Natural language is unreasonably simple, unreasonably often

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

Lyndon White

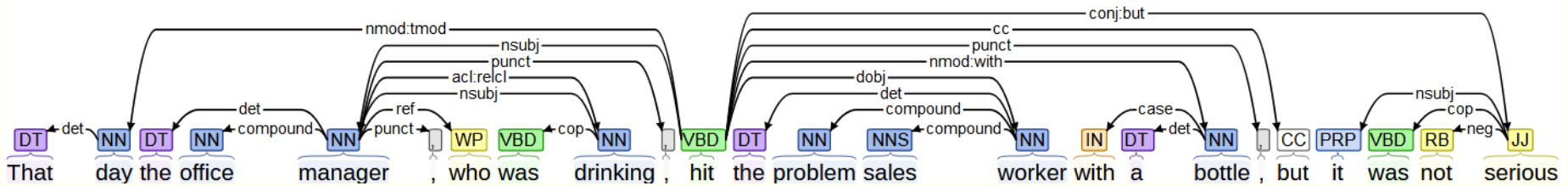
Department of Electical, Electronic and Computer Engineering
The University of Western Australia

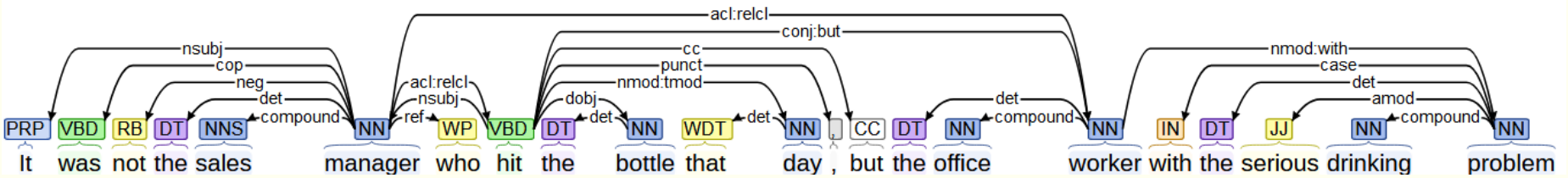
THE UNIVERSITY OF WESTERN AUSTRALIA

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; for goodness sake, 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?

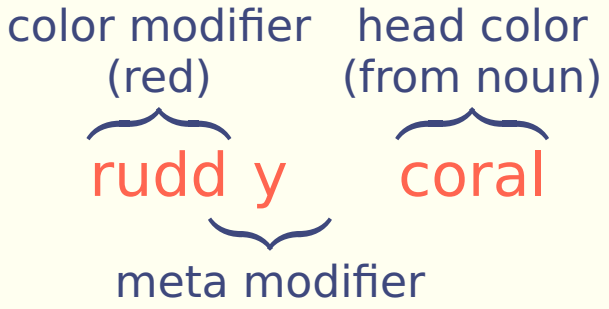
World Knowledge: `girl` is an agent, that can take actions

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.

You can just add up word embeddings and uses it as a representation

acl2018bleuopposedmeaning (acl2018bleuopposedmeaning), acl2018bleuopposedmeaning

"On classification tasks, our [LSTM encoder-decoder] models are outperformed even by GloVe-BOW, except for the NLI tasks. . . "

ac2018probingsentencevectors (ac2018probingsentencevectors), ac2018probingsentencevectors

"Our first striking result is the good overall performance of Bag-of-Vectors, confirming early insights that aggregated word embeddings capture surprising amounts of sentence information. . . "

