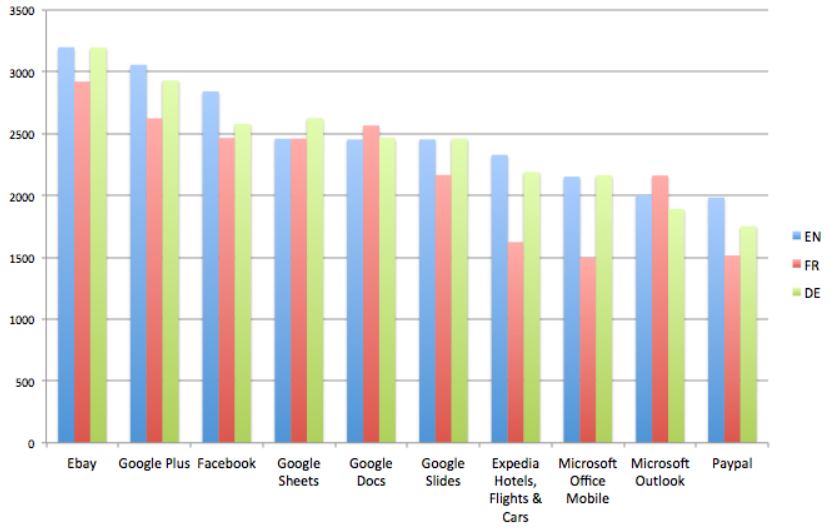

Figure 3.1: Top apps by the number of english translations

3.2 Languages

We collected a total of 423 multilingual apps, offering its content at least for two languages. Figure 3.2 shows the number of apps for the ten most used languages. French and German, where we focus on this thesis, are both in the top 4. The top apps are translated in many languages. In total, 117 different languages were used.

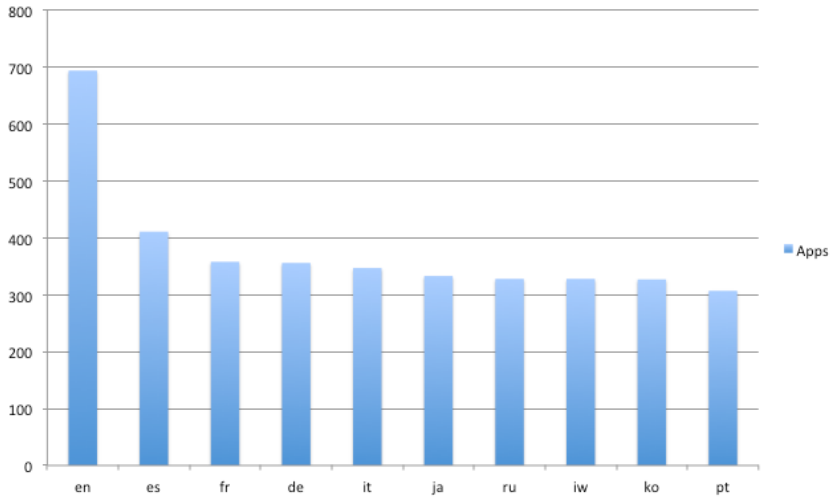

Figure 3.2: Number of apps available in the top ten languages.

3.3 Top Terms

EN	Count	FR	Count	DE	Count
Cancel	1894	Annuler	1067	Abbrechen	886
Done	1230	Supprimer	745	Google Play-Dienste aktivieren	730
Settings	1102	Connexion	745	Google Play-Dienste installieren	728
Search	884	Mettre jour	486	Anmelden	664
Delete	704	Rechercher	485	Lschen	664
Enable Google Play services	692	Paramtres	450	Aktualisieren	652
Get Google Play services	690	Mettre jour les services Google Play	370	Einstellungen	569
Log Out	556	Activer services google Play	368	Fertig	548
Update	545	Activer les services Google Play	366	Weiter	484
Share	541	Installer les services Google Play	366	Schliessen	428

Table 3.1: Top ten terms for english, french and german

4 Data extraction, preparation and storage

4.1 Extract translations

An android app has the form of a single binary file (suffix .apk). The XML files containing the translations are packed inside the .apk file. We used a software called "Apktool"[1] to reverse engineer existing apps and extract the needed XML files.

4.2 Pre-process translations

This task involves two parts. First of all, the translations are sanitized from unwanted information such as HTML tags. Some strings are omitted at this stage because they do not provide any meaningful value to the translation process, for example URLs or placeholders combined with plain numbers. Secondly, the sanitized strings are prepared to be read by an SMT system.

4.2.1 Sanitizing

The following rules are used to get rid of unwanted translations or parts of it:

- Trim the string from newlines (\n), tabs (\t) or carriage return (\r)
- Strip any HTML tags
- Omit strings that are URLs (starting with http or www)
- Remove strings placeholders like `%s` or `%1$s`
- Ignore words with less than 3 characters
- Finally, the resulting string is trimmed again from spaces and must contain at least one alphanumerical character

[1] `http://ibotpeaches.github.io/Apktool/`

4.2.2 Tokenization and Truecasing

Tokenization is required to separate words from punctuation. Each token is separated by a space character: "`Hi, my name is John.`" becomes "`Hi , my name is John .`"

After tokenizsation, we applied Truecasing to our text corpus.[2] In contrast to lowercasing, truecasing determines the proper capitalization of words. This is done by analyzing all data first and then apply the most likely capitalization. It is especially useful for words starting a sentence, that are written uppercase in many languages. For example the word "`This`" starting a sentence will then be recognized as "`this`" by an SMT system.

4.3 Data storage

All translations are stored in text files. Additionally, each translation is stored in a document with Apache Solr[3]. Solr is an open source text search platform built on top of Apache Lucene[4].

4.3.1 Corpus

After pre-processing the translations, they are written to text files, one sentence per line. The corpus is split into parallel and monolingual data. Parallel data of two language pairs contains two text files, one for each language. All sentences must be aligned so that line x of the target language corresponds to line x of the source language. A SMT system uses this parallel data to train the translation model. In addition, data of any language is stored monolingual. All sentences are written into one single text file. This is for example used to build language models. // TODO

4.3.2 Apache Solr

Solr allows to store the translations with additional meta data, such as the translation key and a unique app ID. Due to its way of indexing text data, Solr is very fast when querying translations. Furthermore, built in tools allow us to get nice statistics about our translations, for example the top terms grouped by language. // TODO

4.3.2.1 Setup and schema

Solr offers so called "Cores" where each core manages a separate index, schema and configuration[5]. In our setup we created one core per language holding all the translations. The schema is identical for each core and consist of the following fields:

4.3.2.2 Queries

[2] `https://en.wikipedia.org/wiki/Truecasing`
[3] `http://lucene.apache.org/solr/`
[4] `https://lucene.apache.org/core/`
[5] `https://cwiki.apache.org/confluence/display/solr/Solr+Cores+and+solr.xml`

5 Statistical machine translation SMT

5.1 Phrase-based

5.1.1 Moses

5.2 Deep learning

5.2.1 Tensorflow

6
Prototype web application

6.0.1 Architecture

6.0.2 Frontend

7 Evaluation

7.1 Comparing results

7.2 BLEU score