The Amazing Emoji Searcher

Course: Web Information Retrieval

Richard Luong, 2014403075

June 20, 2015

Contents

1 Introduction

This project is the final assignment of the class ”Web Information Retrieval” given by teacher Min ZHANG at Tsinghua University, spring semester 2015. The class is a part of the graduate program ”Advanced Computing” at Tsinghua University.

2 Motivation

The idea of making an emoji search engine came up because combines information in terms of the language we are using with the visual in forms of emojis and presents this in a way that is easy for even a non-technology interesed person to understand.

For example, adding a emoji to a text message can totally change the meaning of the message. Like

```
I will kill you. :)
I will kill you.
```

So my intent is to do a search engine where you just can enter a word (or multiple) and get the 10 most relevant emojis for your query.

3 Related work

There is not much related work on this field. A quick serach on Google Scholar on ”emoji” only gives us 786 results, with none of the top results related to this field. This because emoji (in the form as we have them today) has not been around for a very long time.

4 Emoji

From `www.unicode.org` [2], they definie emojis as:

“Emoji are pictographs (pictorial symbols) that are typically presented in a colorful form and used inline in text. They represent things such as faces, weather, vehicles and buildings, food and drink, animals and plants, or icons that represent emotions, feelings, or activities.”

The name emoji is Japanese and comes from the two words “e” and “moji”, which means picture and character respectively.

4.1 History

The first emoji was created in 1998 by Shigetaka Kurita, a part of the team working on NTT DoCoMo’s mobiel internet platform. The first set included 172 12x12 pixel emojis. When encoded into unicode, these emojis were encoded into two byte sequences, in the private-use range E63E through E757. [7]

In 2010 hundreds of emoji characters were encoded in the Unicode Standard. The core set in Unicode 6.0 consisted of 722 characters.

4.2 Now

The unicode standard now measures up to version 7.0, released in mid-2014. This set is only fully supported by Windows 10. Version 8.0 is still a draft, at the time of this report (June 2015).

Different platforms implement the emoji repesentation differently. One of the most used is made by Apple, and avaliable on all iOS and OS X devices. In 2015, Apple released a new set of emojis including new skin tone modifiers, familijy combinations and new contry flags. These are available from iOS 8.3 and above, OS X 10.10.3 Yosemite and above.

5 Design

My project consist of three parts: crawling, searching/indexing and presenting. I will describe each part more clearly.

5.1 Crawling

A web crawler is a Internet bot that systematically browses the web, for the purpose of web indexing. In this project, the web crawler is used to download the information needed to deliver the relevant results when using the amazing emoji searcher.

5.1.1 Beautiful Soup

The framework used to crawl the websites in this project is Beautiful Soup [5]. It is a pyhton library for pulling data out of HTML and XML files. It also provide pythonic ways to navigate, search and modyify the parse tree.

5.1.2 Sources

The main sources used is the unicode web site, more specifically `http://www.unicode.org/emoji/charts/full-emoji-list.html`. Additional information about each emoji is downloaded from Emojipedia, `http://emojipedia.org/`.

5.1.3 Document structure

The crawling process downloads information about every emoji available from the Full Emoji List provided by the Unicode Consortium and then saves the information to file.

The structure of each emoji / document can be examined in the Appendix.

All the emojis are downloaded and stored to files using the pickle command in python. In total it is 1281 emojis.

5.2 Searching

Next step when all the documents are downloaded is to actually build the search engine. This also includes indexing, query parsing and scoring of the results.

5.2.1 Whoosh

The framework used for the search engine is Whoosh, a full text indexing, search and spell checking library written in Python. [6]

The main reason I chose this library is because I feel most comfortable working in Python at the moment.

5.2.2 Indexing

In order to start indexing the documents, we need to define the schema of the index. The schema lists the fields in the index, what kind of information it contains.

In this project the schema looks like this:

```
index=ID(stored=True),
name=TEXT(analyzer=stem_ana, stored=True, field_boost=100.0),
annotations=KEYWORD(field_boost=50.0),
related_to=TEXT(analyzer=stem_ana),
known_as=TEXT(field_boost=50.0, analyzer=stem_ana),
browser=STORED,
description=TEXT(analyzer=stem_ana, field_boost=20.0))
```

The fields are boosted differently in order to give the document correct score. The actual score is not the most important, it is the relative score between them that matters the most.

The boosting values are empirically set. The most important field is the name field, as it describes the emoji in a very exact way. Annotations and other names of the emoji are important to as they describe the emoji in other words than the exact name, hence the second highest boost. I find the full text description more related to the query than the emojis that are related to the emoji.

The actual repesentation of the emoji (browser) is a part of the index, because i wanted to show a representation of the emoji, when displaying the results.

5.2.3 Stemming

All text fields are stemmed using StemmingAnalyzer. It is a pre-packaged analyzer that combines a tokenizer, lower-case filter, optional stop filter, and stem filter.

- Tokenizing is dividing the query into smaller tokens. In the default StemmingAnalyzer, it is a simple RegEx tokenizer.
- Lower-case filter turns the query to lowercase.
- The stop filter, filters out stop words. Stop words are words that are too common to index.
- The stem filter removes suffixes fron the tokens using the Porter temming algorithm. Stemming attempts to reduce multiple forms of the same root word.

5.2.4 Scoring

The list of results documents are sorted by score. The Whoosh library contains multiple different kinds of scoring algorithm. In this case I've used the default one, BM25F.