Content is king

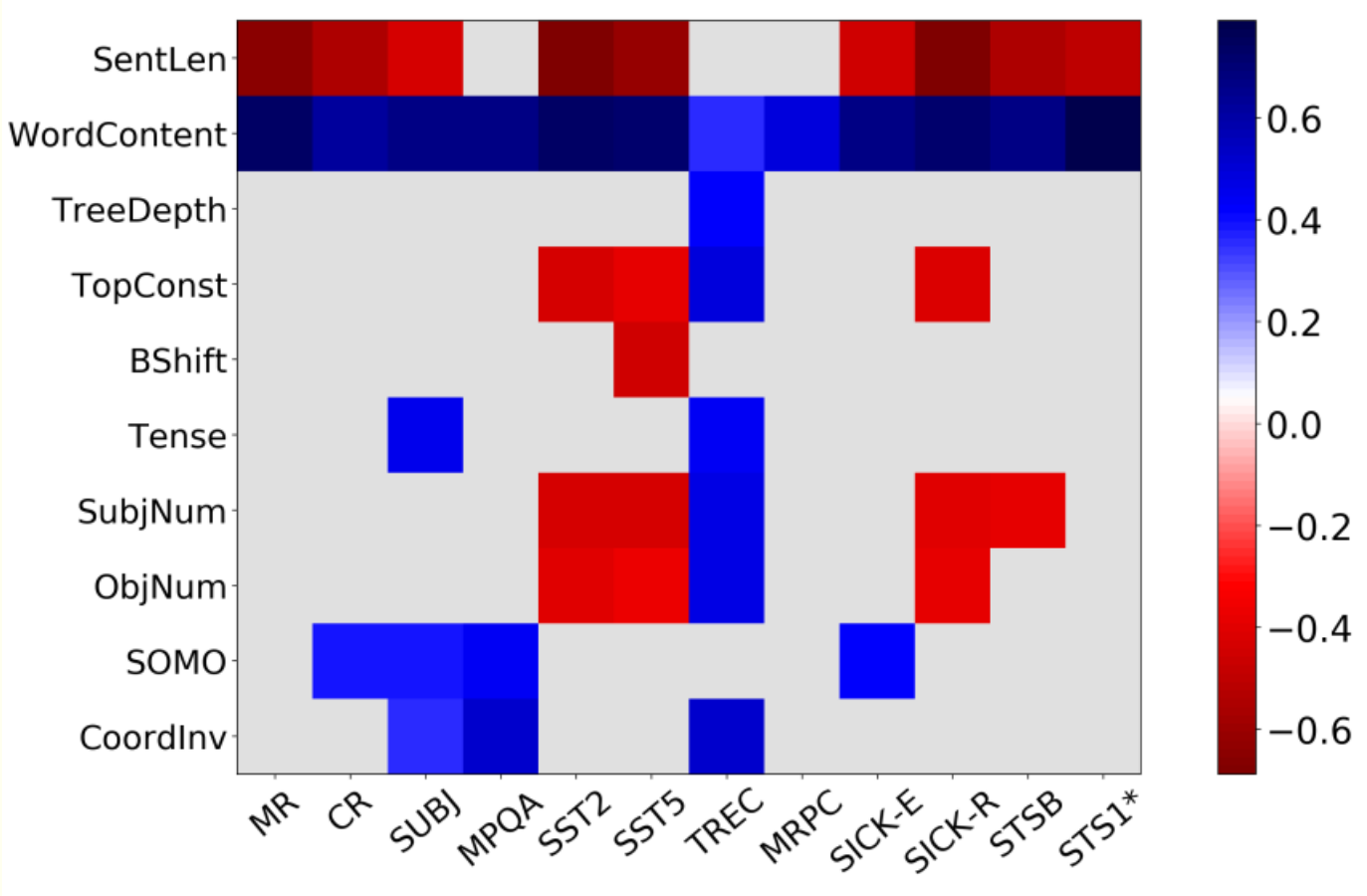
Considering further **ac2018probingsentencevectors**

- They looked at $3 \times 8 + 5$ models
- 10 probing tasks were used to evaluate the models.
- Of the baseline models SOWE was often the best.
- It was rarely better than the best performing sophisticated model
- Except on Word Content

The words people use are the most important part of what they are saying

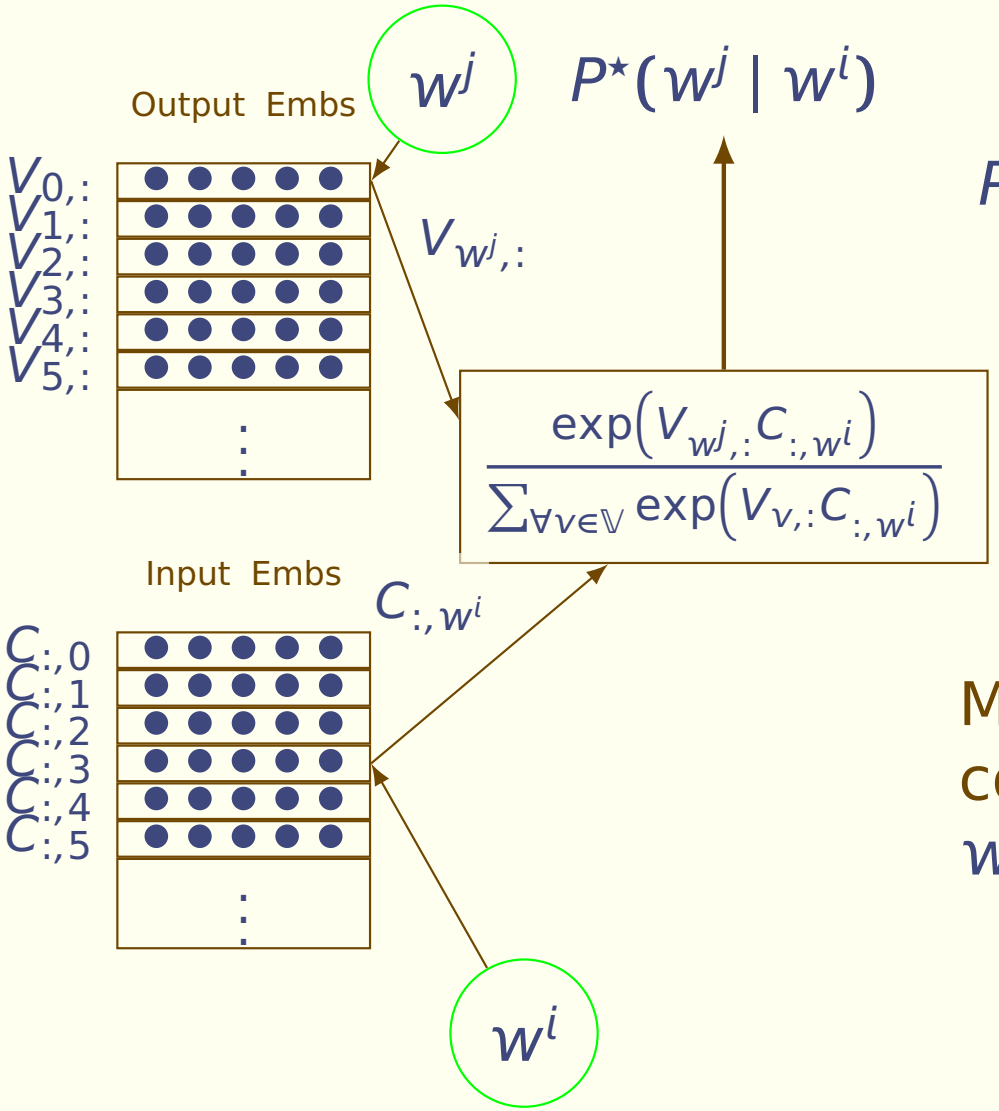
Content is king

Considering further **ac2018probingsentencevectors**
When the probing task results are correlated against 12 down stream tasks.



