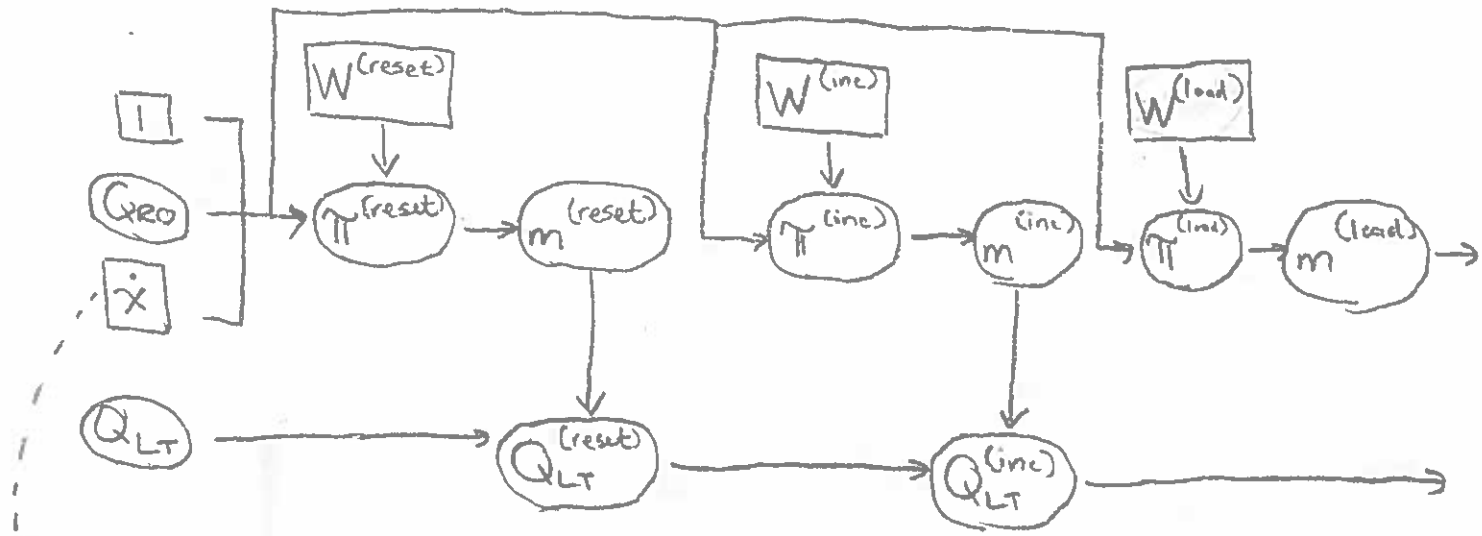
Word Vectors

① As we use increasingly complex architectures for natural language processing, a major problem manifests:



if our vocabulary contains V words, then all the weight matrices will be $V \times H$ if we represent each word as a one-hot vector.

② This can make training difficult and slow. Suppose the vocabulary has 100K words. If our hidden state Q_{LT} has size 100, then $W^{(reset)}$ has 10 million parameters to learn.

With a large enough vocabulary, it's also entirely possible that these weight matrices won't fit in memory.

Word Vectors

(3) It'd be nice to start with vector representations that more compactly encoded each word of the vocabulary.

One could imagine a representation that represents a word in terms of a selection of attributes, e.g.

$$\text{elephant} = \begin{bmatrix} 1.8 \\ 0.4 \end{bmatrix} \quad \text{dolphin} = \begin{bmatrix} 0.7 \\ 1.5 \end{bmatrix} \quad \text{truck} = \begin{bmatrix} 1.7 \\ -3.1 \end{bmatrix}$$

where each vector is:

$$\begin{bmatrix} \text{size} \\ \text{intelligence} \end{bmatrix}$$

Elephants are large and relatively intelligent, while trucks are large but rather dumb.

---

(4) One way to learn these representations is with a simple neural network. It had better be simple, because we'll need to use one-hot vectors as its input (lest we enter some infinite recursion).

