LANGuage IDentification

João Augusto Costa Branco Marado Torres

December 11, 2024

License

Contents

1 Phrases to Numeric Representation

I was really confused about this, but it really is not that complicated.

Before starting to write any code, I wanted to know how can I transform a phrase like the one you are reading right now into `1`s and `0`s because the computer isn't smart enough to know English. So I went to my friend and asked it some questions. It thought me various ways of achieving what I wanted, and between those options I picked the easiest to understand/implement.

Table 1: Rainbow model vocabulary

ID	Word
0	red
1	orange
2	yellow
3	green
4	cyan
5	blue
6	violet

1.1 Bag of Words

The idea it's to build a **vocabulary** which is a list of unique words.

Everyone (including us) has its own vocabulary composed by words of the languages we speak.

Our AI model needs a vocabulary too so he can understand some words. Those words might not exist in our vocabulary.

Let's say our model's vocabulary are the words in English for the 7 colors in the rainbow.

Each word in the vocabulary will have a numerical identifier.

It makes sense for us for that identifiers to start at `0` and increment by `1` for each word.

Now our model can represent our phrases into something it understands.

How?

The question you just asked is represented as a zero column matrix with 7 rows.

The model will try to find words he knows from a phrase and represent it as a column matrix with rows the same as the amount of words in the vocabulary. Each entry represents a word in the vocabulary, here is why it's a good idea to identify the words as I said. The value of each entry can be a simple boolean that represent if a word is present in the phrase or not, or the amount of times the word appears in the phrase. The value really represents how important is the that word in the phrase.

The phrase "How?" does not contain any words know by the model.

The phrase "The colors in the *Guiné-Bissau* flag are red, yellow, green and black in the star." would be represented by the matrix $\begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^T$ and the phrase

"What came first, the orange fruit or the orange color?" could be represented by $\begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ if we count the amount of occurrences of the word in the phrase.

The way I implemented the BoW was by reading a file and collect every single word from it separated by white spaces, dashes, punctuation, parenthesis or quotation marks and then spit to `stdout` a formated vocabulary, so you can pipe it later or write it to a file. Words are case-sensitive, and it's possible to create the vocabulary in alphabetic order.

My implementation can be found in `/src/commands/vocab/bow.zig`.

During the conversation, "tokenization" was mentioned, and it might be cool for you to take a look at it.

1.2 Other ways

- TF-IDF;
- One-Hot Encoding of Words;
- Embedding-like Approach;
- Sub-word tokenization.