The words people use are the most important part of what they are saying

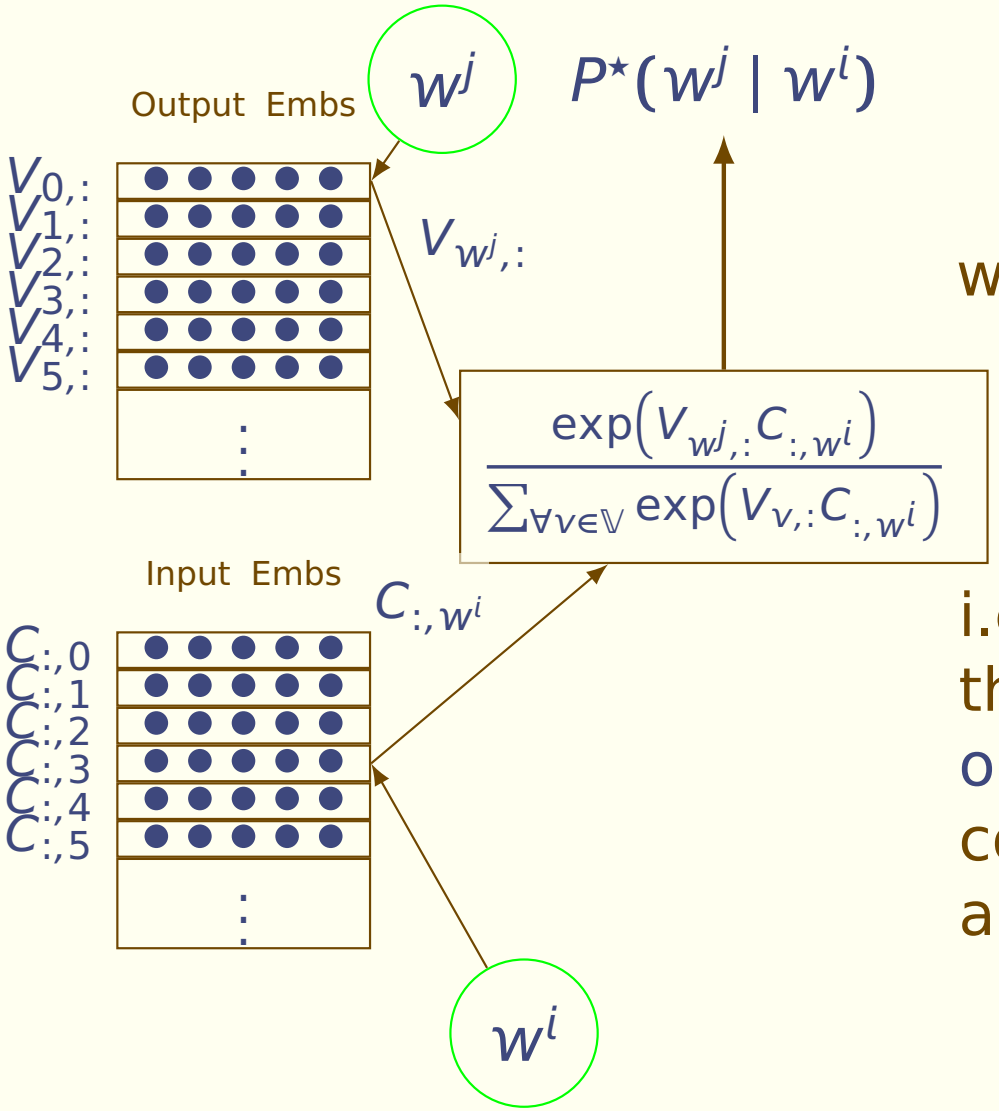
SkipGram is the most well known of recent word embedding methods



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

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.
(**levy2015lsaisbasicallyskipgramswithexperimentstoprove**

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

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

When trying to factorize very large matrices numerical linear algebraicians 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: $(w^1, w^2, \ldots)$

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

$$C(\tilde{a} + \tilde{b}) = C\tilde{a} + C\tilde{b}$$

$$= \begin{bmatrix} C & C \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix}$$

