Representing text for AI

Why is text different to other types of data?

Syntactically similar words often have totally different meanings

E.g. 'closet' vs 'closest'

But syntactically different words can have exactly the same meaning!

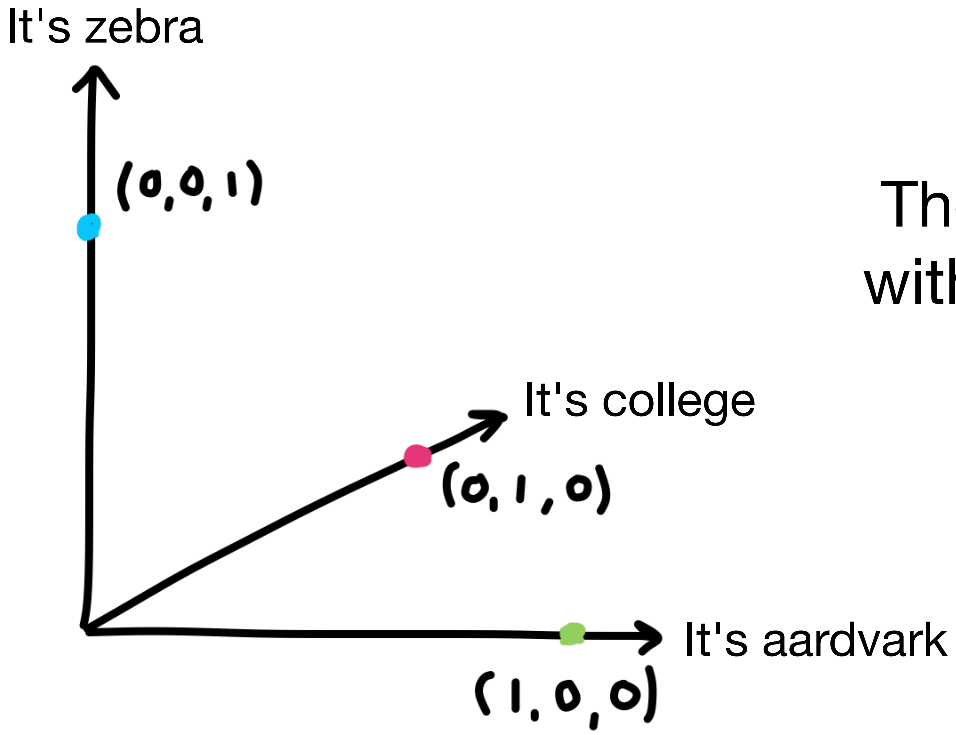
E.g. 'dog' vs 'canine'

There is no obvious numerical representation of text, so we can't feed it directly to algorithms that process numbers

We need a new kind of representation

1-hot vector representation

We need a numerical representation of each element of our text so that we can pass it to our model



E.g in a vocabulary of those 3 words

What if we indexed each element?

Then we could represent each of them as a binary vector with a 1 in the position of that index and zeros everywhere else

This is called a 1-hot encoding

$$\text{aardvark} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \text{college} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{zebra} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

With this representation, each of our input features are whether or not the input word is the word corresponding to that element

What's wrong with 1-hot encodings of words?

What's more similar, aardvarks and college or aardvarks and zebra's?

