Capstone Project

Artem Vovk

May 19, 2017

Task 0

- **The Data**

This exercise uses the files named `LOCALE.blogs.txt` where `LOCALE` is the each of the four locales `en_US`, `de_DE`, `ru_RU` and `fi_FI`. The data is from a corpus called HC Corpora. See the About the Corpora reading for more details. The files have been language filtered but may still contain some foreign text.[1] Each entry is tagged with it's date of publication. Where user comments are included they will be tagged with the date of the main entry.

Each entry is tagged with the type of entry, based on the type of website it is collected from (e.g. newspaper or personal blog) If possible, each entry is tagged with one or more subjects based on the title or keywords of the entry (e.g. if the entry comes from the sports section of a newspaper it will be tagged with "sports" subject).In many cases it's not feasible to tag the entries (for example, it's not really practical to tag each individual Twitter entry, though I've got some ideas which might be implemented in the future) or no subject is found by the automated process, in which case the entry is tagged with a '0'.

To save space, the subject and type is given as a numerical code.

Once the raw corpus has been collected, it is parsed further, to remove duplicate entries and split into individual lines. Approximately 50% of each entry is then deleted.

- *Tasks to accomplish*

  1. Obtaining the data - Can you download the data and load/manipulate it in R?
     - Done
  2. Familiarizing yourself with NLP and text mining - Learn about the basics of natural language processing and how it relates to the data science process you have learned in the Data Science Specialization.
     - Done

- *Questions to consider*

  1. What do the data look like?
     - A bunch of text
  2. Where do the data come from?
     - scraped twitter/blogs/news sites
  3. Can you think of any other data sources that might help you in this project?
     - Wikipedia to give context on current events; dictionaries, duh
  4. What are the common steps in natural language processing?
     - tokenization, tagging, clustering, aggregation
  5. What are some common issues in the analysis of text data?
     - spelling, dialects, miscategorization, erroneous inference
  6. What is the relationship between NLP and the concepts you have learned in the Specialization?

[1] You may still find lines of entirely different languages in the corpus. There are 2 main reasons for that: 1. Similar languages. Some languages are very similar, and the automatic language checker could therefore erroneously accept the foreign language text. 2. "Embedded" foreign languages. While a text may be mainly in the desired language there may be parts of it in another language. Since the text is then split up into individual lines, it is possible to see entire lines written in a foreign language.Whereas number 1 is just an out-and-out error, I think number 2 is actually desirable, as it will give a picture of when foreign language is used within the main language.

– lots of data cleanup necessary, data categorization is also a big part of this, modeling the data can lead to interesting conclusion

Task 1

In this exercise, you will use the English database but may consider three other databases in German, Russian and Finnish.

The goal of this task is to get familiar with the databases and do the necessary cleaning. After this exercise, you should understand what real data looks like and how much effort you need to put into cleaning the data. When you commence on developing a new language, the first thing is to understand the language and its peculiarities with respect to your target. You can learn to read, speak and write the language. Alternatively, you can study data and learn from existing information about the language through literature and the internet. At the very least, you need to understand how the language is written: writing script, existing input methods, some phonetic knowledge, etc.

Note that the data contain words of offensive and profane meaning. They are left there intentionally to highlight the fact that the developer has to work on them.

- *Tasks to accomplish*

    1. **Tokenization** - identifying appropriate tokens such as words, punctuation, and numbers. Writing a function that takes a file as input and returns a tokenized version of it.
        - Done with Maxent_Entity_Annotator
    2. **Profanity filtering** - removing profanity and other words you do not want to predict.
        - Need to explore

Tips, tricks, and hints

- **Loading the data in**. This dataset is fairly large. We emphasize that you don't necessarily need to load the entire dataset in to build your algorithms (see point 2 below). At least initially, you might want to use a smaller subset of the data. Reading in chunks or lines using R's readLines or scan functions can be useful. You can also loop over each line of text by embedding readLines within a for/while loop, but this may be slower than reading in large chunks at a time. Reading pieces of the file at a time will require the use of a file connection in R. For example, the following code could be used to read the first few lines of the English Twitter dataset:

```
con <- file("en_US.twitter.txt", "r")
readLines(con, 1) ## Read the first line of text
readLines(con, 1) ## Read the next line of text
readLines(con, 5) ## Read in the next 5 lines of text
close(con) ## It's important to close the connection when you are done
```

See the `?connections` help page for more information.