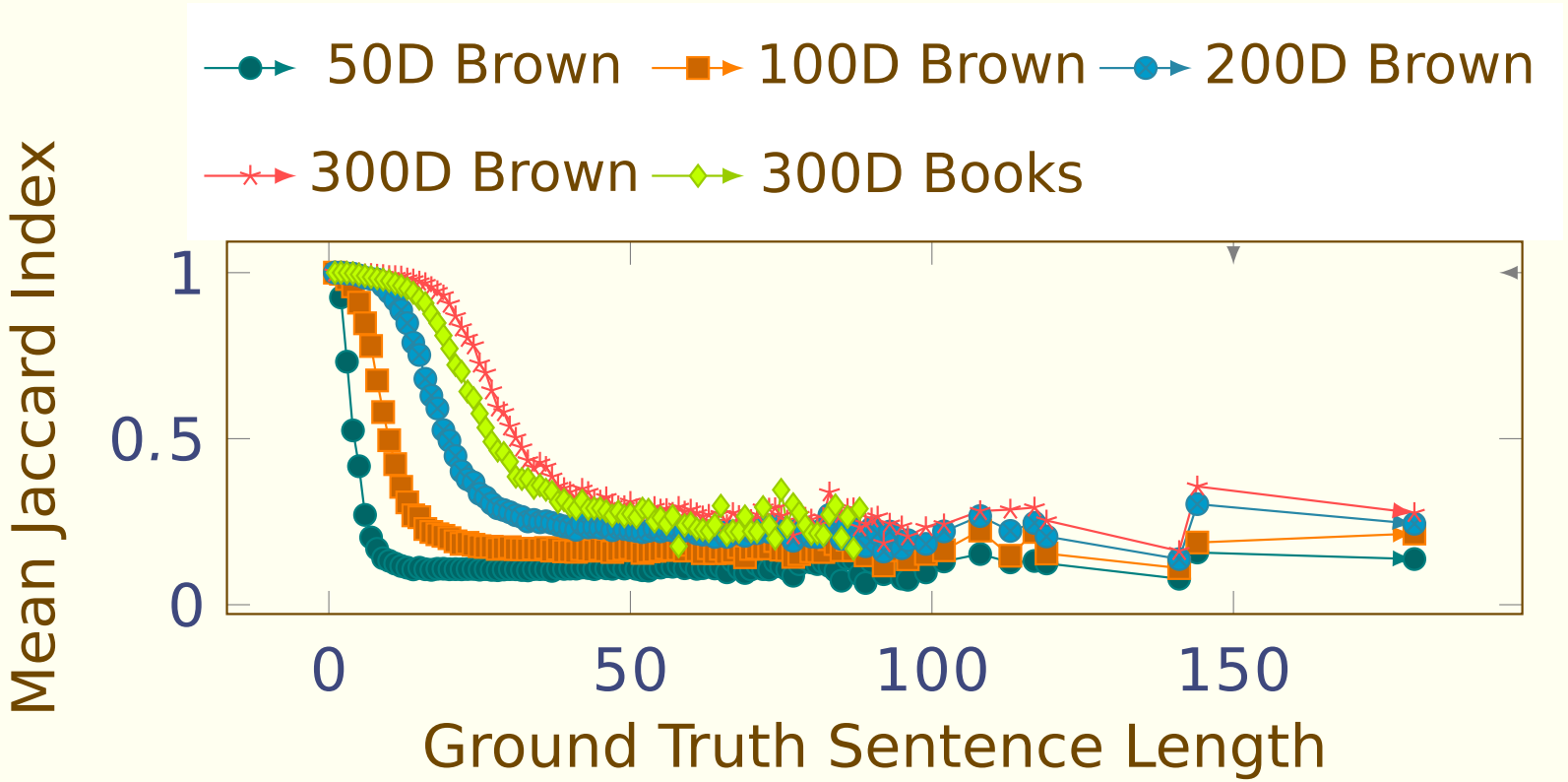
A summed input is the same as a
concatenated input
with blockwise weight tying.

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
- A greedy method can (mostly) recover the BOW from a SOWE
  White2015BOWgen (**White2015BOWgen**), **White2015BOWgen**
  1. Greedily add nearest word to remaining the SOWE to bag
  2. Check each word in bag and swap it for a better one
  3. Repeat until no change.
- Thus, SOWE captures similar most unigram information.

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

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



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

Sentence embeddings space should partition readily according to paraphrases

- Paraphrases are defined by bidirectional entailment.
- This is an equivalence relation
- It thus gives rise to a partition of natural language space.