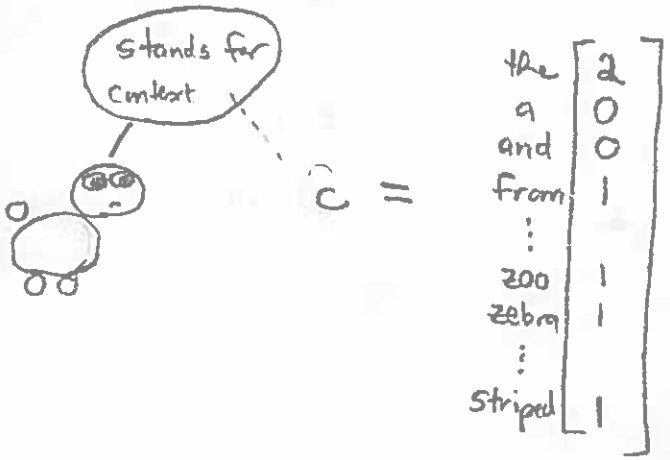
Word Vectors

(5) A popular approach is to train a neural network to solve "fill-in-the-blank" statements, e.g.

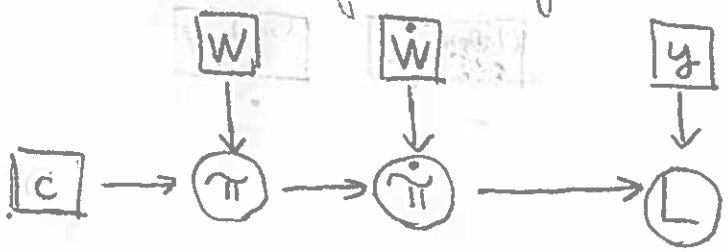
the striped zebra _____ from the zoo

A successful response could be "escaped", while other responses like "horse" or "delighted" should be considered less likely.

(6) The input to the network is the sum of one-hot vectors for the words in the context, e.g.



(7) The network itself is quite simple:



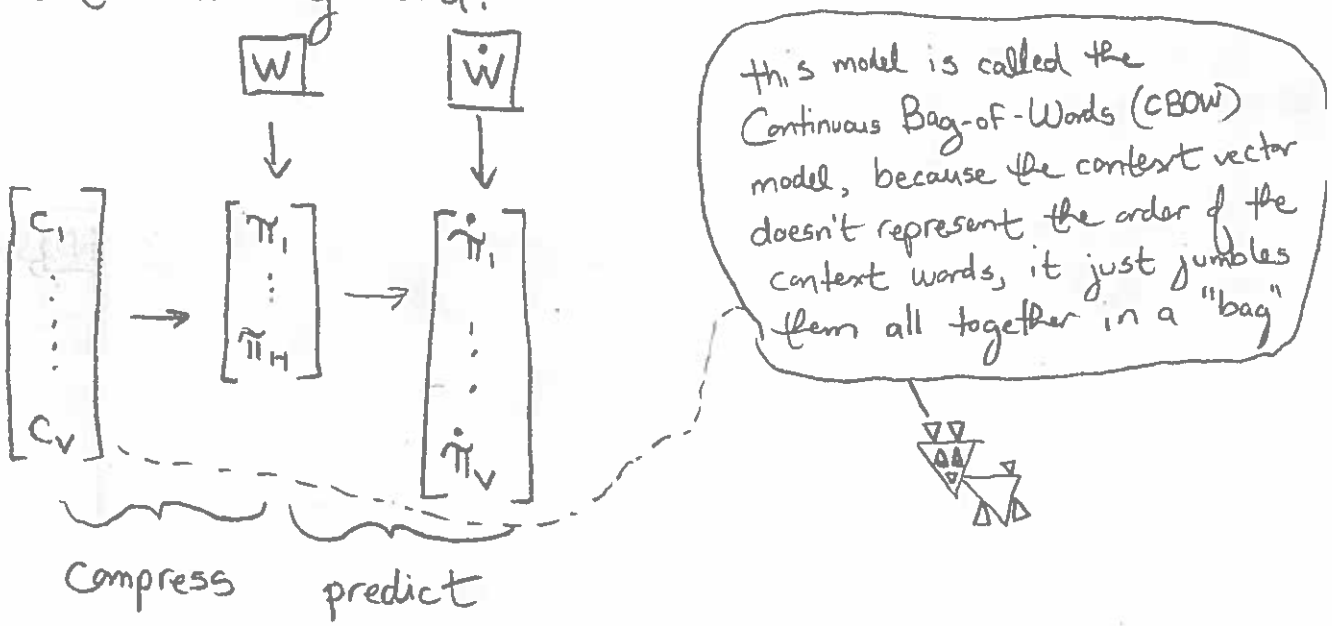
where: $\pi = W^T c$

$\dot{\pi} = \dot{W}^T \pi$

$L = -\log\left(y^T \text{softmax}(\dot{\pi})\right)$

⑧ Observe that the dimension of W is $V \times H$, π is $H \times 1$
$\dot{W}$ is $H \times V$, $\dot{\pi}$ is $V \times 1$.

