Distributional Feature Representations

Jonathan May

September 13, 2024

1 Where did those features come from?

In the last notes, we started with **f**, but nothing much was said about where **f** (the features) came from except I made allusions to hand-defining the features much as was done for linear models. However, a very simple feature template definition could be 'these are the words that appear in the input' (e.g. a bag of words). This is in keeping with the idea of being less prescriptive about the feature templates and allowing the data to speak for itself. Let's say we do this for the first (k) words in the bag. That is, the first d/k features represent the identity of the first word, and so on until the last d/k represents the kth.

What should each of those d/k features be? If we considered every word to be distinct from every other word we could create a *one-hot* vector such that each word is orthogonal to each other, but that seems intuitively incorrect and would make it very hard to learn. This would also make d very large – for each word there would be V (vocabulary size) features, most of which are 0 (a 'one hot' vector). Intuitively it would be nice if we had a fixed set of features for each word, such that words that are similar are close to each other (where 'close' means 'cosine of the angle between their vectors is small'). Otherwise 'cat' and 'feline' are as different from each other as 'cat' and 'rutabaga.'

1.1 Distributional Methods

One way to understand word similarity is by thinking about collections of how words are used and making (perhaps by hand) graphs that relate 'is-a' relationships (e.g. 'yellow' and 'red' are each kinds of 'color') or 'part-of' relationships (e.g. 'wheel' and 'transmission' are part of 'car'). Another is by top-down properties (e.g. part of speech, number, language of origin, age). A different way to understand word similarity is based on a famous quote by linguist John Rupert Firth (1957): "You shall know a word by the company it keeps." That is, words are similar if the words they are near in large collections of text are similar. This is known as the Distributional Hypothesis! We will use co-occurrence to obtain features. Further more, we'll do so in a way so that the number of features we have is much smaller than our vocabulary.

Intuition from Zelig Harris (another linguist) in 1954: "oculist and eye-doctor occur in almost the same environments...thus we say they are synonyms."

Here's another example:

```
A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.
```

What do you think tesgüino is?

1.2 Word co-occurrence matrices and mutual information

Let's try this first, comparing documents and words:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Notice the usage patterns of 'fool' and 'clown' vs 'battle' and 'soldier.' Notice also the similarity of some of the plays.

Important difference from thesaurus-based approaches now; we're losing the ability to distinguish between senses of the same word form (there are ways to try to get these back but we won't cover them here and the 'hard decision' approach doesn't work that well...but we will revisit this when we discuss contextualized representations).

We can also make such a table for smaller contexts, such as a four-word window.

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

In this table the number of times a word in the column was seen within four words of the word in the row is listed in the cell.

In reality this table is $|V| \times |V|$ but the vast majority of cells are 0 (very sparse). Notice again how similar words have similar vector patterns.

This is so because of two co-occurrence phenomena:

- Syntagmatic (first-order) association (surface similarity): sets of words all occur near each other, somewhat interchangeably. E.g. 'wrote', 'book', and 'poem' all tend to occur near each other so they are likely to have similar patterns (example: "Whether a book or a poem, what Jane Austen wrote will live for generations.").
- Paradigmatic (second-order) association (paradigm similarity): words don't necessarily occur near each other but nevertheless do have similar neighbors. E.g. 'wrote', 'said', 'remarked' all share a 'paradigm' of words they occur near. Example: "The

candidate remarked that the troops were important." "The candidate said he valued the importance of the troops." "The candidate wrote that the troops mattered a lot to him."

It has been observed that a narrow co-occurrence window (1-3) will tend to give words with similar *syntactic* properties more similar vectors and with a wider window (4-10) more *semantic* and not necessarily syntactic similarity. Think 'orange/apple/lemon/carrot' for the former and 'kill/death/killing/killed' for the latter. These are not hard and fast rules.

