Natural language is unreasonably simple, unreasonably often

Adding up word embeddings works far too well, why is that?

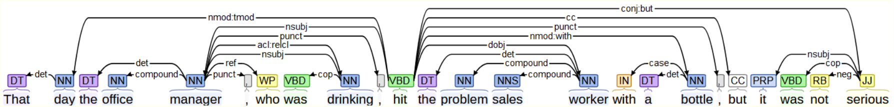
Lyndon White

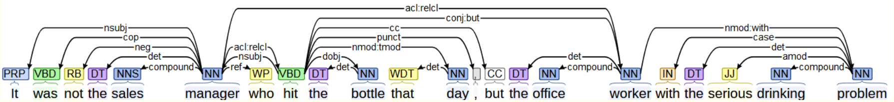
School of Electical, Electronic and Computer Engineering
The University of Western Australia

Australia, really it is quiet far away

The University of Western Australia

We like to think language is very complicated





This is what we do, and complicated models make us feel good and publish well.

and sometimes language is

This movie is a truly excellent example of the quality of cinematography this century; bring back the good old days of real cinema!

You shouldn’t miss this,
that would be the worst mistake.

It’s not that it is was bad,
but it wasn’t what I hoped for.

And so we need the complicated models.

but sometimes language isn’t

- `The girl stands on the tennis court.`
- Not: `The tennis court stands on the girl.`

but sometimes language isn't

- `The girl stands on the tennis court.`
- Not: `The tennis court stands on the girl.`

How do we know?

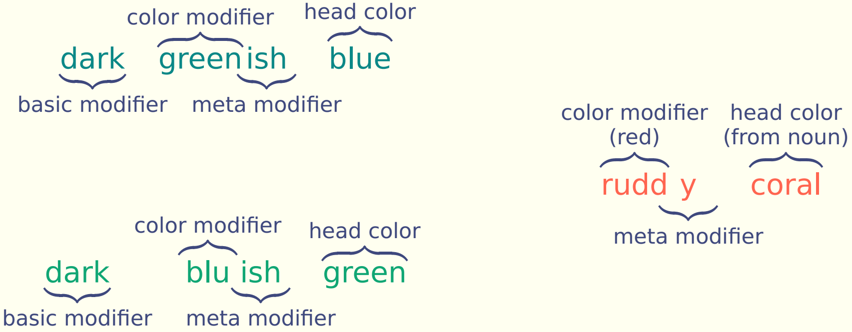
World Knowledge: `girl` is an agent, that can take actions

OR

Language Modelling: the trigram `tennis court stands` never occurs in the Google Books corpus.

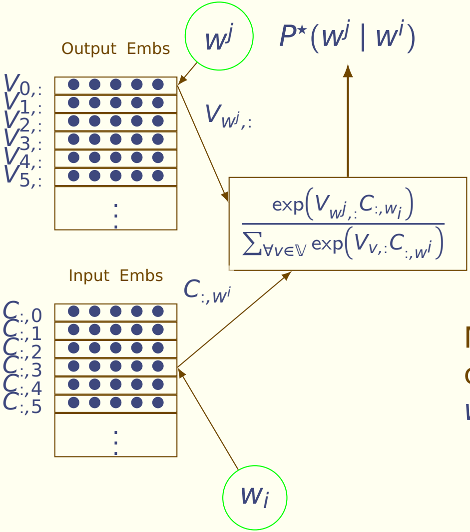
And so simple methods work

and often it looks like it is complicated
but isn't



and so using complicated methods leads to
worse performance.

SkipGram is the most well known of recent word embedding methods

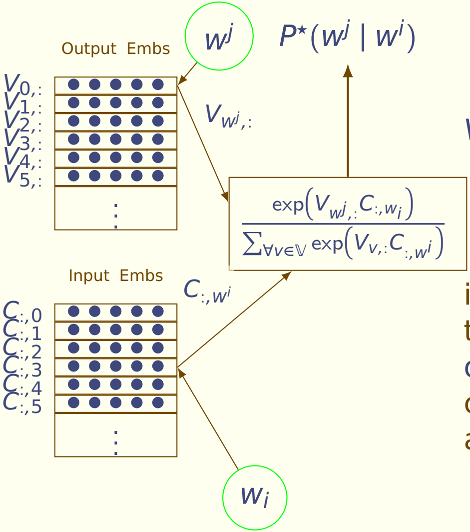


$$P^\star(w^j \mid w^i) = \operatorname{smax}\left(V\, C_{:,w^i}\right)_{w^j}$$

$$= \frac{\exp\left(V_{w^j,:} C_{:,w^i}\right)}{\sum_{\forall k \in \mathbb{V}} \exp\left(V_{k,:} C_{:,w^i}\right)}$$

Maximise $P^\star(w^j \mid w^i)$ for combinations of words w^j, w^i that actually co-occur.

SkipGram is the most well known of recent word embedding methods



$P^\star(w^j \mid w^i)$ is maximised when $V_{w^j,:} C_{:,w^i}$ is maximized.

i.e. when the dot-product of the input embeddings and the output embeddings of collocated words approaches 1

SkipGram is a iterative algorithm for weighted collocation matrix factorization.