Essentially this simple network needs to learn to compress large context vectors (of size V) into a compact vector (of size $H \ll V$), which can be used to predict the missing word.

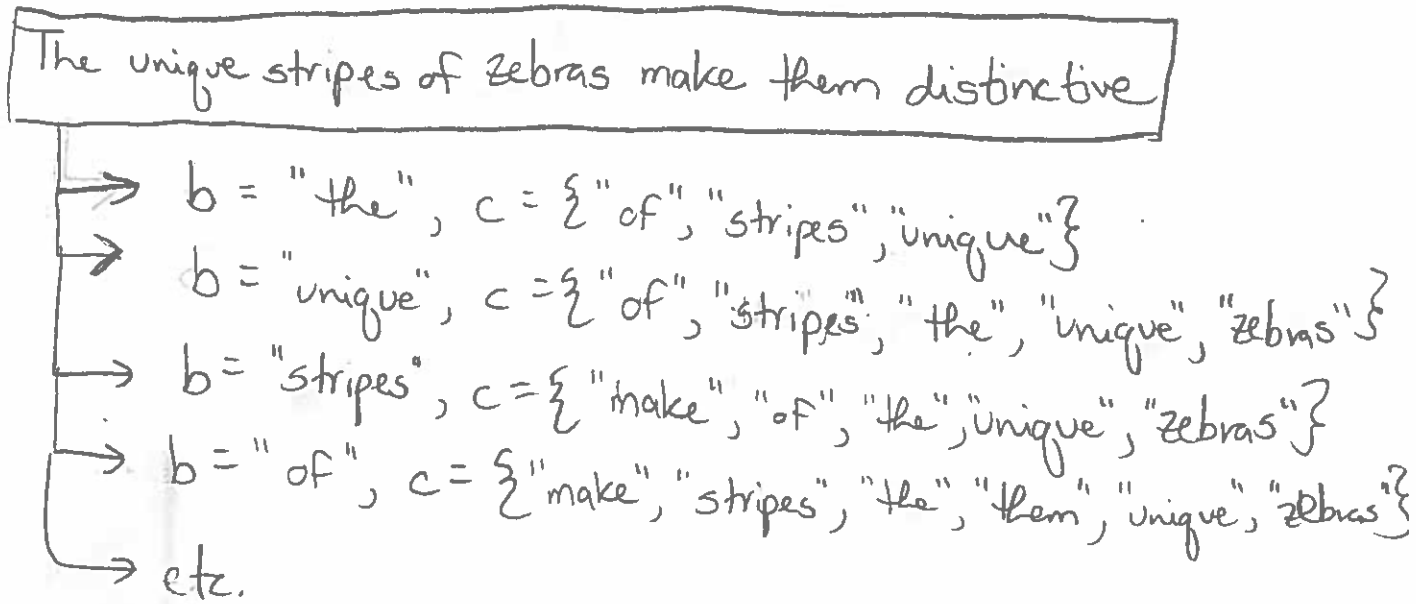


⑨ Interestingly, no activation function is ever applied.

This is a design decision to keep the model extremely simple so it can be trained on billions of word/context pairs.

