CSC 412/2506:
Probabilistic Learning and Reasoning

Week 10 - 1/2 Embeddings I

Michal Malyska

University of Toronto

Overview

- Embeddings
  - Bag of Words
  - Term Frequency - Inverse Document Frequency
- NLP concepts

Machine Learning on Text

So far in the class (and probably all other machine learning classes) we have been operating under the assumptions that all the inputs to our models are numbers. Today we will focus on another kind of input - text. In particular, Natural Language.

- How is natural language represented?

NLP concepts

Some names we will be using throughout this lecture:

- **Token** - A single "atom" of text, usually a word
- **Document** - A complete datapoint of text
- **Span** - A subset of a document, a group of Tokens
- **Vocabulary** - A list of all possible tokens

NLP Tasks

What do we want to do with text?

- Classification (Documents, Spans, Tokens)
  - Hate speech detection
  - Spam filtering
  - Social Media drug adverse effect identification

NLP Tasks

What do we want to do with text?

- Generation (Question Answering, Summarization, Free-text generation, . . .)
  - Translating natural language into SQL queries
  - "Hey siri what is the weather like?"
  - Chatbots
  - Talk to a transformer

NLP Tasks

What do we want to do with text?

- Regression (Essay scoring, like count prediction)
  - How many retweets will this tweet get?
- Information Extraction
  - Who are the people mentioned in this text?
- Document Retrieval
  - Google search
  - Automatic literature review

Spam

We begin by revisiting a familiar example from earlier lectures: determining whether an email is spam or not.

- What kind of task is it ?
- How would you approach it?

More formally: Consider a set of observations $(x_i, y_i)_{i=1:N}$ where $y_i = 1$ means datapoint i was spam, and x_i is the text of the email.
Our goal is to create / learn a function f_θ such that $f_\theta(x_i) = p(y_i|x_i)$

Heuristics

Perhaps it is possible to find some heuristics that work:

- If x_i contains any prescription medication name $\hat{y}_i = 1$
- If x_i is is mostly capital letters $\hat{y}_i = 1$
- If x_i contains "Make **Amount** every **Time Period**" $\hat{y}_i = 1$

```python
import re

SPAM = 1
MEDICINE_LIST = ["Acetylcysteine", "Apixaban","..."]

def spam_classifier_heuristic(text: str) -> bool:
    """
    Heuristics to classify whether a string is a spam email or not
    """
    for medicine in MEDICINE_LIST:
        if text.contains(medicine):
            return SPAM
    if text.upper() == text:
        return SPAM
    if re.match(r"((M|m)ake)\s*((\w*)\s*)*(\$\d*)\s*((per|for\sjust)\s*(week|day|year))|(weekly|daily|monthly)", text) is not None:
        return SPAM
    else:
        return 1 - SPAM
```

Can we learn simple heuristics?

To apply any kind of learning algorithms we have seen before we need to convert x into a numerical representation h.

- Given a vocabulary $V_{j=1:M}$, determine h such that: $h_j = 1$ whenever token j from the vocabulary is present in x.
- Each datapoint x is represented by an M dimensional vector of 0's and 1's
- How do we determine the vocabulary?
- Just list and count all the words in all of the documents, and then only keep the top M.
- We have numerical features, so just plug them as input into any "standard" algorithm: Logistic Regression

Bag of Words

This simple binary representation is called a **(binary) Bag of Words**.

- What is included in the representation h of x?
- What if we care about more than just the presence / abscence of a specific word?
- We could just include the count of each word turning h from a vector of 1's and 0's into a vector of counts.
- What about phrases? "Polyethylene Glicol"?

N-Grams

Instead of considering single words as entries in a vocabulary, perhaps we would want to consider phrases.
An **N-gram** is a contiguous sequence of N tokens from a given text.
Under $N = 1$; also called unigrams

$$V = (\text{"This"}, \text{"is"}, \text{"a"}, \text{"sentence"})$$

$$\text{"This is a sentence"} = (1, 1, 1, 1)$$

$$\text{"A sentence"} = (0, 0, 1, 1)$$

Under $N = 2$; also called bigrams

$$V = (\text{"This is"}, \text{"is a"}, \text{"a sentence"})$$

$$\text{"This is a sentence"} = (1, 1, 1)$$

$$\text{"A sentence"} = (0, 0, 1)$$

Common words

Some words are incredibly common, but do not contribute a lot in terms of distinguishing between texts.