BM25F is a version of the classic Okapi BM25 ranking function that take document structure and anchor text into account. The BM25 algorithm ranks a set of document based on the query terms appearing in each document, regardless of the relationship between the query terms in the document.

Given a query Q, containing keywords $q_1, ..., q_n$, the BM25 score of a document D is:

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

where $f(q_i, D)$ is q_i's term frequency in the document D, $|D|$ is the length of the document D in words, and *avgdl* is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2.0]$ and $b = 0.75$. $\text{IDF}(q_i)$ is the IDF (inverse document frequency) weight of the query term q_i.

5.3 Presenting

The search engine is presented as a web application, mainly written in HTML. As web server framework I have used Flask, a microframework written in Python. [4]

When a user enters a query on the website, he or she sends a request to the server. The query gets parsed by a MultiFieldParser, a parser that searches multiple fields at once. In this case, all the fields in the schema, except browser and id, are included. The query is also variated with Variations, a term class that automatially searches morphological variations of the given word in the same field. In return will get the top 10 results.

The top 10 results are listed under the search bar, sorted by score. The top result will also be the emoji shown on top of the website.

If a user click on a result, he or she will be taken to the specific page of that emoji.

If no results were found a small alert box will notify the user about this.

5.3.1 Bootstrap

In order to make a beatuiful website, with as little work as possible, I've used Bootstrap for the styling. [1]

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

6 Results

The result of this project can be examined on `http://emoji-search.herokuapp.com/`.

7 Discussion

The Amazing Emoji Searcher is a project that fulfills its purpose as a search engine for emojis. As every project out there, there are both advantages and drawbacks with it.

7.1 Advantages

One of the main advantages with The Amazing Emoji Searcher is that it is specialized for just emojis and give results which the user expects. This is why The Amazing Emoji Searcher performs better than the general search engines as Google and Yahoo. It also delivers information that relates to the users query.

Another advantage is that it not only search for an exact match in the name of the emoji. This is the main drawback of other emoji searchers, such as Emojipedia and Apple Character Viewer keyboard. When searching for "gun", The Amazing Emoji Searcher manages to find the pistol emoji. The same query on the Apple keyboard results in no hits.

7.2 Drawbacks

The main drawback is that The Amazing Emoji Searcher can't capture the semantic meaning of a query. It can only display results that includes the terms in the query. This can sometimes lead to very misleading results.

For example, a search on "toy" delivers a poddle and a pistol as results. Not suitable for children at all.

8 Future work

The extensions of this project could be to implement semantics in the search engine. A very interesting work has been done by the engineers at Instagram [3], that tried to use machine learning on emojis. The authors managed find words that worresponds to the specified emoji, grouping them into clusters. Adding this to this project would allow the users to search for more emojis.

This project could extend the normal emoji keyboards, by allowing the users to select an emoji based on a search query than choosing from a list. The user often knows which emoji he or she wants, and selecting from a list of 10 emojis is way easier than doing it from 1281 different ones.

References

[1] Bootstrap. Bootstrap, 2015. [Online; accessed 11-June-2015].

[2] Unicode Consortium. Unicode emoji, 2015. [Online; accessed 11-June-2015].

[3] Thomas Dimson. Emojineering part 1: Machine learning for emoji trends, 2015. [Online; accessed 11-June-2015].

[4] Flask. Flask, 2015. [Online; accessed 11-June-2015].

[5] Beautiful Soup. Beautiful soup, 2015. [Online; accessed 11-June-2015].

[6] Whoosh. Whoosh, 2015. [Online; accessed 11-June-2015].

[7] Wikipedia. Emoji — wikipedia, the free encyclopedia, 2015. [Online; accessed 11-June-2015].

Appendix

Emoji document structure

Each emoji document as the following structure.

`index`
: Index for the emoji, as describe on the unicode website.

`code`
: The corresponding unicode code for the emoji

`browser`
: The actual characted representation of the emoji. Platform-dependent. Stored as a character

`bw`
: Black and white representation of the image. Stored as base64 encoded image, for ease of displaying on a website.

`apple`
: Representation of the Emoji from the AppleColorEmojiFont on iOS and OS X platforms. Stored as base64 encoded image, for ease of displaying on a website.

`andr`
: Representation of the Emoji from the Android platforms. Stored as base64 encoded image, for ease of displaying on a website.

`twit`
: Representation of the Emoji from Twitter. Stored as base64 encoded image, for ease of displaying on a website.

`gmail`
: Representation of the Emoji from Google. Stored as base64 encoded image, for ease of displaying on a website.

`wind`
: Representation of the Emoji on Windows platforms. Stored as base64 encoded image, for ease of displaying on a website.

`name`
: The name of the emoji.

`version`
: Which version of unicode the emoji first was included.

`default`
: Default emoji style. Can either be a text representation or emoji representation.

`annotations`
: All emoji characters are annotated in english based on the Unicode CLDR data. It is a project to provide locale data in XML format for use in computer applications.

`emojipedia_url`
: URL for the entry on the Emojipedia website. Used for crawling the description of the emoji, related emojis and other names of the emoji.

`desc`
: Description of the emoji, as described on Emojipedia.

`related_to`
: A list of related emojis, as described on Emojipedia.

`known_as`
: A list of other names of the emoji, as described on Emojipedia.