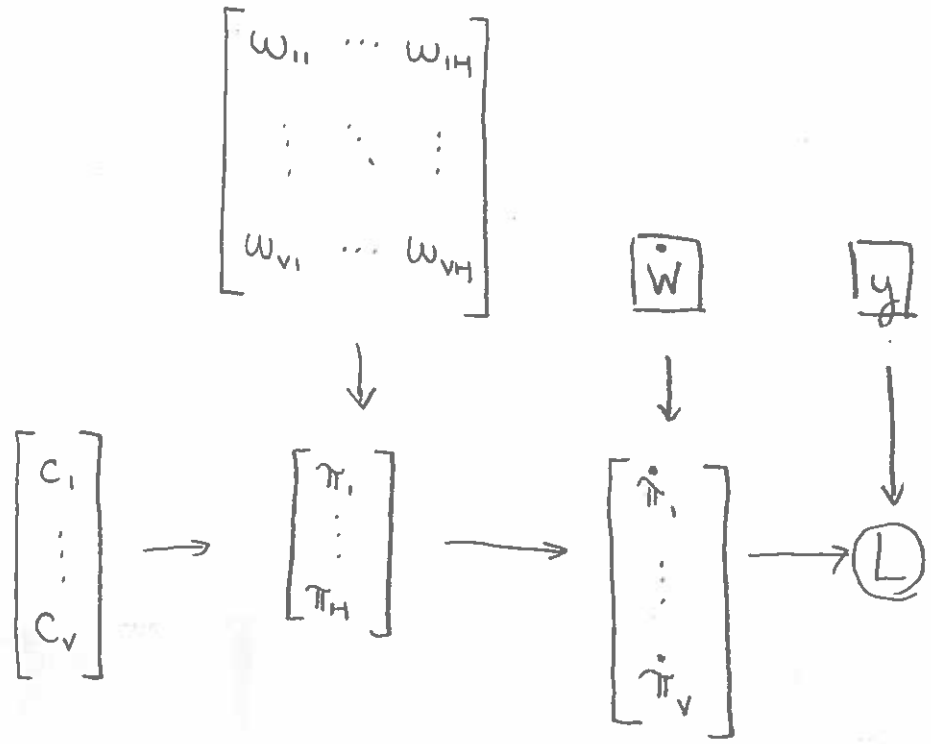
Word Vectors

(10) Billions? Why yes. Notice that it's easy to get training data for this task just by scraping documents from the web.