- Virtually any text in English will contain "the" or "a"
- Should we include those in the vocabulary?
- Can we learn which words to include in the vocabulary?
- We can better represent documents by the **relative frequency** of words in them.
- Note: in practice we often remove some words from all text and ignore them completely. You will see those referred to as **stopwords**

Term Frequency

We can represent the "count" (**Term Frequency**) of a word in many different ways:

- Raw count of times it is present in x: **BoW**
- Binarized count of times it is present in x: **binary BoW**
- Count of times it is present in x divided by number of tokens in x
- Raw count scaled by number of other terms (not count of) in x
- Log of the raw count

Term Frequency example

$V = (\text{"This"}, \text{"is"}, \text{"a"}, \text{"sentence"}, \text{"another"}, \text{"not"}, \text{"Yet"})$
$x_1 =$"This is a sentence. This is another sentence."
$x_2 =$"This is not a sentence. Yet another not a sentence"

Inverse Document Frequency

Just like with term frequency, we can represent the "relative prevalence" (**Inverse Document Frequency**) of a word in many different ways:
Denote $N_j = \sum_{i=1}^{N} I\left(V_j \in x_i\right)$ count of datapoints that include j-th word in the vocabulary.

- $$\frac{1}{N_j}$$
- $$\frac{N}{N_j}$$
- $$\log(\frac{N}{N_j})$$

You can think of it as a scaling factor for each of the words in the vocabulary.

TF-IDF

By combining Term Frequency with Inverse Document Frequency we can measure how common the word is in a particular datapoint relative to other documents.

$$\text{TF-IDF}(x) = TF(x) \times IDF(x)$$

TF-IDF example

$V = (\text{"This"}, \text{"is"}, \text{"a"}, \text{"sentence"}, \text{"another"}, \text{"not"}, \text{"Yet"})$
$x_1 =$"This is a sentence. This is another sentence."
$x_2 =$"This is not a sentence. Yet another not a sentence"

Embeddings

All of the methods we talked about can be used to generate numerical representations of whole documents, by the use of just the word occurences, and simple functions. We say that h_i is the **embedding** of x_i, and we call $g(x) = h$ an embedding function. We can then re-frame our problem of learning $f_\theta(x) = y$ as:

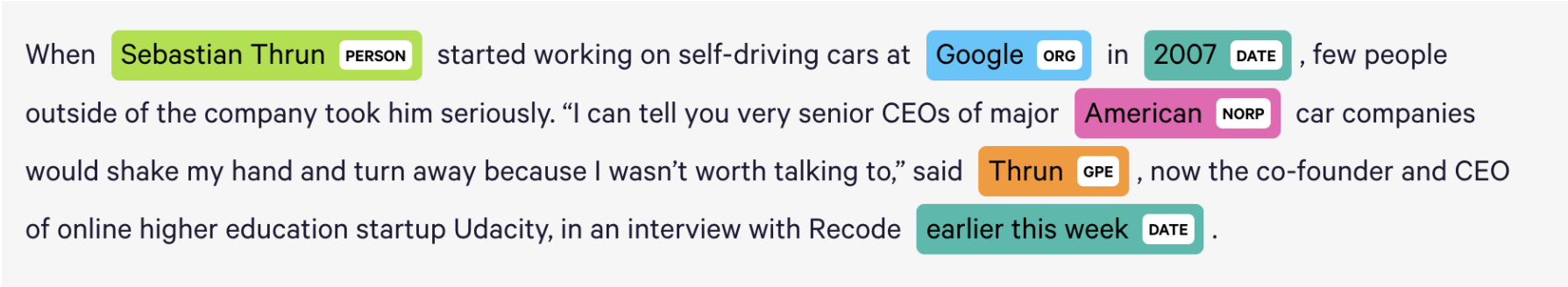
$$f_\theta(x) = c_{\theta_1}(h) = c_{\theta_1}(g_{\theta_2}(x)) = y$$

Where c_θ is any classification / regression function, and g_θ is an embedding function.

- Embeddings are not restricted to documents.
- How could we embed an image?
- What if our datapoint consists of an image and text?

Token classification

Sometimes we care about something more granular than just the whole document. Perhaps we want to identify each parts of text that correspond to certain concepts:



- What should we get a represantation of?
- How could we do this?

[1] from SpaCy: https://explosion.ai/demos/displacy-ent