- **Sampling**. To reiterate, to build models you don't need to load in and use all of the data. Often relatively few randomly selected rows or chunks need to be included to get an accurate approximation to results that would be obtained using all the data. Remember your inference class and how a representative sample can be used to infer facts about a population. You might want to create a separate sub-sample dataset by reading in a random subset of the original data and writing it out to a separate file. That way, you can store the sample and not have to recreate it every time. You can use the rbinom function to "flip a biased coin" to determine whether you sample a line of text or not.

Task 2

The first step in building a predictive model for text is understanding the distribution and relationship between the words, tokens, and phrases in the text. The goal of this task is to understand the basic relationships you observe in the data and prepare to build your first linguistic models.

- *Tasks to accomplish*
    1. Exploratory analysis - perform a thorough exploratory analysis of the data, understanding the distribution of words and relationship between the words in the corpora.
    2. Understand frequencies of words and word pairs - build figures and tables to understand variation in the frequencies of words and word pairs in the data.
- *Questions to consider*
    1. Some words are more frequent than others - what are the distributions of word frequencies?
    2. What are the frequencies of 2-grams and 3-grams in the dataset?
    3. How many unique words do you need in a frequency sorted dictionary to cover 50% of all word instances in the language? 90%?
    4. How do you evaluate how many of the words come from foreign languages?
    5. Can you think of a way to increase the coverage – identifying words that may not be in the corpora or using a smaller number of words in the dictionary to cover the same number of phrases?

Task 3

The goal here is to build your first simple model for the relationship between words. This is the first step in building a predictive text mining application. You will explore simple models and discover more complicated modeling techniques.

- *Tasks to accomplish*

    1. Build basic n-gram model - using the exploratory analysis you performed, build a basic n-gram model for predicting the next word based on the previous 1, 2, or 3 words.
    2. Build a model to handle unseen n-grams - in some cases people will want to type a combination of words that does not appear in the corpora. Build a model to handle cases where a particular n-gram isn't observed.

- *Questions to consider*

    1. How can you efficiently store an n-gram model (think Markov Chains)?
    2. How can you use the knowledge about word frequencies to make your model smaller and more efficient?
    3. How many parameters do you need (i.e. how big is n in your n-gram model)?
    4. Can you think of simple ways to "smooth" the probabilities (think about giving all n-grams a non-zero probability even if they aren't observed in the data) ?
    5. How do you evaluate whether your model is any good?
    6. How can you use backoff models to estimate the probability of unobserved n-grams?

Hints, tips, and tricks

As you develop your prediction model, two key aspects that you will have to keep in mind are the size and runtime of the algorithm. These are defined as:

- **Size**: the amount of memory (physical RAM) required to run the model in R
- **Runtime**: The amount of time the algorithm takes to make a prediction given the acceptable input

Your goal for this prediction model is to minimize both the size and runtime of the model in order to provide a reasonable experience to the user.

Keep in mind that currently available predictive text models can run on mobile phones, which typically have limited memory and processing power compared to desktop computers. Therefore, you should consider very carefully

1. how much memory is being used by the objects in your workspace; and
2. how much time it is taking to run your model. Ultimately, your model will need to run in a Shiny app that runs on the shinyapps.io server.

Tips, tricks, and hints

Here are a few tools that may be of use to you as you work on their algorithm:

- `object.size()`: this function reports the number of bytes that an R object occupies in memory
- `Rprof()`: this function runs the profiler in R that can be used to determine where bottlenecks in your function may exist. The profr package (available on CRAN) provides some additional tools for visualizing and summarizing profiling data.
- `gc()`: this function runs the garbage collector to retrieve unused RAM for R. In the process it tells you how much memory is currently being used by R.

There will likely be a tradeoff that you have to make in between size and runtime. For example, an algorithm that requires a lot of memory, may run faster, while a slower algorithm may require less memory. You will have to find the right balance between the two in order to provide a good experience to the user.