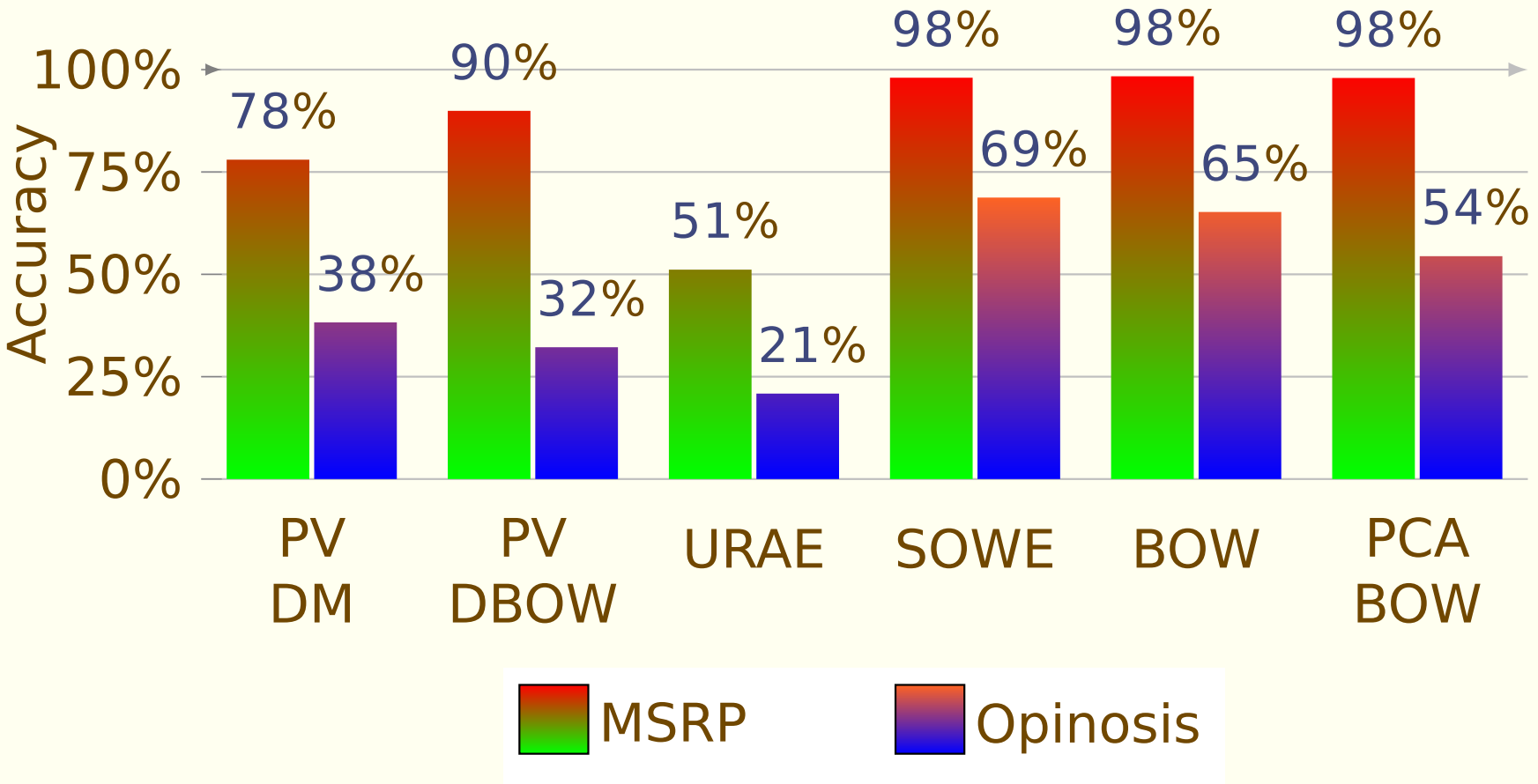
What does it mean to partition readily?

There are many ways one could define the quality of a space, on its ability to be partitioned according to anther linked space's partitions.

Convex	No twists, bulges, holes, jumps etc.
Seperable	Should not overlap, should be separate
Concentrated	small area

Notice: these are the same criteria needed linear SVM to work well.

When assessing ability to match partitions using a linear SVM classification task



Knowing word content is really useful

What is going on here?

PV-DM / PV-DBOW
le2014distributed (**le2014distributed**),
le2014distributed

"It shows that the paragraph vectors, when evaluated correctly, do not work better than bag-of-words (bag-of-ngrams being even better)"
– Tomas Mikolov (23/11/17) w.r.t **mesnil2014ensemble**

URAE
SocherEtAl2011:PoolRAE
(**SocherEtAl2011:PoolRAE**),
SocherEtAl2011:PoolRAE

Only 200D so not entirely fair.
Based on binary constituency parse structure
Deep, compressing each layer, 2 inputs to one output.

What is going on here?

BOW
Bag of Words

Several thousand dimensions
1000D spaces tend to be very separable
No synonym capacity

PCA BOW
Principle Component Analysis **hotelling1933analysis**

Drop down to 300D to match others
PCA is inferior to GloVE as a dimensionality reduction technique, for semantic preservation.