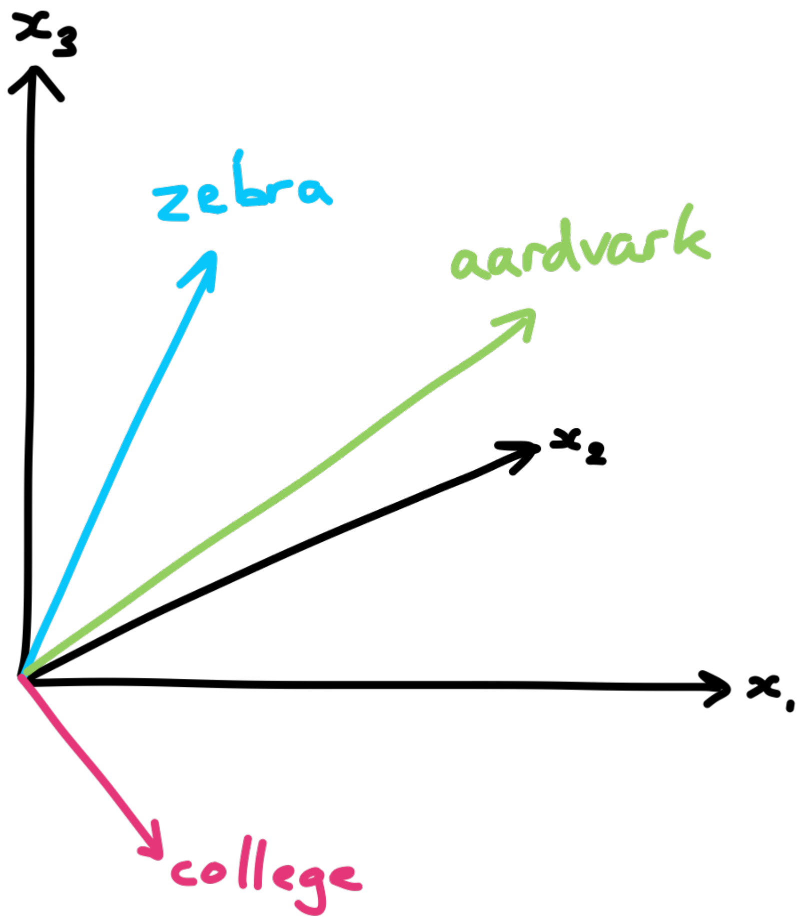
Well aardvarks and zebra's are both animals

But any conventional distance function will tell us that the 1-hot encodings of the words are equally dissimilar

E.g. dot product similarity

$$\overset{\text{aardvark}}{\begin{bmatrix}1\\0\\\vdots\\0\\0\end{bmatrix}} \cdot \overset{\text{college}}{\begin{bmatrix}0\\\vdots\\1\\0\\0\\\vdots\\0\end{bmatrix}} = 0 \qquad \overset{\text{aardvark}}{\begin{bmatrix}1\\0\\\vdots\\0\\0\end{bmatrix}} \cdot \overset{\text{zebra}}{\begin{bmatrix}0\\0\\\vdots\\0\\1\end{bmatrix}} = 0$$

Dense word representations



$$\text{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{college} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$\text{zebra} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Dense representations of words are not binary, and are not 1-hot

Each element in a dense word embedding can represent something much more complex, and hopefully useful

Dense representations of words can give much richer representations of words

Comparing these dense word vectors can give us a much more representative similarity between words, based on their semantic meaning