Not all co-occurring words are equally informative! Consider 'the' and 'of' which occur many times very frequently with other words. It's better to ask which words are particularly informative. Specifically, if words occur more frequently than they do by chance, this is interesting to us[1]. We specifically define *pointwise mutual information* for words w_1, w_2:

$$PMI(w_1, w_2) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \tag{1}$$

If w_1 and w_2 are IID, we'd expect $P(w_1, w_2) = P(w_1)P(w_2)$. If this is so, then $PMI = 0$. If the words co-occur more likely than expected, i.e. $P(w_1, w_2) > P(w_1)P(w_2)$, then $PMI > 0$. If they occur less frequently $PMI < 0$. This last element is often ignored; we don't really know what it means to be some degree of 'unrelated' plus the resolution needed to detect events *less* likely than the product of two events necessitates very large corpora. Typically we instead study *positive pointwise mutual information*:

$$PPMI(w_1, w_2) = \max(\log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}, 0)$$

Here's a worked example. Using the table above of frequency counts f_{ij} for word i in context of word j, we can calculate joint probability, word probability, and context probability, as:

$$\begin{aligned} p_{\text{joint}}(i,j) =& \frac{f_{ij}}{\sum_w \sum_c f_{wc}} \\ p_{\text{word}}(i) =& \frac{\sum_c f_{ic}}{\sum_w \sum_c f_{wc}} \\ p_{\text{context}}(j) =& \frac{\sum_w f_{wj}}{\sum_w \sum_c f_{wc}} \end{aligned}$$

	$p(w,c)$					$p(w)$
	computer	data	pinch	result	sugar	
apricot	0	0	.05	0	.05	.11
pineapple	0	0	.05	0	.05	.11
digital	.11	.05	0	.05	0	.21
information	.05	.32	0	.21	0	.58
$p(c)$	.16	.37	.11	.26	.11	

[1]See also TF*IDF, another way to formulate the same idea

PMI(information, data) $= \log_{\frac{.32}{.37 \times .58}} = .57$

Here are all PMIs:

	computer	data	pinch	result	sugar
apricot	0	0	2.25	0	2.25
pineapple	0	0	2.25	0	2.25
digital	1.66	-.56	0	-.07	0
information	-.8	.57	0	.47	0

To get PPMI, replace the negative values with 0.

1.3 Cosine similarity

A nice number for characterizing the closeness of two vectors is the *cosine* of these vectors. Each word is represented as a vector in $|V|$-space. If the angle they make is small, the cosine is close to 1. Cosine is just a normalized dot-product. Simple dot product isn't a great way to calculate closeness, because longer vectors (i.e. with high values in some dimensions) will lead to larger dot product. Cosine normalizes this:

$$\cos(a, b) = \frac{a \cdot b}{|a||b|}$$

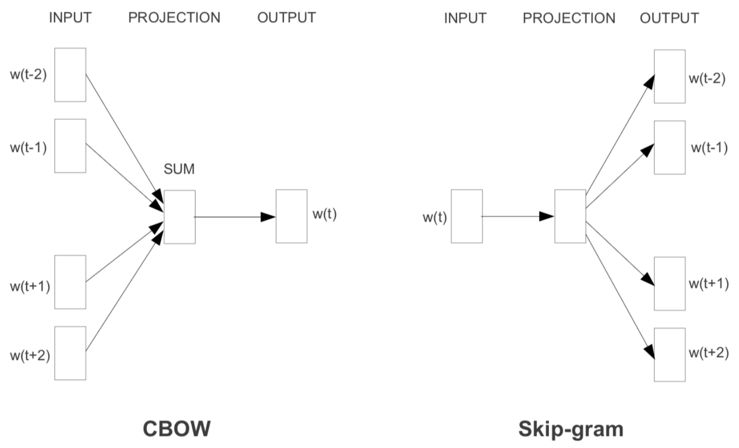
where

$$|x| = \sqrt{\sum_i x_i^2}$$

1.4 Neural(-inspired) distributional representations

An issue with PPMI is the vectors are very sparse and the dimensions very large. We previously saw embeddings were lower-dimensional dense representations of words. We want embeddings for words to be aware of the contexts in which these words occur. One way to do this is inspired by the neural networks (multi layer perceptrons) we have already been looking at. Mikolov's *skip-gram* tries to predict the identity of a word given features derived from another word around it. Specifically, it contains a word embedding matrix E and an output matrix O but note there is no hidden matrix and no nonlinear function. Given an input word w_i we can look up the features in E by representing w_i as a one-hot vector and multiplying it by E. Alternatively we can just consider E_{w_i}, the w_ith row of E. If we want to predict some nearby word w_o, the logit for w_o is simply $(E_{w_i})O_{w_o}$. Given some text, training data is formed by taking w_o to be any word within some range r before or after w_i. An alternative framework called the *continuous bag-of-words* sums together the embeddings of context words within r of w_i to predict it. In other words, for $r = 2$, the logit for w is $((E_{w-2} + E_{w-1} + E_{w+1} + E_{w+2})O)_w$.

