

Natural language is
unreasonably simple, unreasonably often
Adding up word embeddings works far too well,
why is that?

Lyndon White

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

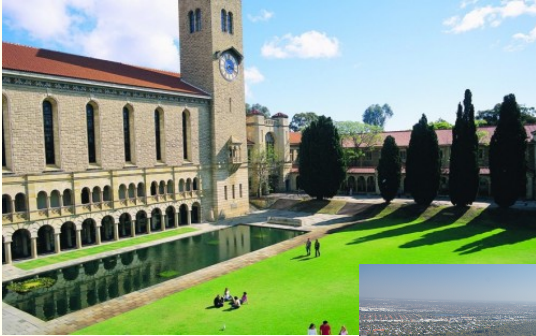


THE UNIVERSITY OF
**WESTERN
AUSTRALIA**

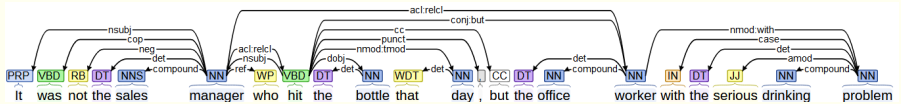
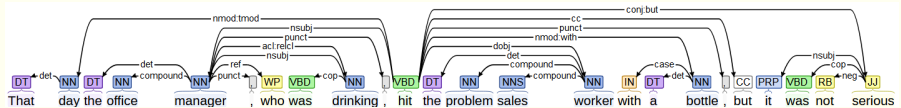
Australia, really it is quiet far away



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

color modifier head color
dark greenish blue
basic modifier meta modifier

color modifier head color
(red) (from noun)
rudd y coral
meta modifier

color modifier head color
dark bluish green
basic modifier meta modifier

and so using complicated methods leads to
worse performance.

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

Cířka and Bojar (2018), “Are BLEU and Meaning
Representation in Opposition?”

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

Conneau et al. (2018), “What you can cram into a single
\&!#* vector: Probing sentence embeddings for linguistic
properties”

*“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

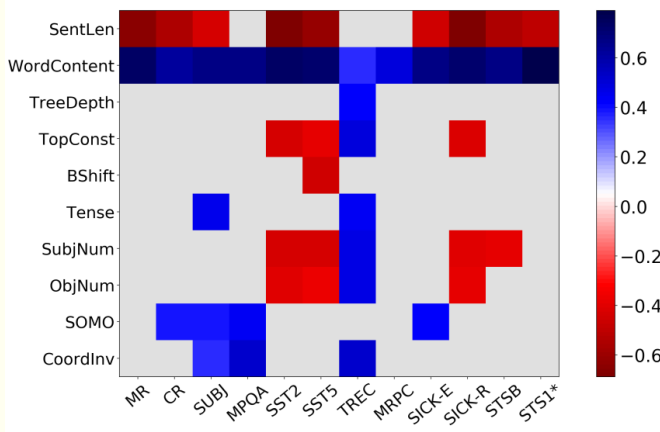
Considering further Conneau et al. 2018

- ▶ 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

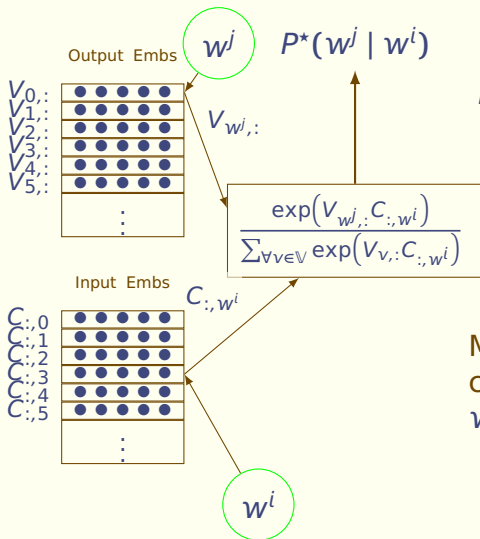
Content is king

Considering further [Conneau et al. 2018](#) 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

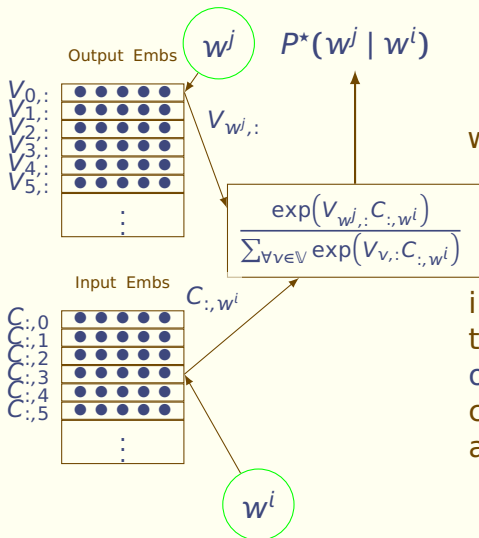
SkipGram is the most well known of recent word embedding methods



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

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

SkipGram is the most well known of recent word embedding methods



$P^*(w^j | w^i)$ 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 an iterative algorithm for weighted collocation matrix factorization.