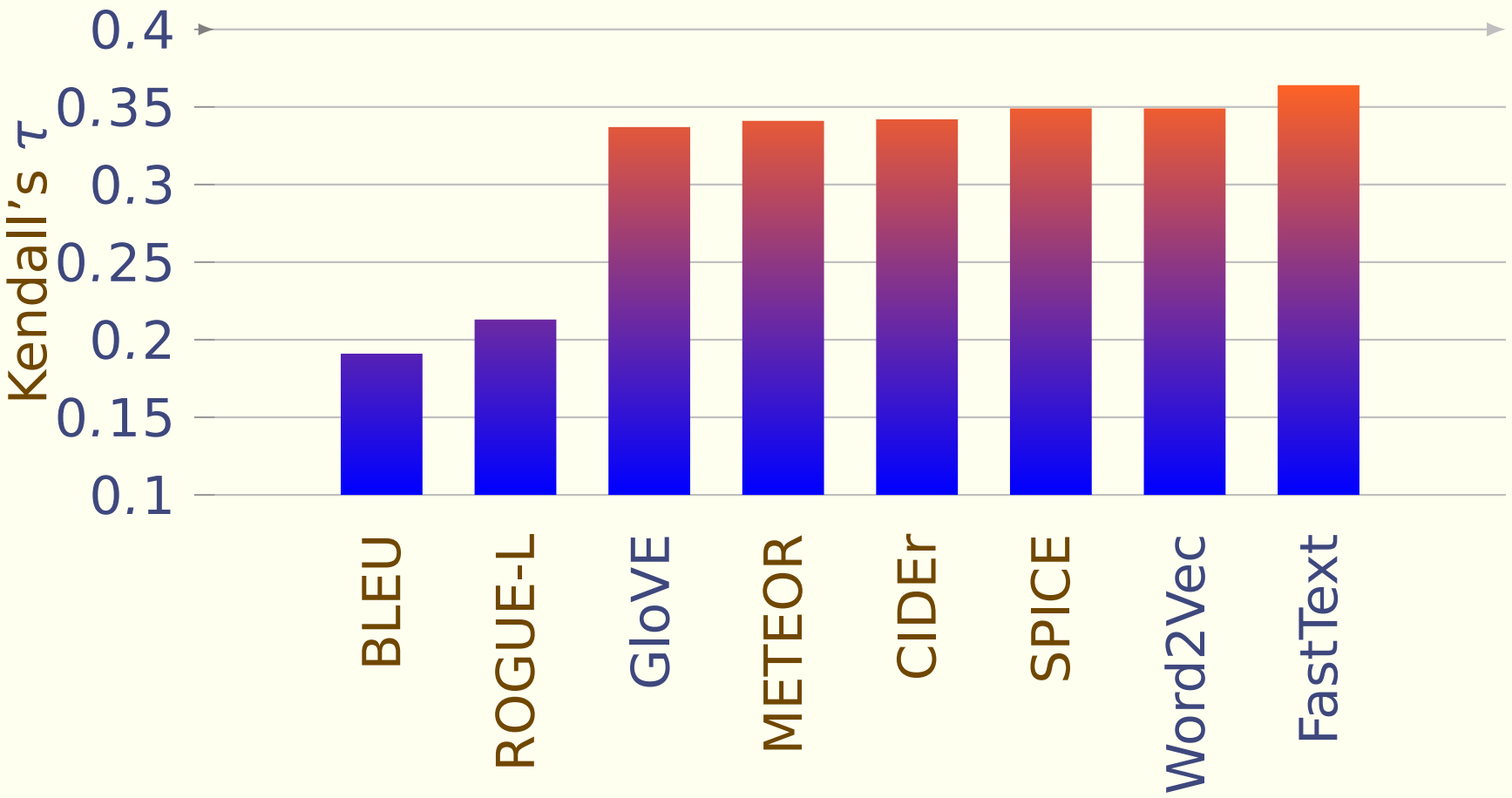
Consider machine captioning evaluation

Correlation with human ranking in the COMPOSITE captioning evaluation dataset. **Aditya2017**

- For each image, there are 5 captions.
- A mix of human generated
- and machine generated.
- this is an evaluation of evaluation metrics

Task is to rate the captions in the same order as the human rankers.

Look what wins



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

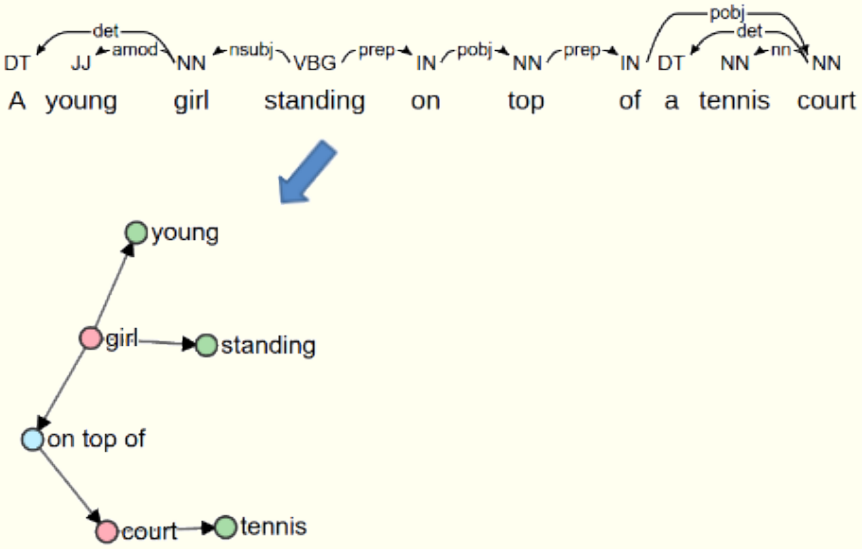
Captioning quality can be assessed on **fluency** and on **adequacy**

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 .
spice2016
You could use AMR
Banarescu13abstractmeanin
or ERS**bender2015ERS**.
To get to a form that reasoning can be applied on.

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

syntactic graph comes from token order

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

Word order is more predictable than you may think

- Consider some sentence; can you rearrange the words and punctuation and make new sentence
- Is it grammatical?
- Is it a paraphrase?
- Is it likely (or sensible) based on world knowledge?
- Was it following some standard transform, like just swapping to proper nouns?

Word order is more predictable than you may think

- Consider some sentence; can you rearrange the words and punctuation and make new sentence
- Is it grammatical?
- Is it a paraphrase?
- Is it likely (or sensible) based on world knowledge?
- Was it following some standard transform, like just swapping to proper nouns?

