**Prerequisites**

- multisets (basics)

## Set-of-words revisited: adding counts

I have a horrible confession to make: the previous unit contains a lie. The set-of-words model isn't actually used in practice. Instead, it's a bag-of-words model. The bag-of-words model works like the set-of-words model, except that every word type also gets a number to indicate how often it occurs in the text. As you can imagine, this immediately fixes the second problem we identified: websites that mention a word in the user query many times should be preferred over those that mention it only once or twice. Mathematically, this step from a record of the words in a website to a record of how often each word occurs corresponds to the step from sets to **multisets**.

### Keeping track of counts

A multiset is like a set, except that it can contain one and the same element multiple times. The count is usually written write after the element, separated by a colon. For instance, $\{a : 5, b : 0\}$ denotes a multiset that contains 5 instances of $a$ and 0 instances of $b$. Sometimes, we only refer to a multiset by its name without listing any specific counts. In this case, I will prefix multisets with a subscripted $M$ to distinguish them from normal sets. So whereas $S$ denotes a set, ${}_MS$ denotes a multiset.

**Example 1** The set-of-words model converts the mini-text *Only John could like John* (modulo capitalization) to the set $\{\text{only}, \text{john}, \text{could}, \text{like}\}$. The sentence *If police police police police police, then police police police police police* is converted to $\{\text{if}, \text{police}, \text{then}\}$.
The bag-of-words model gives different results. The first sentence is mapped to the multiset $\{\text{only} : 1, \text{john} : 2, \text{could} : 1, \text{like} : 1\}$. The second one yields $\{\text{if} : 1, \text{police} : 10, \text{then} : 1\}$.

**Exercise 1** Consider the two sentences below.

- John shaved.
- John shaved John.

Are their sets identical? Are their multisets identical?

**Exercise 2** This exercise is just for fun. Can you figure out all the possible interpretations of the sentence *If police police police police police, then police police police police police*?
A few hints:

- In English, *police* can be a noun or a verb (*to police something or somebody*).
- English allows for reduced relative clauses, e.g. *the man that Mary saw* can be reduced to *the man Mary saw*.

Just like sets, multisets aren't limited to unigrams. One can just as well count bigrams, trigrams, and so on.

```
from collections import Counter

def ngram_list(text, n):
    return zip(*[text[pos:] for pos in range(n)])

def ngram_set(text, n):
    return set(ngram_list(text, n))

def ngram_multiset(text, n):
    return Counter(ngram_list(text, n))

sentence1 = ["only", "john", "could", "like", "john"]
sentence2 = ["if", "police", "police", "police", "police", "police",
             "then", "police", "police", "police", "police", "police"]

print("Set for\n{}\n is\n{}".format(" ".join(sentence1), ngram_set(sentence1, 2)))
print("Multiset for\n {}\n is\n{}".format(" ".join(sentence1), ngram_multiset(sentence1, 2
print()
print("Set for\n{}\n is\n{}".format(" ".join(sentence2), ngram_set(sentence2, 2)))
print("Multiset for\n{} is\n{}".format(" ".join(sentence2), ngram_multiset(sentence2, 2)))
```

**Example 2** Given the sentence *if police police police police police then police police police police police*, its bigram multiset is

$$\{\text{if police} : 1, \text{police police} : 8, \text{police then} : 1, \text{then police} : 1\}.$$

**Exercise 3** Construct the trigram multiset for *if police police police police police then police police police police police*.

Since multisets aren't limited to invididual words in language-related applications, one often encounters the term **type** instead for the members of the multiset. The **tokens**, on the other hand, are the counts of the types.

**Example 3** The bigram multiset

$$\{\text{if police} : 1, \text{police police} : 8, \text{police then} : 1, \text{then police} : 1\}.$$

contains four distinct types, which are *if police*, *police police*, *police then*, and *then police*. The overall number of tokens is 11 as the type *police police* has 8 tokens and the other 3 types have 1 token each.