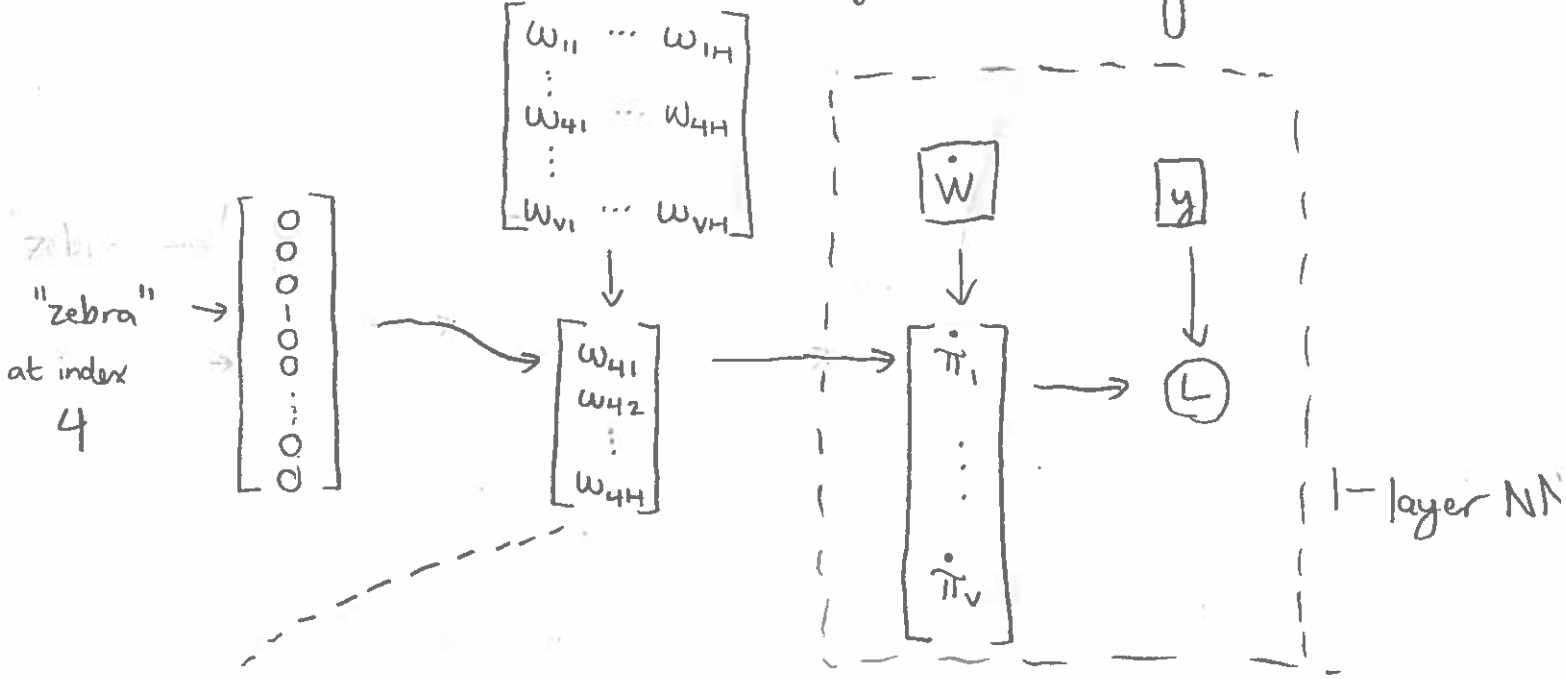


(11) So we've trained a network. What then? I thought we wanted vector representations of words, not a classifier for fill-in-the-blank questions.

(12) Well, let's take a closer look at what the trained network looks like:



(13) What if we provide a single word as an input context? Then c would be a one-hot representation of that word:



The vector π needs to capture H features of the context that are sufficient to predict nearby words with a 1-layer NN without activation (i.e. a logistic regression model).

(14) Observe that $\pi = \begin{bmatrix} w_{4i} \\ w_{42} \\ \vdots \\ w_{4H} \end{bmatrix}$, which is just the 4th raw of matrix W. So effectively the 4th raw is compressing word 4 of the vocabulary ("zebra") into a fixed size vector that is optimized to be able to predict nearby words.

And in general, the h^{th} raw of W compresses the h^{th} word of the vocabulary into a set of predictive features.