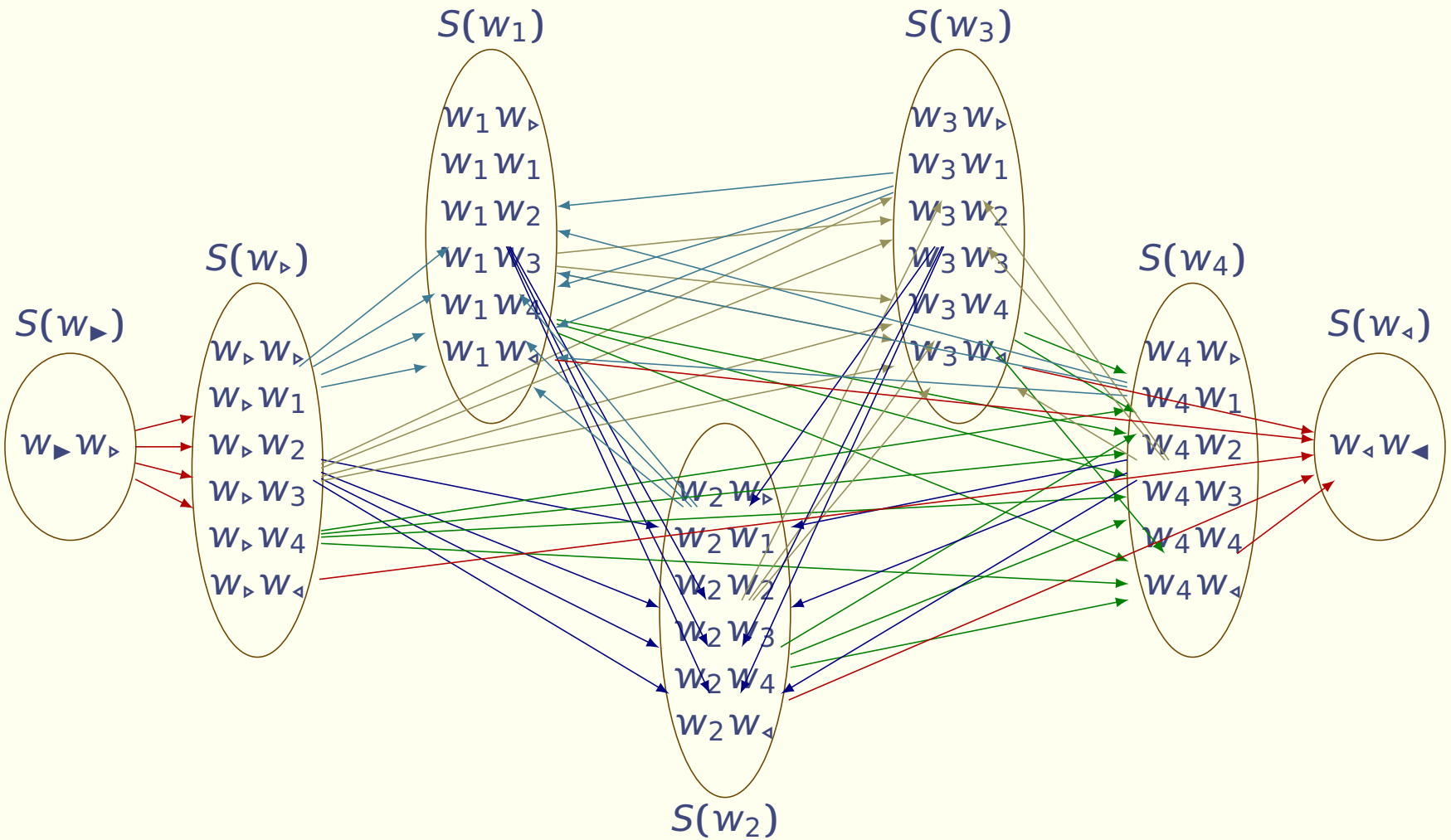
You can recover most probable sentence order from bag of words

White2016a (**White2016a**), **White2016a**

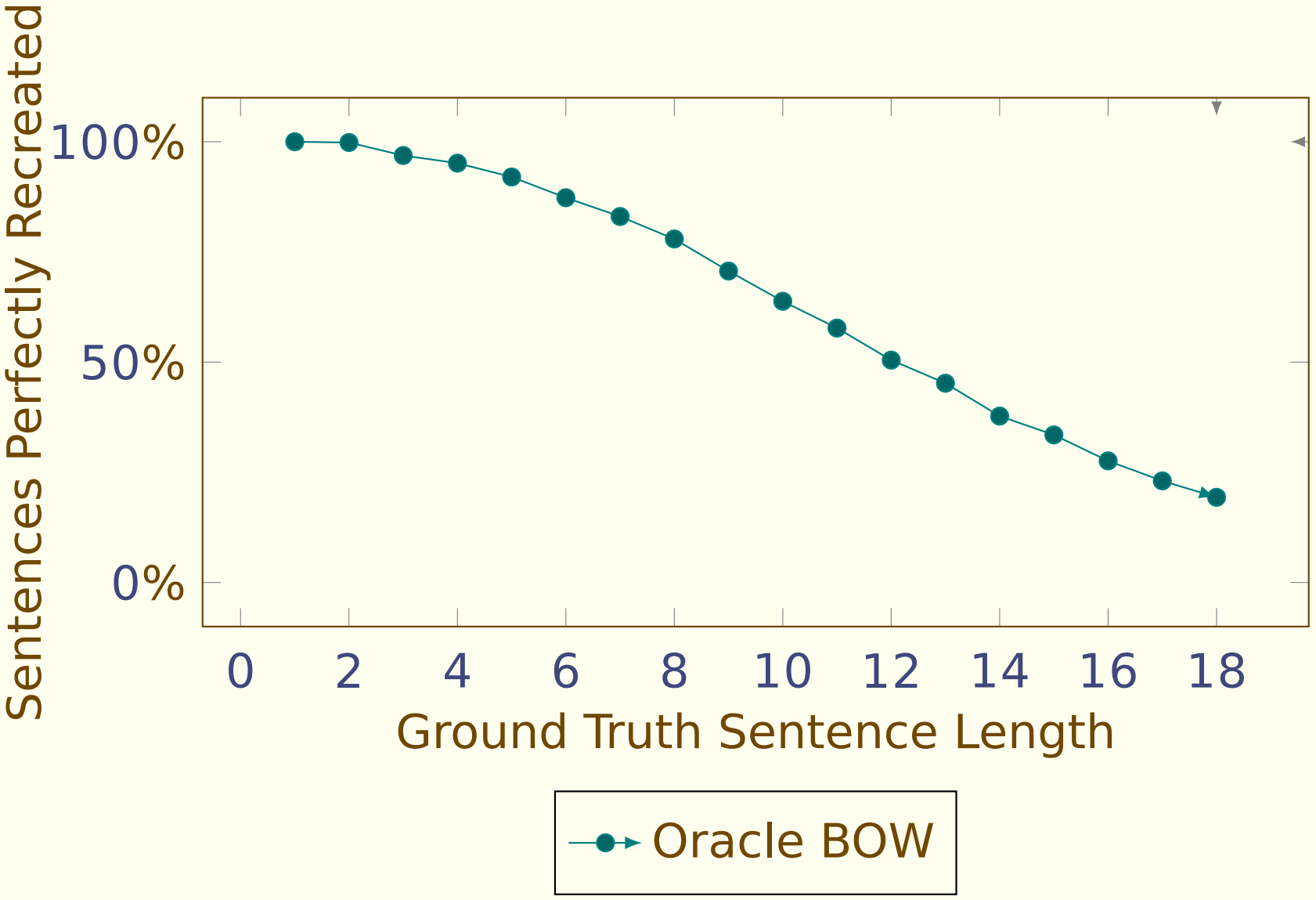
- Using a language model you can evaluate the probability of any given order for a bag of words.
- With an appropriate optimization/search technique you can find the most likely order
- This particular work was a simple trigram language model