V is a $300 \times |\mathbb{V}|$ input embeddings matrix
C is a $|\mathbb{V}| \times 300$ output embeddings matrix
X is a $|\mathbb{V}| \times |\mathbb{V}|$ collocation count matrix
f is some monotonic weighting function.
(Levy, Goldberg, and Dagan 2015)

$$
\begin{aligned}
Loss &\propto -X \odot \exp(VC) \\
&\propto VC - \log X \\
&\propto VC - f(X)
\end{aligned}
$$

Loss is minimized when $VC \approx f(X)$

When trying to factorize very large matrices numerical linear algebraticians often use iterative methods.

So SkipGrams are a dimensionality reduction algorithm, that tries remember collocated words

- Contrast: PCA is a dimensionality reduction algorithm, that tries to remember the most variant factors
- Contrast: t-SNE is a dimensionality reduction algorithm, that tries to preserve similarity as distance
- Compressing knowledge of collocated words into a dense vector, gives us Firth's Criterion.

Matrix product with onehot vector is indexed slicing

Consider the onehot representation of some word w, as $\tilde{e}_w = \left[0, \ldots, \underbrace{1}_{w\text{th position}}, 0 \ldots, 0\right]$

It's word embedding is given by $C_{:,w} = C^\intercal e_w$

Sum of word embeddings is the same as matrix product with bag of words

A bag of words can be represented as a vector of the counts of each word in the vocabulary.

For a sequence of words: $\left(w^1, w^2, \ldots\right)$

The bag of words can be given by
$\tilde{x} = \sum_{\forall i} \tilde{e}_{w^i}$.

The sum of word embeddings for the same sequence is:
$\sum_{\forall i} C_{:,w^i} = C^\intercal \sum_{\forall i} \tilde{e}_{w^i}$

Concatenation followed by matrix product
is the same as
matrix product followed by addition

$$\begin{bmatrix} U & V \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} = U\tilde{a} + V\tilde{b}$$

Thus

$$\begin{aligned} C(\tilde{a} + \tilde{b}) &= C\tilde{a} + C\tilde{b} \\ &= \begin{bmatrix} C & C \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} \end{aligned}$$

A summed input is the same as a
concatenated input
with blockwise weight tying.

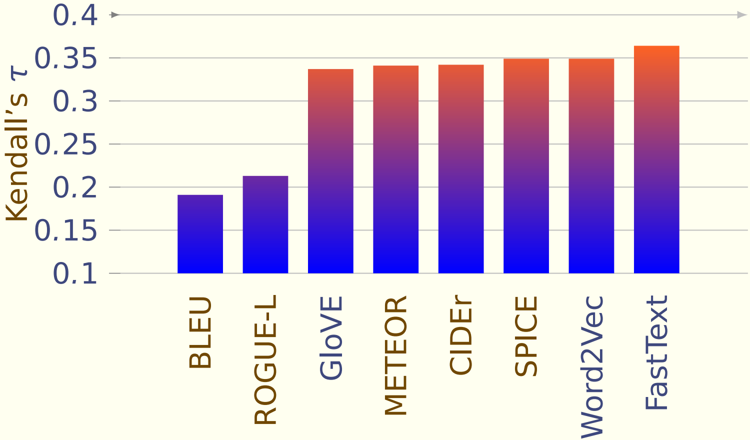
Bag of words information is not lost in sums of word embeddings

- A bag of words captures all unigram information in a sentence/document etc

We can reliably recover all words from a sum of word embeddings

Consider machine captioning evaluation

Correlation with human ranking in the COMPOSITE captioning evaluation dataset. Aditya et al. 2017



$*$Forthcoming publication Naeha Sharif, Lyndon White, Mohammed Bennamoun and Syed Afaq Ali Shah.

What is going on? How can a unigram method be beating everything?

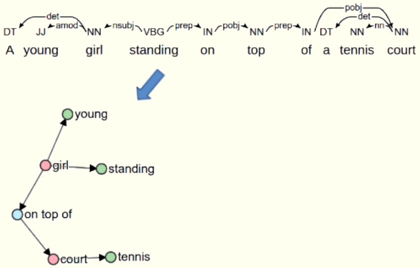
Captioning quality can be assessed on **fluency** and on **adequacy**

All captions in COMPOSITE are **fluent**

- We are really good at language modelling now.
- In theory our RNN language models can capture all needed state
- COMPOSITE captions are a mix of human generated and state-of-the-art machine generated.

Trying to capture fluency in your captioning metric is thus not important

The proper way to look at adequacy is to build a semantic graph, and apply reasoning to it



This is what SPICE does . Anderson et al. 2016
You could use AMR Banarescu et al. 2013, or ERSBender et al. 2015.
To get to a form that reasoning can be applied on.

This semantic graph must be derived from the **right words** in the right **right order**

Semantic graph comes from syntactic graph

- The syntactic graph comes from the word order and word content.
- In theory, different words in different orders could give the same semantic graph
- and the same words in a different order could give a different semantic graph.

Due to ambiguity in possible word order semantic meaning should not be derivable from averaged lexical meaning representation

- Well written sentences are short: 14-17 words
- They don't have complicated clauses and negations.
- Words are used in consistent phrases:
  - `The girl stands on the tennis court`
  - Not: `The tennis court stands on the girl`
- Good captions are such good sentences.

But in-practice, it probably is

Some might say the problems we are assessing on are not sufficiently difficult

The problems are exactly as difficult as they are.

- real world problems on real data.

Maybe we are not really doing natural language understanding

but practically we are certainly doing something useful