Stanford - CS224n - Lecture 1 - Introduction and word vectors

*) language and social interactions make humans different.

mankind is like a human network and language is an important means of communication. (knowledge → intelligence)

↓

The way we acquire knowledge is from our ancestors, through books (their observations), through language.

↓

Language can be seen as one of the most important differentiator between hemans and other animals.

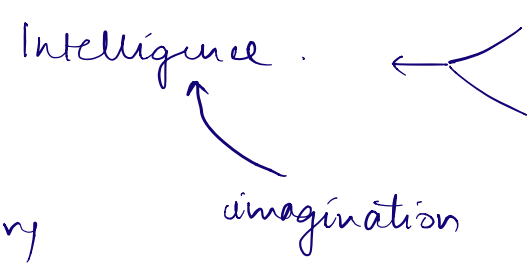
Even animals with a good sense of control and vision.

↓

Again, language and social interaction can be seen as the key difference, the way of passing knowledge, the way of acquiring intelligence

---

Sides



Information Theory

Imagination and General AI.

---

*) A gamechanger was writing, that allowed knowledge to be transferred temporally and spatially. (5000 yrs old)

very recent on scale of evolution. Enable major human progress.

*) The human communication network is pathetically slow as compared to what computer networks have been able to achieve

*) But we have a sort of adaptive compression mechanism, wherein, we need a very few bit sequences to convey information. for example, by passing a basic explanation about a scene, we are able to imagine the picture in our minds. This also works as we assume the person on the other hand to have a similar amount of acumen to understand what we're trying to convey.

*) meaning → idea → denotational semantics (linguists)

↓

also employed in programming languages.

what is something that the word or words actually denote

↓

ex. chair → four legged wooden things (more like picture in a book while learning words, and their associated meanings)

running → set of actions taken to execute that activity.

↓

hard to use in a computational space.

*) one way of getting a sense of meaning has been through the use of Wordnet, but it didn't give good results for a set of reasons. It contains set of synonyms for words to get a sense of meanings along with pos tags based on different contexts. Hard to maintain, and also doesn't give a good sense of similarity b/w words in two different

similarity sets.

*) Traditional NLP (pre 2012 period, before Neural Networks were being used) used one-hot-encoded (localised approach) vectors, where 1 was assigned to the word corresponding to its index in a vocabullary. Issues:

:) very large vocabularies → very large vectors

:) understanding relationships b/w words.

for example if we search for "seattle motel", we would like to match documents containing "seattle hotel", since "motel" and "hotel" represent basically the same thing.

↓

One have no similarity notion.

↓

People tried to rely on wordnet building similarity tables, and owing to the large size of vocabulary would be even bigger.

*) Distributional semantics → If we can say on which context, the word can be used, and in which context it can't be, then we can say, we have an understanding of the meaning of the word.

↓

context represents meaning (word vectors)

*) Owing to the above idea, we build a dense vector for each word, chosen so that similar words appear in

similar contexts.

word vectors ↔ word embeddings ↔ word repr.

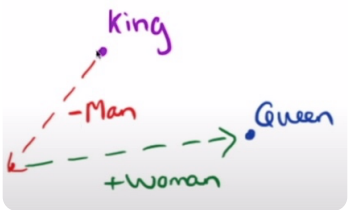
↓

(distributed representation)

*) Numeric vector, smallish, dense. meaning distributed along the dimension of the vector. min: 50,

on personal mle: 300

max out performance: 1000s.



Dimensions of meaning in this vector space.

↓

many magnitudes smaller than one vectors.

*) Now, since words are being represented using vectors, they can be plotted in the particular vectorspace. The closer words will correspond to being more similar, as compared to others. This can also be visualized by projecting words in a 2D space.

Note: NLP people call large pile of text a corpus.

*) word2vec: An algorithm to learn the dense vector representations of words. Basically done by going over a large corpus of text. for each position t in the text, we can consider a center word 'c', and context words represented as 'o'.