Not a great way to generate sentences, though

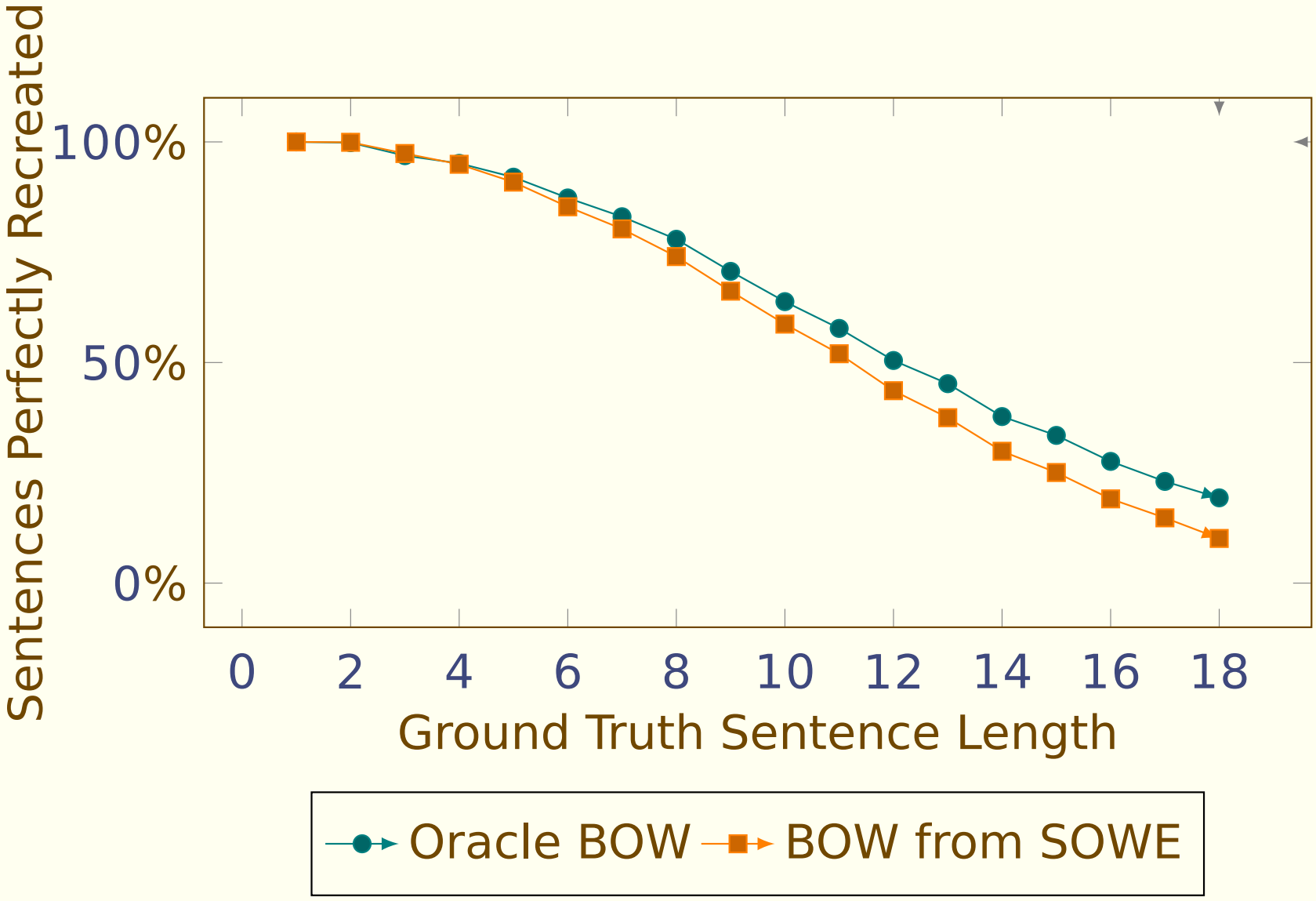
For trigram language models, word ordering is a variation of travelling salesman



How well can word order be recovered?



How well can word order be recovered?



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