V is a $300 \times \mathbb{V} $ input embeddings matrix	$Loss \propto -X \odot \exp(VC)$
C is a $ \mathbb{V} \times 300$ output embeddings matrix	$\propto VC - \log X$
X is a $ \mathbb{V} \times \mathbb{V} $ collocation count matrix	$\propto VC - f(X)$
f is some monotonic weighting function.	
(Levy, Goldberg, and Dagan 2015)	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 $\tilde{e}_w = [0, \dots, \underbrace{1}_{\text{with position}}, 0, \dots, 0]$

It's word embedding is given by $C_{:,w} = C^T e_w$

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, \dots)

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:

$$\sum_{\forall i} C_{:, w^i} = C^T \sum_{\forall i} \tilde{e}_{w^i}$$

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

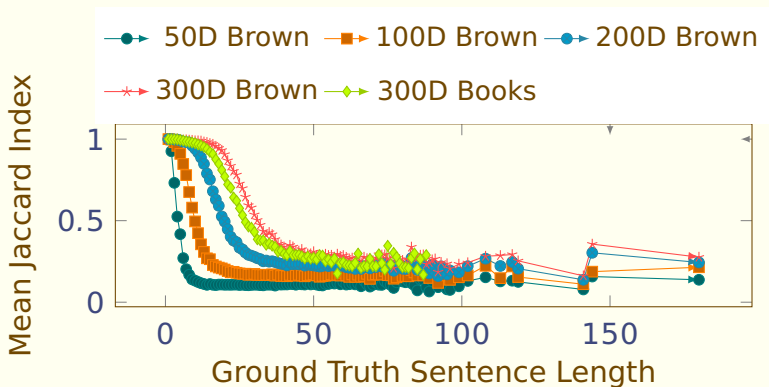
- ▶ A bag of words captures all unigram information in a sentence/document etc
- ▶ A greedy method can (mostly) recover the BOW from a SOWE

White et al. (2016), “Generating Bags of Words from the Sum of their Word Embeddings”

1. Greedily find word embeddings 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.

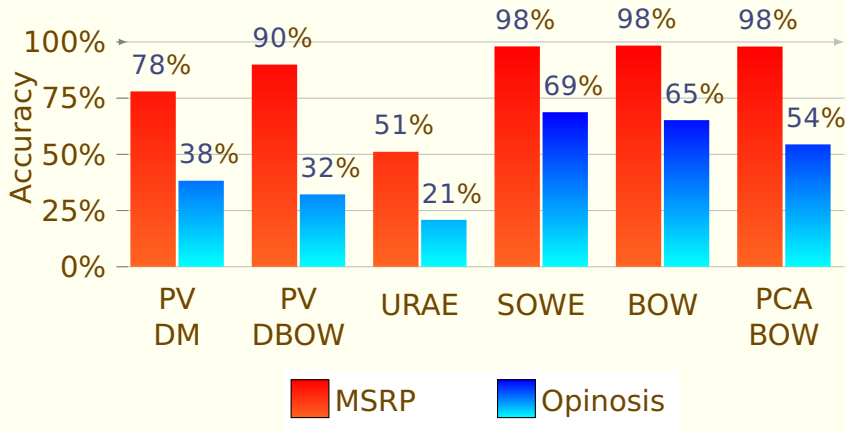
What does it mean to partition readily?

There are many ways one could define the quality of a space, on its ability to be partitioned according to another 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

Le and Mikolov (2014), “Distributed Representations of Sentences and Documents”

“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 Mesnil et al. 2014

URAE

Socher et al. (2011), “Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection”

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 Hotelling 1933

Drop down to 300D to match others

PCA is inferior to GloVe as a dimensionality reduction technique, for semantic preservation.

