Twittercorp. A python script for creating and distributing location-aware Twitter corpora.

Tom Ruette[*]

Version of November 14, 2013[†]

Abstract

Corpora of Computer-mediated Communication allow for a perspective on written public language production. Also, there is a (questionable?) tendency to consider such corpora also as indicative of characteristics of spontaneous language use, due to its communicative and immediate character. With the institutionalization of online communication – as an example, many blogs are professionally written with the goal of making money and many twitter feeds are produced by companies or advertisers – the status of such corpora may need to be redefined. The youngest member on the list of Computer-mediated Communication protocols is Twitter. This whitepaper describes a python implementation for collecting a location-aware Twitter corpus. The possibility to automatically annotate tweets with the location affiliation of the Twitter user makes such collections a valuable tool for dialectologists. Also, given the strict Terms of Service of Twitter, a method was implemented to distribute Twitter corpora without violating these terms. Finally, the script contains aa search method that supports regular expressions.

1 Introduction

Corpora of Computer-mediated Communication allow for a perspective on written public language production. Also, there is a (questionable?) tendency to consider such corpora also as indicative of characteristics of spontaneous language use, due to its communicative and immediate character. With the institutionalization of online communication – as an example, many blogs are professionally written with the goal of making money and many twitter feeds are produced by companies or advertisers – the status of such corpora may need to be redefined.

In order to investigate the language use of Computer-mediated Communication, it is necessary to have access to the linguistic production of this medium. Therefore, an easy to use python script was implemented to compile a Twitter corpus in a highly controlled way, so that it meets the highest corpus linguistic standards. Moreover, since science is a business of verification and falsification,

[*] KU Leuven

[†] The development of this script has been a collaborative effort. A list of contributing developers can be found at GitHub:
`https://github.com/ruettet/twittercorp/graphs/contributors`.

a method is provided to distribute the compiled corpora (or compile a comparable one) without violating the Twitter Terms of Service. As such, research on the basis of a Twitter corpus that is compiled with the here described script should also comply with perhaps the most important requierement of science, i.e. data transparancy.

An important aspect of the Twitter corpus that you can compile with this script is that it is location-aware. This means that the tweets can be linked to a certain location. However, unlike the available geolocation possibilities of tweets, which is based on the physical location of the device that was used to broadcast the tweet, the location in the corpus will be based on the location that the user has provided in his Twitter profile. Using this location is much more informative for linguistic purposes, as we can correlate linguistic phenomena to the location of the user, and not to the location of the device.

After running the script – which may take several weeks – a controlled corpus of Twitter messages is presented in an XML format, which contains per tweet the ID of the tweet, its Twitter user, the reported location of the Twitter user, a normalized location, the GPS coordinates of this normalized location, the timestamp, and obviously the text of the tweet itself. How to get to such a corpus is explained in detail in the following sections.

2 Quick user guide

In this quick user guide, the script is explained from the perspective of the user who is primarily interested in compiling the corpus so that linguistic research can be performed. No technical matters are discussed, and only the necessary steps that need to be taken are explained.

2.1 Installing dependencies

The twittercorp script `tw.py` depends on `python-geopy` and the `python-twitter` module, which in its turn depends on `python-oauth2`, `python-urllib2` and `python-json`.

2.2 Obtaining a Twitter API key

To access Twitter, it is necessary to obtain a Twitter API key via `dev.twitter.com`.

2.3 Entering settings

The main settings for the script need to be entered in the file `settings.txt`. In total, five settings need to be provided by the user.

2.4 Running the script

Assuming a working python installation is available on your system, the script can simply be started by issueing the command `python tw.py` at your command prompt. If no further flags are set, the script will try to build the corpus on the basis of the settings in the `settings.txt` file. Further flags are available (to be implemented still) for exporting and importing corpora, and to search in an available corpus.

2.5 Potential errors

It is possible that the script runs into an error. To my knowledge, all possible errors are supposed to be caught, but it is possible that something has slipped through. Use the Issues in GitHub to contact the developers.

3 Methodological steps

In this section, some methodological decisions that were taken during the implementation of the script are made explicit, so that it is completely transparant which assumptions are being made during the compilation phase.

Generally speaking, though, it is important to explain that the script works in two steps. The first step is the collection of Twitter users with a reported location that is within the scope of the geographical coverage of the corpus. Obviously, this means that during the first step, no actual tweets are being downloaded. The second step is downloading the tweets of the collected Twitter users.

3.1 Recursive finding of users

The list of users is collected by means of a recursive methodology that looks for friends of friends. In the settings file, a first (small) list of users is provided. It is advised that this is a list of users with a lot of followers, typically newspapers or politicians. These starting users will be added to the list of users that will be used to collect tweets.

