Tokenization and datasets

Gustave Cortal

Summary

Regular expression

ELIZA, one of the first chatbot

```
User: I am unhappy.
ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User: I need some help, that much seems certain.
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User: Perhaps I could learn to get along with my mother.
ELIZA: TELL ME MORE ABOUT YOUR FAMILY
User: My mother takes care of me.
ELIZA: WHO ELSE IN YOU FAMILY TAKES CARE OF YOU
User: My father.
ELIZA: YOUR FATHER
User: You are like my father in some ways.
```

ELIZA, one of the first chatbot

- Uses pattern matching to recognize phrases like “I need X” and translate them into outputs like “What would it mean to you if you got X?”
- Pattern-based methods play a crucial role in NLP
- Famous tool for describing text patterns: the regular expression

Regular expression (regex)

What are regular expressions?

A sequence of characters that define a search pattern, primarily used for string matching and manipulation.

Commonly used symbols

- `.` (Dot): Matches any single character.
- `\d`: Matches any digit.
- `\w`: Matches any word character (alphanumeric & underscore).
- `\s`: Matches any whitespace character.
- `*`, `+`: Match 0 or more, 1 or more occurrences of the preceding element, respectively.
- `^`, `$`: Start and end of a string, respectively.

ELIZA's pattern recognition

Regex example in ELIZA

`"^[Ii] (do not|don't) know"`

Detailed breakdown

- `^[Ii]`: Matches "I" in both uppercase and lowercase at the start of a sentence
- `(do not|don't)`: Captures either "do not" or the contraction "don't"

Example interaction

- User: "I don't know what to do."
- ELIZA: "Why do you think you don't know what to do?"

Tokenization

What is a word?

Example 1

Hey!! How are you? *:)*

Example 2

I do *uh main-* mainly business data processing.

Example 3

They are studying because *they* want to succeed.

Example 4

We love the *cats* of Jean, but we don't like this *cat*.

Tokenization: principles

- Tokenization is splitting text into individual words or *tokens*.
- Essential for NLP tasks
- Multiple challenges:
  - Different delimiters: spaces, punctuation
  - Contractions: "can't" $\rightarrow$ "can not"
  - Special cases: dates, numbers, URLs, hashtags, email addresses

Tokenization: basic example

Input

"Natural language processing enables computers to understand human language."

Tokenized output

Natural, language, processing, enables, computers, to, understand, human, language, .

Tokenization: dealing with contraction

Input

"I can't believe it's already 2023!"

Tokenization with contraction

I, can't, believe, it's, already, 2023, !

Tokenization with expansion

I, can, not, believe, it, is, already, 2023, !

Tokenization: complex example

Input

"Dr. Smith's email, dr.smith@example.com, isn't working since 01/02/2023; try reaching out at (555) 123-4567 in San Francisco."

Tokenized output

Dr., Smith's, email, ,, dr.smith@example.com, ,, isn't, working, since, 01/02/2023, ;, try, reaching, out, at, (, 555,), 123-4567, in, San Francisco, .

Tokenization: methods

Rule-based approach

Use predefined rules, like splitting by spaces or punctuation.

Machine learning approach

Learn from data to handle complex cases

Library support

NLTK, SpaCy

Tokenization: Byte-Pair Encoding (BPE)

Learn the tokenization

Instead of defining tokens as words, or as characters, we can use our data to automatically induce sets of tokens that include tokens smaller than words, called *subwords*

Training and testing

NLP algorithms learn some patterns from one corpus (*a training set*) and then use these patterns to make decisions about a separate corpus (*a test set*)

Tokenization: Byte-Pair Encoding (BPE)

Problem

How to deal with unknown words?