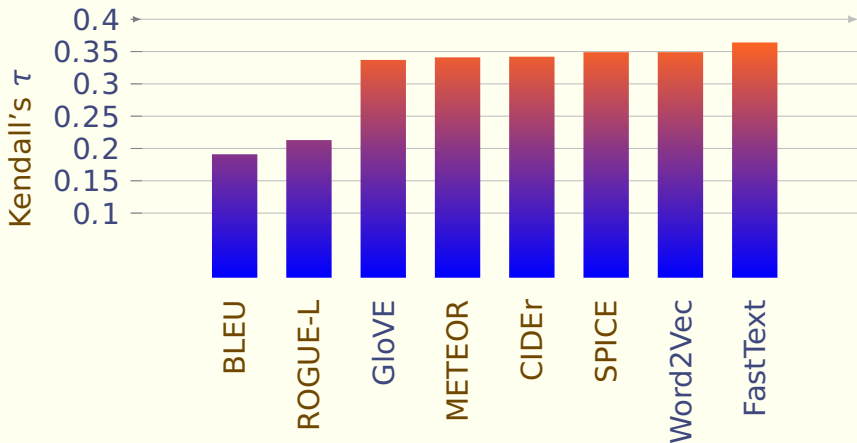
Consider machine captioning evaluation

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

- ▶ 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.

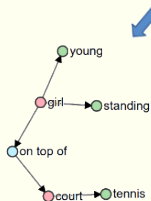
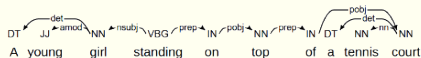
Captioning quality can be assessed on
fluency and on **adequacy**

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 .
(Anderson et al. 2016)

You could use AMR (Banarescu et al. 2013), or ERS (Bender 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 order**

The proper way to assess adequacy is a semantic 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.

Semantic graph comes from syntactic 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

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?

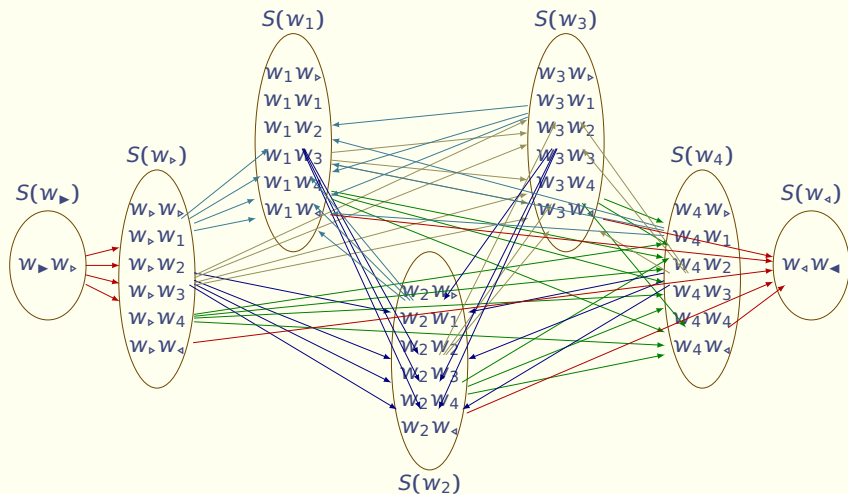
You can recover most probable sentence order from bag of words

White et al. (2016), “Modelling Sentence Generation from
Sum of Word Embedding Vectors as a Mixed Integer
Programming Problem”

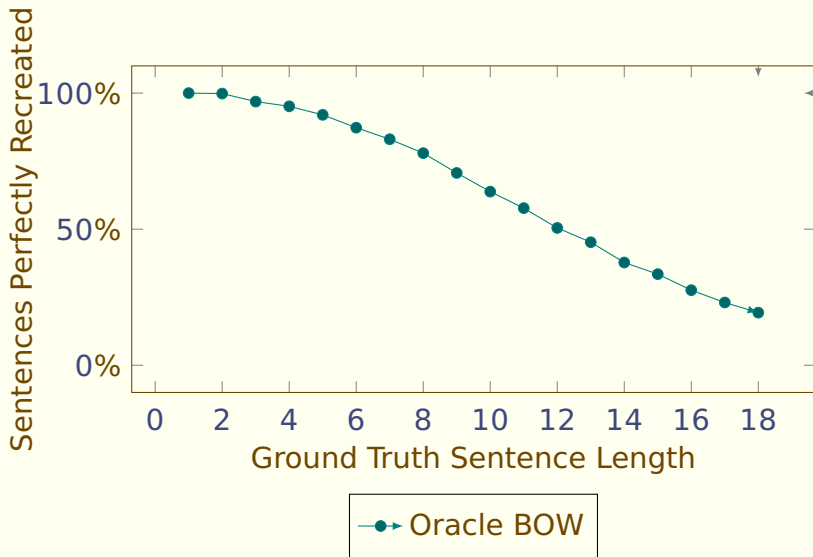
- ▶ Using a **language model** you can evaluate the probability of any given order for a bag of words.
- ▶ With an appropriate **optimization/search** technique you can find the most likely order
- ▶ This particular work was a simple **trigram language model**