↓

we use the similarity between the word vectors of 'o'

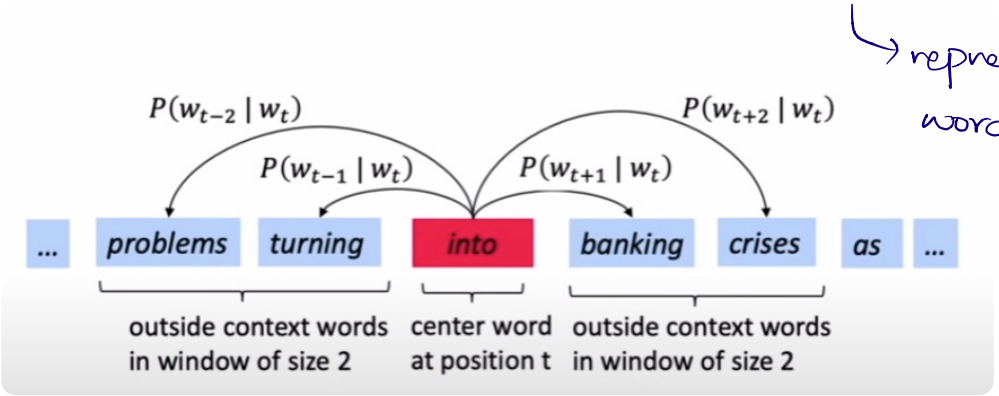
and 'c' to calculate the probability of o given c.

↓

we keep adjusting the word vectors to maximise this probability.

context of into → meaning of into

↳ represented by the word vector.



*) objective fn :

for each position $t = 1 \ldots T$, predict context words within a window of fixed size, m, given center word, w_j.

→ going through all the words one by one in the corpus

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} / w_j)$$

$L(\theta)$ → Likelihood function

θ → represents parameters to be optimized.

$P(w_{t+j} / w_j)$ → conditional probability of context word wrt center word → dependent on the parameters. → vector representation of words.

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} / w_j)$$

$J(\theta)$ → objective fn

minimizing objective fn ⟷ maximizing predictive accuracy.

log → summation instead of product

(solves the problem of very small values, due to multiplication of values b/w of 1)

*) How to calculate $P(w_{t+j} / w_t ; \theta)$?

Two vectors per word; one representation as center word, other as context word.

v → center

u → context

Then for center word c and a context word o

$$P(o/c) = \frac{\exp(u_o^T \cdot v_c)}{\sum_{w \in V} \exp(u_w^T \cdot v_c)}$$

→ measure of similarity

↓

large dot product, larger probability

↓

normalize over entire vocabulary.

↓

Softmax Distribution

↓

maps numbers to a probability distribution

*) Softmax because it assigns max probability for largest individual value, but does so in a manner that even the smallest values have some probability assigned to them.

*) Optimization of parameters to minimize the objective vector. θ represents all model parameters, in one long vector. for d dimensional word representations,

V size of vocabulary and 2 vectors for each word, the long vector is of the dimension $(2V * d)$.

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

$$= \frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T \cdot v_c)}{\sum \exp(u_w^T v_c)}$$

$$= \frac{\partial}{\partial v_c} \log \exp(u_o^T \cdot v_c) - \frac{\partial}{\partial v_c} \log \sum \exp(u_w^T \cdot v_c)$$

(property of the log)

$$\frac{\partial}{\partial v_{c_1}} u_{o_1} v_{c_1} + u_{o_2} v_{c_2} + \cdots + u_{on} v_{cn} = u_{o_1}$$

↓

Similarly for complete v_c

$$\frac{\partial}{\partial v_c} (u_o^T \cdot v_c) \longrightarrow \text{numerator}$$

↓

multivariate calculus

$$\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T \cdot v_c) = \sum_{x=1}^{V} \frac{\exp(u_x^T \cdot v_c) \cdot u_x}{\sum_{w=1}^{V} \exp(u_w^T \cdot v_c)}$$

↓ denominator

↓

$$\frac{\partial}{\partial v_c} \log P(o/c) = u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T \cdot v_c) \cdot u_x}{\sum_{w=1}^{V} \exp(u_w^T \cdot v_c)}$$

$$= u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T \cdot v_c)}{\sum_{w=1}^{V} \exp(u_w^T \cdot v_c)} \cdot u_x$$

$$= u_o - \sum_{x=1}^{V} p(x/c) \cdot u_x$$

↓ context word

expected context → convex sum

↓ difference b/w the actual context word and expected context word