## Counts for frequencies

One major advantage the proper bag-of-words model has over the set-of-words model is that the counts provide direct information about frequency, which can be used in various ways. Given a multiset ${}_M S$, the frequency of some type $s \in_M S$ is the

number of tokens of $s$ divided by the total of all tokens. In mathematical notation, the frequency of $s$ is

$$\text{freq}(s) := \frac{{}_MS(s)}{|{}_MS|}$$

where ${}_MS(s)$ is the count of $s$ in ${}_MS$, and $|{}_MS|$ is the total of all tokens.

We can replace the absolute counts in a bag of words by relative frequencies. Strictly speaking this is no longer a multiset, since the values for a multiset can only be natural numbers. It's impossible for a multiset to contain an item .73 times. In addition, relative frequencies destroy information: an item with frequency .5 could have occurred 5 times, 500, or 5 trillion. It's impossible to tell unless one knows the total size of the corpus that the counts were collected from. In practice, it is best to keep the bag of words untouched and store the frequencies separately. But just for notation, it is often nicer to have frequencies instead of counts.

**Example 4** Let's go back to the multiset

$${}_SM := \{\text{if police} : 1, \text{police police} : 8, \text{police then} : 1, \text{then police} : 1\}.$$

The total number of counts is 11. So after we replace counts by frequencies, ${}_MS$ becomes $\left\{\text{if police} : \frac{1}{11}, \text{police police} : \frac{8}{11}, \text{police then} : \frac{1}{11}, \text{then police} : \frac{1}{11}\right\}$ (which is no longer a multiset).

```
import matplotlib.pyplot as plt

print("N-gram frequencies for \"{}\"".format(" ".join(sentence2)))

def freq_conversion(ngram_multiset):
    total = sum(ngram_multiset.values())
    return {" ".join(key): val/total for key, val in ngram_multiset.items()}

def plotting_conversion(ngram_multiset):
    return list(zip(*sorted(ngram_multiset.items(), key=lambda x: x[1], reverse=True)))

unigrams = plotting_conversion(freq_conversion(ngram_multiset(sentence2, 1)))
bigrams = plotting_conversion(freq_conversion(ngram_multiset(sentence2, 2)))
trigrams = plotting_conversion(freq_conversion(ngram_multiset(sentence2, 3)))

for ngrams in [unigrams, bigrams, trigrams]:
    labels = ngrams[0]
    data = ngrams[1]
    plt.figure(figsize=(4,4), dpi=100)
    plt.bar(range(len(data)), data, color='#336699')
    plt.ylim((0,1))
    plt.xticks(range(len(data)), labels, rotation=80)
    plt.tick_params(axis='both', which='major', labelsize=12)
    # plt.text(str(label)+"-gram")
    plt.show()
```

**Exercise 4** Construct a bag of words from the following text of four sentences, then compute the relative frequency of each word:

- John misses Mary
- Mary misses Misses Chives
- I hate chives with a passion
- I love passion fruit

It is up to you to decide what should be treated as different words. You do not have to ignore capitalization in every case if you think that keeping capitalization gives a better result. Similarly, you may posit words that contain spaces. But take the opportunity to reflect on whether a computer program could easily make the linguistic distinctions you are making.

Counts and frequency information can be used in various ways, for example for search engines. One can measure the relevance of a text for a given search query based on how much of the text consists of the words in the search query.

**Example 5** The sentence *Only John thinks John thinks John likes John* corresponds to the multiset

$$_{M}S := \{\text{only} : 1, \text{john} : 4, \text{thinks} : 2, \text{likes} : 1\}.$$

With frequencies instead of counts this is

$$_{M}S := \{\text{only} : 0.125, \text{john} : 0.5, \text{thinks} : 0.25, \text{likes} : 0.5\}.$$

So the relevance score to the query *john* would be 0.5.

**Exercise 5** Suppose the sentence were *Only John thinks he thinks he thinks he likes himself*. What would be the score of *john* in this case? Is that a problem?