Also, from every starting user, his or her friends are fetched. These friends are added to the list of Twitter users that can be considered during the collection of the tweets. However, not all friends are added. There is one restrictions in place, which is discussed below.

At this stage, the list of users that can be used to collect tweets from has grown. From this list, a new list of "starting users" is derived. With this new list of "starting users" the whole procedures starts over, i.e. friends for each starting user are sought, and they are added to the list of Twitter users that can be used to collect tweets from (if the restriction is met).

This procedures repeats itself over and over, and during each cycle, the list of Twitter users for collection grows. The cycles stop when a certain amount of users is reached, and this amount can be set in the settings file with the `convergence` parameter.

There is one restriction on whether or not the friend of a starting user is allowed onto the list of Twitter uses that will be used to collect tweets. The (normalized) location of the friend needs to be within the geographical area that is defined in the `cities.txt` file. The verification is fairly simplistic: the normalized location needs to contain (case-insensitively) at least one of the locations in the `cities.txt` file.

The cities file is therefore usually filled with a single city per line (hence the name of the file). However, it is also possible to just provide a country per line, since the normalized location will also contain the country name.

3.2 Location normalization

The reported locations of the Twitter users are not necessarily in a standard format. To normalize these reported locations, the reported location is fed into the `geocoder` functionality of `python-geopy`. The non-normalized reported location is basically sent to Google Maps, and the first hit of Google Maps is returned, including latitude and longtitude, and the country name.

In most cases, this normalization is correct. However, sometimes, it is possible that this normalization is simply wrong. There is no means of identifying the correctness of the normalization automatically. Therefore, the final output contains both the reported and the normalized location, so that the researcher can manually exclude or correct those observations that have a wrong normalization.

There are obvious cases in which the normalization fails. When a certain location name corresponds to multiple actual locations, the normalization defaults to one of these locations. Also, many people now have mobile lives, and live in multiple locations. If this is the case, the normalization may find weird locations, e.g. a streetname that contains one of the locations in the other location.

3.3 XML creation

During the second step of the corpus compiliation, i.e. the step in which actual tweets are being downloaded, several xml files will be created. After every 10.000 tweets, a single xml file is written, in which every individual tweet has the structure in Figure 1. These individual tweets are contained within the root node `<tweets>...</tweets>`.

```
<tweet
  user="chanlerone"
  norm_loc="Zeewolde, Netherlands, Zeewolde, Netherlands"
  rep_loc="Zeewolde"
  date="Fri Sep 07 18:40:47 +0000 2012"
  id="244143189617950720"
>
    Ik stem dit keer op meerdere partijen via stembreker.nl.
    Verdeel ook jouw stem. https://t.co/rF6mvoab
</tweet>
```

Figure 1: XML structure of a single Tweet.

The structure in Figure 1 contains a single node, of which the content is the actual text in the tweet. The node itself contains several attributes, which contain meta-information about the tweet. Attribute `user` contains the username, `norm_loc` contains the normalized location, `rep_loc` contains the reported location, `data` contains the timestamp on which the tweet was sent, and `id` contains the unique Twitter ID of the tweet.

4 Distributing the corpus

Since the Twitter Terms of Service state that it is illegal to distribute full-text collections of tweets, a workaround needs to be found. An official workaround is to distribute the list of tweet ids, in addition to potential further annotations that do not come from Twitter, i.e. the normalized location.

Therefore, the tw.py script contains a method that can be used to export a tab-delimited file that contains tweet ids and their normalized location.

```
$ python tw.py -export corpusexport.tab
```

This export file can be distributed without violating the Twitter Terms of Service. To recompile the corpus, the tw.py script contains an import function that will re-download the tweets with the given IDs.

```
$ python tw.py -import corpusexport.tab
```

This feature is not yet implemented.

5 Searching the corpus

The corpus is made persistent in an ad-hoc XML format, and may therefore not be compatible with existing corpus tools. Therefore, the tw.py script contains a function that allows researchers to search in the full text of the tweets by means of regular expressions. Filtering on the basis of the node attributes is not foreseen. However, the results will be returned as a Keyword in Context, with additional columns that contain these meta-data. The results can therefore be filtered afterwards. The search is performed in the XML files that are available in the `tweets` folder at the same file structure level as the tw.py script.

```
$ python tw.py -search "\b[Gg]oulash.+?\b"
```

This feature is not yet implemented.

6 Obtaining the script

The script is being released under the most free Apache2 license. This means that the script can be distributed and modified freely, without any restrictions.

Download the script from http://www.github.com/ruettet/twittercorp.