Word2Vec visualisation

Word embeddings

To implement word embeddings, we can use an embedding matrix

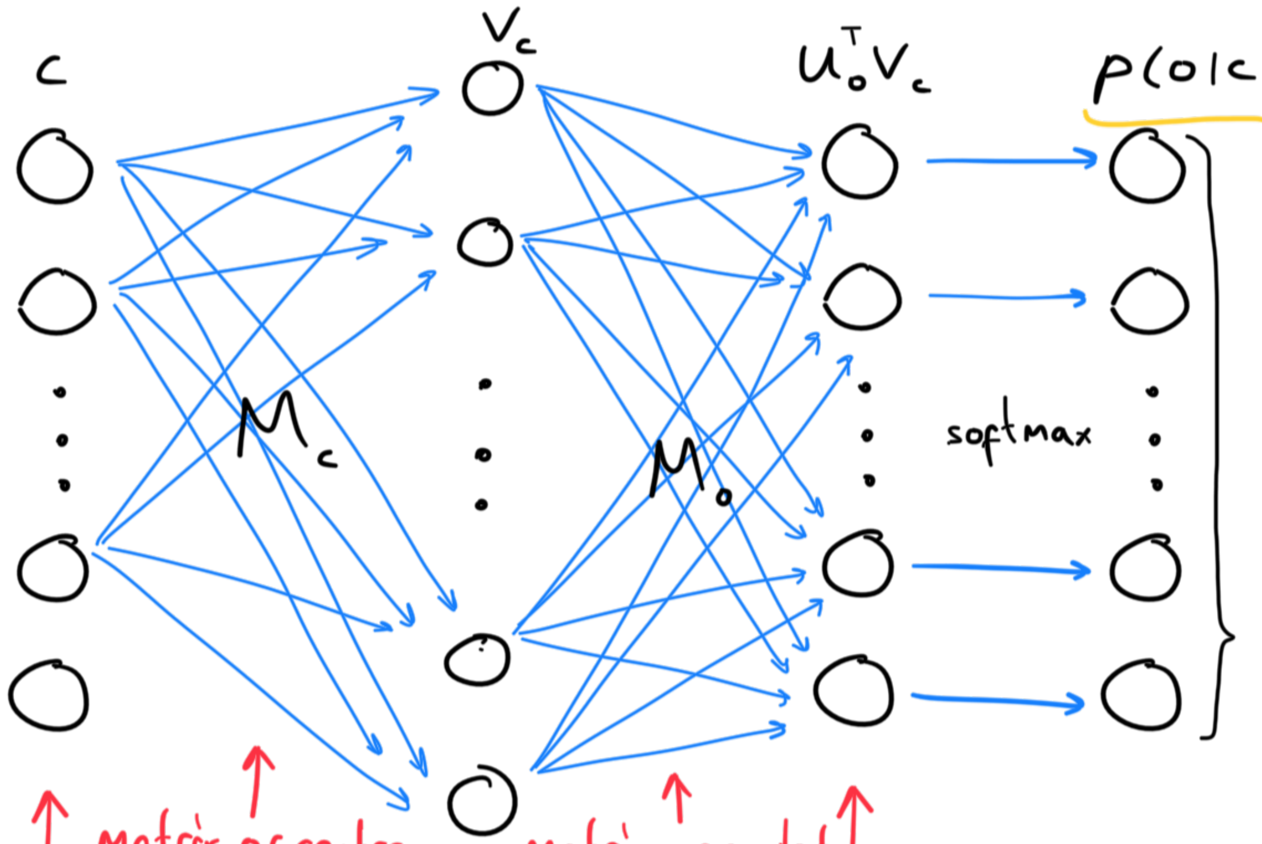
An embedding matrix has a word embedding in each column

$$E = \overset{\xrightarrow{\text{\# words in vocab}}}{\begin{bmatrix} | & | & & | & | \\ e^{(1)} & e^{(2)} & \cdots & e^{(m-1)} & e^{(m)} \\ | & | & & | & | \end{bmatrix}} \Big\downarrow \text{embedding dim} \qquad e^{(i)} = \begin{bmatrix} e^{(i)}_1 \\ \vdots \\ e^{(i)}_n \end{bmatrix}$$

By pre-multiplying a 1-hot encoding with the embedding matrix, we can slice out the embedding for that word

$$Ey = \begin{bmatrix} | & | & & | & | \\ e^{(1)} & e^{(2)} & \cdots & e^{(m-1)} & e^{(m)} \\ | & | & & | & | \end{bmatrix} \overset{\text{1-hot aardvark}}{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}} = e^{(1)} = \overset{\text{dense aardvar}}{\begin{bmatrix} 0.2 \\ 1.4 \\ \vdots \\ 0.3 \\ -0.6 \end{bmatrix}}$$

Learning word embeddings



$$p(o|c) = \underbrace{\frac{e^{u_o^T v}}{\sum_v e^{u_o^T v}}}_{\text{predict outside words given centre word}}$$

$$P = \begin{bmatrix} p(o_1|c) \\ \vdots \\ p(o_v|c) \end{bmatrix} = \text{softmax}(M_o M_c c)$$

binary vector to index centre word $\in \mathbb{R}^v$

Matrix of centre word embeddings $\in \mathbb{R}^{d \times v}$

centre word embedding

Matrix of context word embeddings $\in \mathbb{R}^{v \times d}$

dot product