For practical purposes, both absolute counts and relative frequency can provide important information. Absolute counts by themselves are not enough because a document that mentions *rules* 10 times might still not be a good match for "rules and representations" if the whole document contains 10,000 words. On the other hand, a document containing just the phrase *rules schmules* would also be a horrible match despite 50% of it being mentions of *rules*. There are many different strategies for optimising matches, and we will not go into details here. However, it is instructive to see how queries consisting of multiple words can be handled mathematically.

## Multi-set sums and scalar multiplication for multi-word queries

Suppose a student has heard that the term *metaphysics* is somehow related to Aristotle and would like to learn more about that. So they go to their search engine of choice and enter the search query *Aristotle metaphysics*. How can the search engine provider make sure they will serve the user with a good list of matches?

Let us assume that the search engine already has absolute unigram, bigram, and trigram counts for each website. We first need to determine which of these counts are most useful for the query *Aristotle metaphysics*. Well, the trigram counts don't

matter since the query itself only consists of two words. This leaves us with a choice between unigram and bigram counts. But the search query wasn't meant as a bigram by the user, in that case they would have searched for *"Aristotle metaphysics"* (notice the quotation marks). Moreover, very few texts would actually contain the string *Aristotle metaphysics* (although some might contain *Aristotle's metaphysics*). For all these reasons, it makes more sense to treat *Aristotle* and *metaphysics* as unigrams and find the website that has the highest combined number of tokens for these two types — which is surprisingly easy with multisets.

First, suppose that the search engine has a large index of unigram multisets, where each unigram multiset encodes the unigram counts for a specific website. Since we only care about the counts for *Aristotle* and *metaphysics*, we will construct two new multisets that encode for each website how many tokens of *Aristotle* and *metaphysics* it contains.

**Example 6** To keep the example simple we assume that our search engine has indexed only four websites, which we refer to as $w_1$, $w_2$, $w_3$, and $w_4$. In doing so, it has only recorded counts for a few words. Each one of the four indeces created by the search enginge is represented as a multiset below:

$$\begin{aligned} w_1 &:= \{\text{Aristotle} : 9, \text{ethics} : 3, \text{metaphysics} : 0, \text{Plato} : 6, \text{Sokrates} : 0\} \\ w_2 &:= \{\text{Aristotle} : 1, \text{ethics} : 0, \text{metaphysics} : 9, \text{Plato} : 5, \text{Sokrates} : 0\} \\ w_3 &:= \{\text{Aristotle} : 4, \text{ethics} : 9, \text{metaphysics} : 5, \text{Plato} : 8, \text{Sokrates} : 2\} \\ w_4 &:= \{\text{Aristotle} : 0, \text{ethics} : 7, \text{metaphysics} : 3, \text{Plato} : 9, \text{Sokrates} : 0\} \end{aligned}$$

From this we construct an *Aristotle* multiset ${}_MA$ and a *metaphysics* multiset ${}_MP$ (naming it ${}_MM$ would be too confusing).

$$\begin{aligned} {}_MA &:= \{w_1 : 9, w_2 : 1, w_3 : 4, w_4 : 0\} \\ {}_MP &:= \{w_1 : 0, w_2 : 9, w_3 : 5, w_4 : 3\} \end{aligned}$$

As you can see, each multiset has at its types the websites in the search engine's index, and the counts are carried over directly from the corresponding multisets above. For instance, ${}_MA$ could be represented more abstractly as

$${}_MA := \{w_i : w_i(\text{Aristotle}) \mid w_i \text{ is in the search engine's index}\}.$$

**Exercise 6** Construct similar multisets for *ethics* and *Plato*.

At this point we have two multisets over websites that tell us how well each website fits the individual terms *Aristotle* and *metaphysics*. However, we still need to combine them into some kind of aggregate score. This is easily accomplished with **multiset sum**. That sounds like a fancy operation, but it's little more than adding up the counts for each element across multisets.