Unknown word as composed of known subwords

Every unknown word like *lower* can be represented by some sequence of known subword units, such as *low* and *er*

Tokenization: Byte-Pair Encoding (BPE)

Initialization

The BPE token learner begins with a vocabulary that is the set of all individual characters

Algorithm

Examines the training corpus, chooses the two symbols that are most frequently adjacent (say 'A', 'B'), adds a new merged symbol 'AB' to the vocabulary, and replaces every adjacent 'A' 'B' in the corpus with the new 'AB'. It continues to count and merge, creating new longer and longer character strings, until k merges have been done

Applying BPE on an example

Corpus vocabulary with frequencies

```
5 l o w
2 l o w e s t
6 n e w e r
3 w i d e r
2 n e w
```

Initial vocabulary

Unique characters: l, o, w, d, e, i, n, r, s, t

Most frequent pair (e, r)

Merge 'e' and 'r' into 'er'. New vocabulary: l, o, w, d, e, i, n, r, s, t, er
Updated corpus:

```
5 l o w
2 l o w e s t
6 n e w er
3 w i d er
2 n e w
```

Next frequent pair (n, e)

Merge 'n' and 'e' into 'ne'. New vocabulary: l, o, w, d, e, i, n, r, s, t, er, ne Updated corpus:

```
5 l o w
2 l o w e s t
6 ne w er
3 w i d er
2 ne w
```

Sequence of merges

1. Merge (ne, w). New vocabulary: d, e, i, l, n, o, r, s, t, w, er, ne, new
2. Merge (l, o). New vocabulary: d, e, i, l, n, o, r, s, t, w, er, ne, new, lo
3. Merge (lo, w). New vocabulary: d, e, i, l, n, o, r, s, t, w, er, ne, new, lo, low
4. Merge (new, er). New vocabulary: d, e, i, l, n, o, r, s, t, w, er, ne, new, lo, low, newer

To tokenize a text, always apply the merge rules **in the order they were learned during training**

This fixed, sequential order is what makes the tokenization **deterministic**

Language diversity

Beyond English

Current trends often focus on English, overlooking the necessity of testing algorithms on a diverse range of languages

Varieties of language

Most languages have multiple dialects or varieties influenced by regions or social groups

Variation in genre

Algorithms encounter text from various genres: newswires, books, scientific articles, Wikipedia, religious texts, spoken conversations, etc

Demographic influence

The demographic characteristics of writers or speakers (age, gender, race, socioeconomic class) impact the linguistic properties of texts

Temporal influence

Language evolves over time (Old French vs. Current French)

Datasheet

A datasheet specifies the properties of a dataset:

- **Motivation:** Why was the corpus collected, by whom, and who funded it?
- **Situation:** When and in what situation was the text written/spoken? For example, was there a task? Was the language originally spoken conversation, edited text, or social media communication?
- **Language variety:** What language was the corpus in?
- **Speaker demographics:** What was, e.g., the age or gender of the text's authors?
- **Collection process:** How big is the data? If it is a subsample, how was it sampled? Was the data collected with consent? How was the data pre-processed, and what metadata is available?
- **Annotation process:** What are the annotations, what are the demographics of the annotators, how were they trained, and how was the data annotated?
- **Distribution:** Are there a copyright or other intellectual property restrictions?

Definitions

Definitions

Text normalization

The process of transforming text into a more uniform format, facilitating easier processing

Lemma

The base form of a word. Lemmatization involves reducing inflected words to their lemma form (e.g., "running" to "run")

Corpus

A structured collection of texts. It serves as a dataset for various language processing tasks

Token

A single, meaningful element in a text. Tokenization is the process of splitting text into tokens

Definitions

Training corpus

A dataset used to train a NLP model. It teaches the model to recognize patterns and make predictions

Test corpus

A separate dataset used to evaluate the model performance

Vocabulary

The set of unique tokens a model uses for processing

Exercices

Exercices

- Using regular expressions, create a tokenizer handling some complex examples (∼30m)
- Using regular expressions, create an ELIZA-like system (∼30m)
- Implement Byte-Pair Encoding from scratch (∼1h)

Pseudo-code for BPE

```
function BYTE-PAIR ENCODING(strings C, number of merges k)
    V ← all unique characters in C # initial set of tokens
    for i ← 1 to k do # merge tokens k times
        tL, tR ← Most frequent pair of adjacent tokens in C
        tNEW ← tL + tR # make new token by concatenating
        V ← V + tNEW # update the vocabulary
        Replace each occurrence of tL, tR in C with tNEW
        # update the corpus
    return V
```