Not a great way to generate sentences,
though

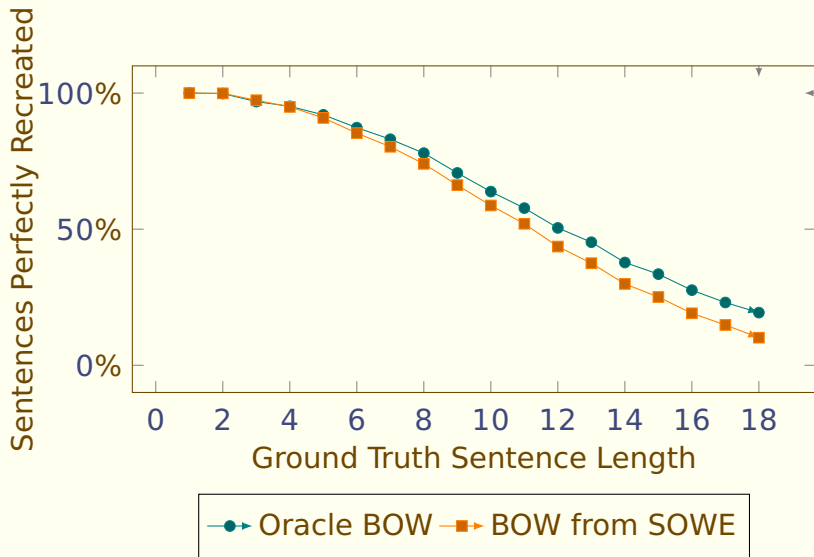
For trigram language models, word ordering is a variation of travelling salesman



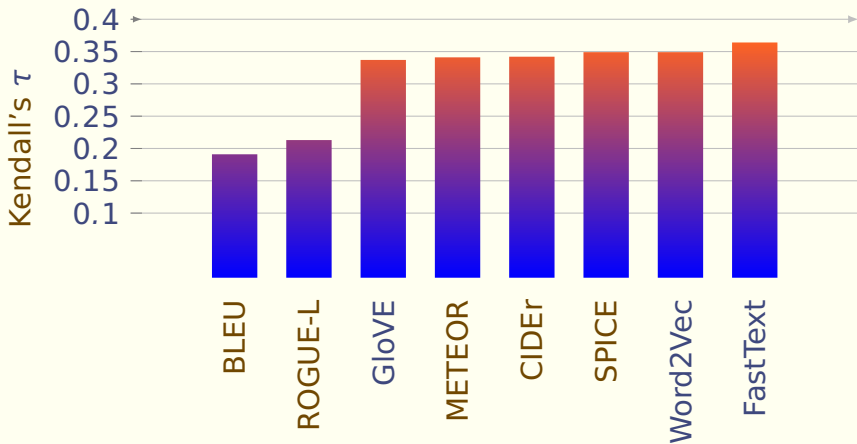
How well can word order be recovered?



How well can word order be recovered?



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**

- ▶ A caption being close in SOWE space to reference, means
 - ▶ Same word content
 - ▶ One of only a small number of possible orders

Colors look like a complex subdomain

- ▶ A lot of structure
- ▶ A lot of words used in different ways
 - ▶ noun-adjective dual identity
 - ▶ arbitrary nouns being used as color words
 - ▶ modifiers, meta-modifiers
 - ▶ Order matters for identity

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

color modifier head color
dark greenish blue
basic modifier meta modifier

color modifier head color
(red) (from noun)
rudd y coral
meta modifier

color modifier head color
dark bluish green
basic modifier meta modifier

and so using complicated methods leads to
worse performance.

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

- ▶ Often NLP is an engineering discipline
- ▶ and Computational Linguistics is maybe a scientific one
- ▶ NLP: what can linguistics do for you
- ▶ CL: what can you do for linguistics

As engineers doing something useful is what matters.

but practically we are certainly doing something useful

We need **adversarial** examples to advance the field

- ▶ People use words to communicate – they are trying to be clear.
- ▶ Adversarial examples are easy enough to create, but rare in-practice.
- ▶ Handling them can cause problems that overall decrease performance.

but do not forget the **non-adversarial majority**