We still evaluate using cross-entropy as the objective, which comes down to:



$$H_{(p,q)} = -\sum_{w'_o} p(w'_o|w_i) \log q(w'_o|w_i)$$

But assuming only the observed w_o have any probability according to p.[2] And then $q(w_o|w_i)$ given E and O is:

$$\frac{e^{(E_{w_i})O_{w_o}}}{\sum_{w'_o \in V} e^{(E_{w_i})O_{w'_o}}} \tag{2}$$

which is our familiar softmax. Then $\log q(w_o|w_i)$ is

$$E_{w_i}O_{w_o} - \log \sum_{w'_o \in V} e^{(E_{w_i})O_{w'_o}} \tag{3}$$

Unlike in the MLP examples we worked with before, where we had a small number of output label classes, here V can be very large (200,000 is not uncommon if we're just talking about space-separated word types).

A trick that avoids having to calculate this huge denominator is *negative sampling*.[3] Consider the question 'did w_i, w_o appear in proximity in the data set?' We can try to build a model of this by letting the score of the comparison of some w_i, w_o be precisely the $E_{w_i}O_{w_o}$ we're already calculating. We can further imagine a two-class problem, class 1 if the answer is 'yes' and class 0 if it's 'no.' The sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$ is softmax for two variables.

Let's optimize for class 1 for observed pairs and class 0 ($= 1 - \sigma(x)$) for unobserved pairs. Further, for every observed pair, we can sample some k[4] random words, drawn according to the unigram distribution raised to 3/4 power. Our objective is then:

$\log \sigma(w_i, w_o) + \sum_{w'_o \in R} \log(1 - \sigma(w_i, w'_o))$

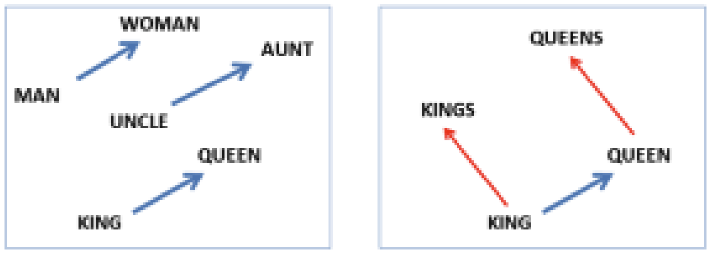
[2]Strictly speaking these wouldn't be one-hot, but we can consider them all in one batch and then normalize, or we can consider a distribution of n context items with $1/n$ probability for each.

[3]Originally from `https://arxiv.org/abs/1310.4546` but see also explained in `https://arxiv.org/pdf/1402.3722.pdf`.

[4]In practice, from 5–25

where R is the sample set. It is basically the same as the above but with binary softmax and a much smaller summation.

These models, along with some techniques for training them very quickly, are known collectively as *Word2Vec* (w2v). Some nice properties observed with them is that one could do *vector math*; the vector formed by subtracting *big-biggest* is very similar to that formed from *small-smallest*. To this end, the W2V authors created an analogy test set. To evaluate vectors, you consider an analogy like "brother:sister::grandson:granddaughter." You calculate *grandson*+(*sister-brother*). If the closest embedding to that vector is *granddaughter* the relationship has been captured. The relationship types are shown below, as are some results.



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Model	Semantic-Syntactic Word Relationship test set	
Architecture	Semantic Accuracy [%]	Syntactic Accuracy [%]
RNNLM	9	36
NNLM	23	53
CBOW	24	64
Skip-gram	55	59

The last SOTA results I have seen on this test[5] were 74.0 on semantic accuracy and 60.0 on syntactic accuracy, in 2015. The task was not a major focus after that, perhaps because neural language modeling became more of interest. Additionally, this method of analysis has attracted a great deal of criticism.[6] Further, it was shown that the relationships between gender-specific roles like 'king' and 'queen' extended to historical tendencies such as affiliating 'homemaker' and 'nurse' with 'female' and 'maestro' or 'protege' with 'male'.

[5]From USC: `https://aclanthology.org/W15-1513.pdf`
[6]`https://aclanthology.org/2020.conll-1.29/`