**Example 7** Here are the multisets ${}_MA$ and ${}_MP$ that we computed before.

$$\begin{aligned} {}_MA &:= \{w_1 : 9, w_2 : 1, w_3 : 4, w_4 : 0\} \\ {}_MP &:= \{w_1 : 0, w_2 : 9, w_3 : 5, w_4 : 3\} \end{aligned}$$

Their multiset sum is

$$
\begin{aligned}
{}_MA \uplus_M P &= \{w_1 : 9+0, w_2 : 1+9, w_3 : 4+5, w_4 : 0+3\} \\
&= \{w_1 : 9, w_2 : 10, w_3 : 9, w_4 : 3\}
\end{aligned}
$$

The best match according to this measure is the website $w_2$, closely followed by $w_1$ and $w_3$.

**Exercise 7** Calculate the multiset sum of the two multisets you constructed for *ethics* and *Plato* in the previous exercise.

While multiset sum offers an easy way to convert the multisets produced according to the query into a single score for each website, it is also a bit too simplistic. Multiset sum is *commutative*, which means that the order of arguments does not matter: ${}_MA \uplus_M P =_M P \uplus_M A$. But the same arguably isn't true for the order of unigrams in a search query. The fact that the student typed *Aristotle metaphysics* rather than *metaphysics Aristotle* suggests that *Aristotle* may be more important. In line with this interpretation we can use **scalar multiplication** to assign ${}_MA$ a greater weight relative to ${}_MP$.

**Example 8** The values of ${}_MP$ and ${}_MP$ are listed here once more for the sake of easier reference:

$$
\begin{aligned}
{}_MA &:= \{w_1 : 9, w_2 : 1, w_3 : 4, w_4 : 0\} \\
{}_MP &:= \{w_1 : 0, w_2 : 9, w_3 : 5, w_4 : 3\}
\end{aligned}
$$

Suppose that ${}_MA$ should be about 50% more important than ${}_MP$. We can express this by multiplying ${}_MA$ by 3 and ${}_MP$ by 2. Since $\frac{3}{2} = 1.5$, this assigns ${}_MA$ a 50% higher weight than ${}_MP$.

$$
\begin{aligned}
3 \otimes_M A &= \{w_1 : 3 \times 9, w_2 : 3 \times 1, w_3 : 3 \times 4, w_4 : 3 \times 0\} \\
&= \{w_1 : 27, w_2 : 3, w_3 : 12, w_4 : 0\} \\
2 \otimes_M P &= \{w_1 : 2 \times 0, w_2 : 2 \times 9, w_3 : 2 \times 5, w_4 : 2 \times 3\} \\
&= \{w_1 : 0, w_2 : 18, w_3 : 10, w_4 : 6\}
\end{aligned}
$$

As before, we combine the two multisets via multiset sum.

$$
\begin{aligned}
3 \otimes_M A \uplus 2 \otimes_M P &= \{w_1 : 27 \uplus 0, w_2 : 3 \uplus 18, w_3 : 12 \uplus 10, w_4 : 0 \uplus 6\} \\
&= \{w_1 : 27, w_2 : 21, w_3 : 22, w_4 : 6\}
\end{aligned}
$$

As you can see, $w_1$ is now the best match, followed by $w_3$ and $w_4$. Considering that $w_1$ does not contain any mention of metaphysics, we actually ended up making things worse.

**Exercise 8** Suppose the user entered *ethics Plato Aristotle*. Use scalar multiplication to assign appropriate weights to each word (it is up to you to decide

what is appropriate). Then compute the multiset sum and determine which website is the best fit for the query.

## Recap

- Multisets are the formal foundation of bag-of-words models.
- Each text can be compressed into a multiset to keep track of counts for relevant words.
- Counts can be combined and manipulated with multiset sum and scalar multiplication.
- Counts are often converted to percentages, but the result is no longer a multiset.