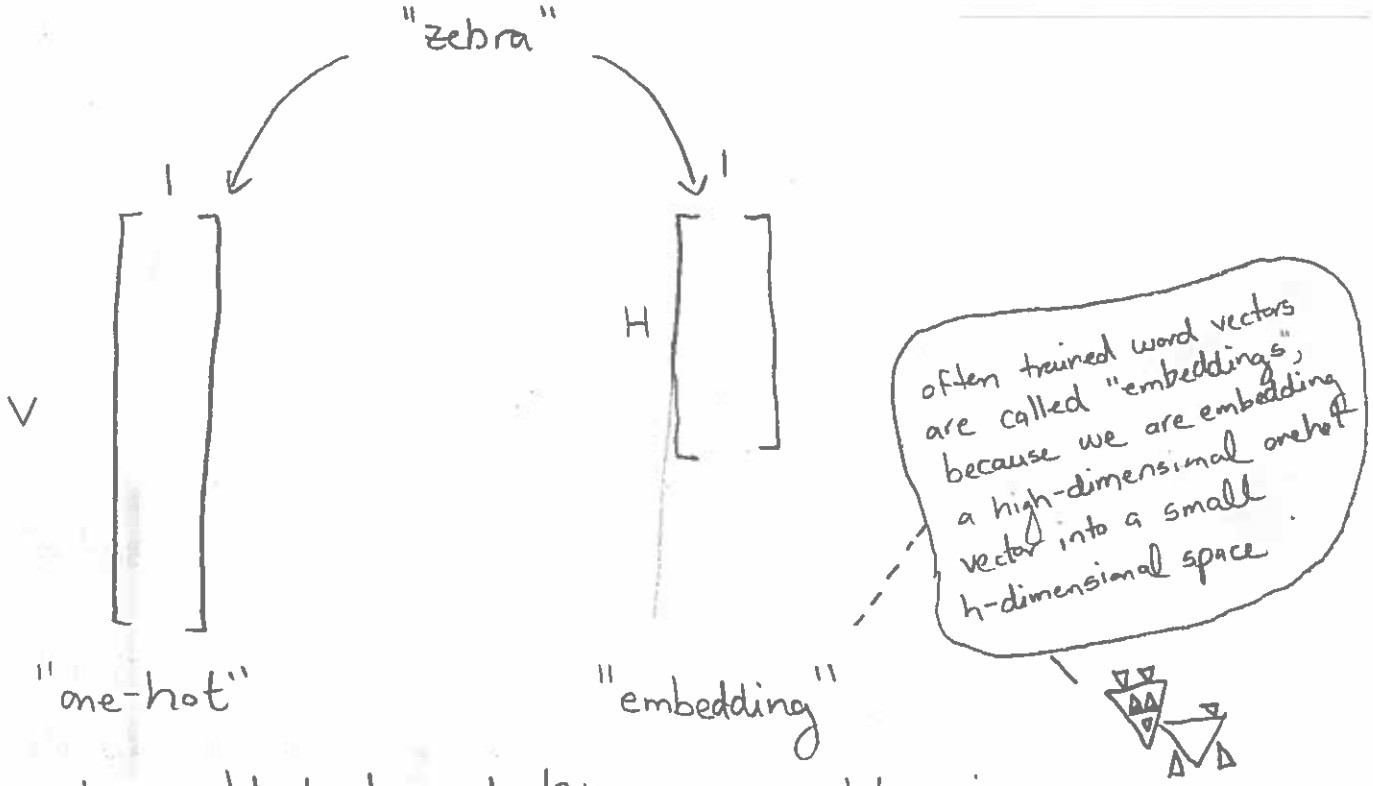
---

(15) This can give us some very useful representations of words. Notice that dogs and cats are both often adopted, petted, fed, loved, and housebroken. This gives the words "dog" and "cat" a distinct context from, say, "zebras," whom are rarely loved due to their anarchistic tendencies.

Therefore, the trained vectors may encode some idea of "being a pet," in order to more accurately predict surrounding context.

Word Vectors

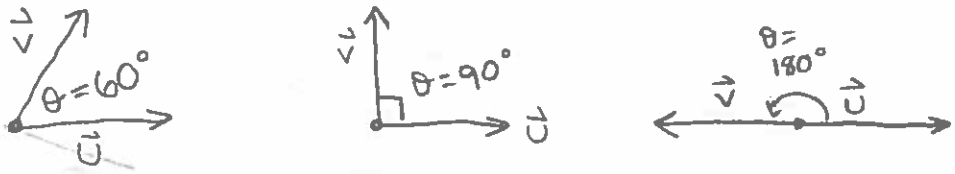
(16) A practical upshot is that we can then use these word vector representations in more complex models instead of a one-hot representation.



We can choose H to be whatever manageable size we like (H = 300 is a common choice), whereas the vocab size V is out of our control.

Word Vectors

(17) What do the trained word vectors end up looking like? Well, there are challenges to visualizing them, since they are in something like a 300-dimensional space, but even in a high-dimensional space, two vectors are always separated by some angle θ.



If $\theta=0°$, then they're pointing in the same direction (and therefore very similar)

If $\theta=90°$, then they're orthogonal (and presumably also appear in unrelated contexts)

If $\theta=180°$, then they're pointing in opposing directions (and therefore may have opposing contexts)

---

(18) Note that $\theta=180°$ doesn't suggest opposite meanings, but rather opposite contexts. "fast" and "slow" are opposites, but often appear in similar sentential contexts.

(19) Given two vectors $\vec{u}$ and $\vec{v}$, how do we compute the angle between them? Well, the geometric formulation of dot product tells us:

$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \, \|\vec{v}\| \cos\theta$$

where $\|\vec{x}\| = \sqrt{x_1^2 + \ldots + x_n^2}$ is the length (L_2-norm) of vector $\vec{x}$, and θ is the angle that separates $\vec{u}$ and $\vec{v}$. Thus:

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}$$

---

(20) This measure, $\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}$ is called the <u>cosine similarity</u> of vectors $\vec{u}$ and $\vec{v}$. Observe:

when $\cos\theta = 1$, then $\theta = 0°$

$\cos\theta = -1$, then $\theta = 180°$

$\cos\theta = 0$, then $\theta = 90°$

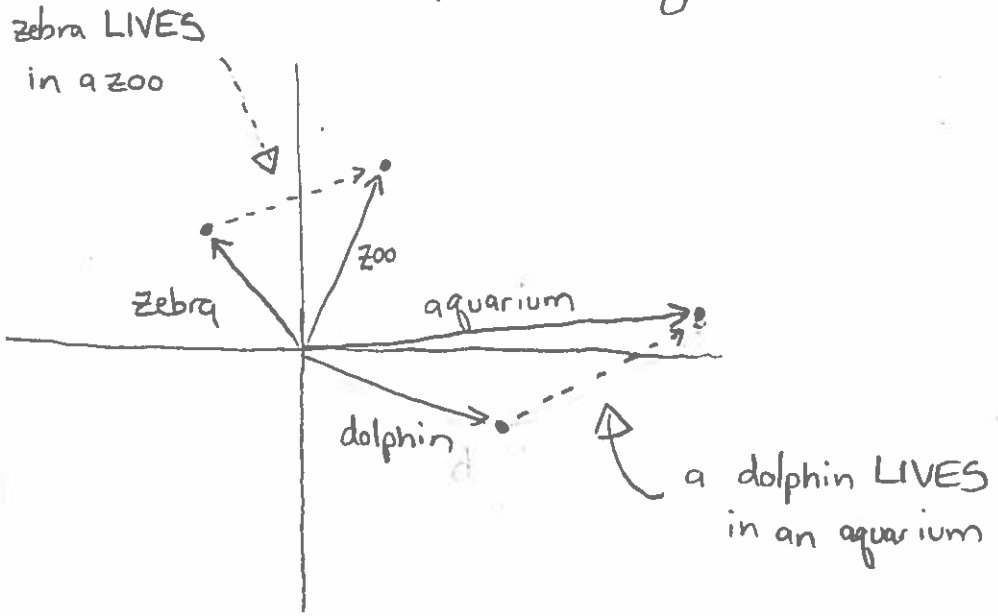
So a cosine similarity of 1 indicates maximal similarity, while a cosine similarity of -1 indicates maximal dissimilarity.

Word Vectors

(21) A commonly used set of word vectors was released by Google and trained on 100 billion words of Google News documents. These vectors have dimension 300. If we compute the cosine similarity of the word "king" with various other words, we get:

queen : 0.651
chess : 0.179
zebra : 0.098
farmed : 0.063

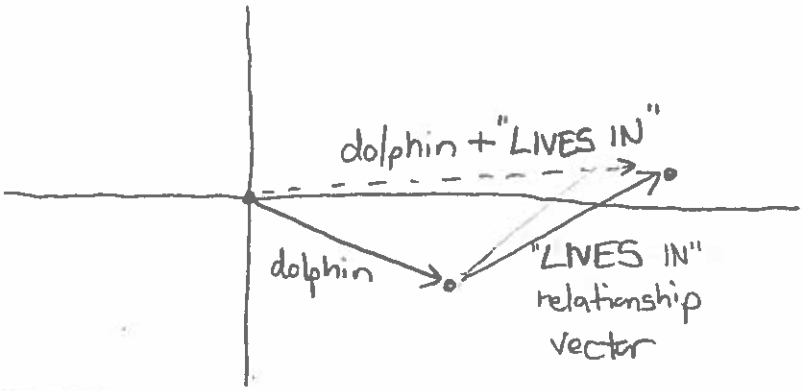
(22) But there are even deeper, more interesting structures that one can uncovered with cosine similarity. One could hypothesize that perhaps word relationships are also captured in the vector space, e.g.



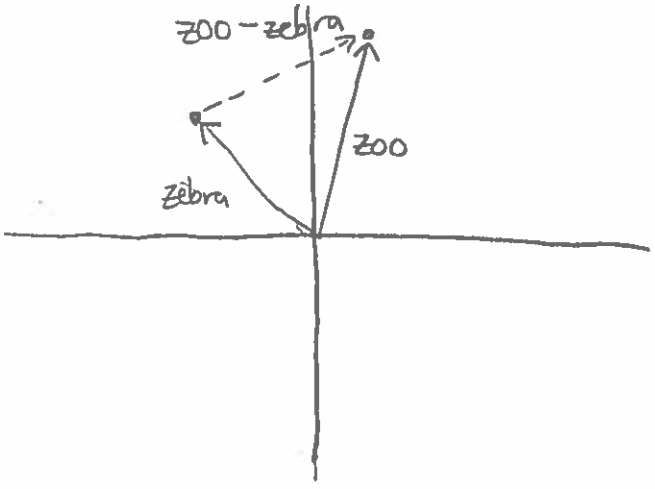
(23) IF that were indeed the case, then we should be able to solve analogies, e.g.

zebra is to zoo AS dolphin is to ____

Namely, we look for the closet vector $\vec{v}$ such that



(24) We can compute the relationship vector between zebra and zoo geometrically:



Thus, we look for the closest vector to:

$$\text{dolphin} + (\text{zoo} - \text{zebra})$$

(25) Shockingly, this often works. Here are the closest five vectors using the Google News vectors:

zebra is to zoo AS dolphin is to
1. aquarium
2. Aquarium
3. dolphins
4. Vancouver Aquarium
5. Oceanarium