Natural Language Processing

Tianchu Zhao@uts.edu.au

Stages

- Text processing
- Text embedding (turn text to vectors)
- Modelling

NLP task:
Mostly solved:

Spam detection

BUY! V1GGG

spam

Part-of-speech(POS) tagging

sheep is eating green grass

noun verb adj noun

Named entity recognition (NER)

I saw Einstein's statue in Princeton

Person Object Location

Making very good progress:

Sentiment analysis

The package feels flimsy

negative

Coreference resolution

Peter told Parker he should do his homework

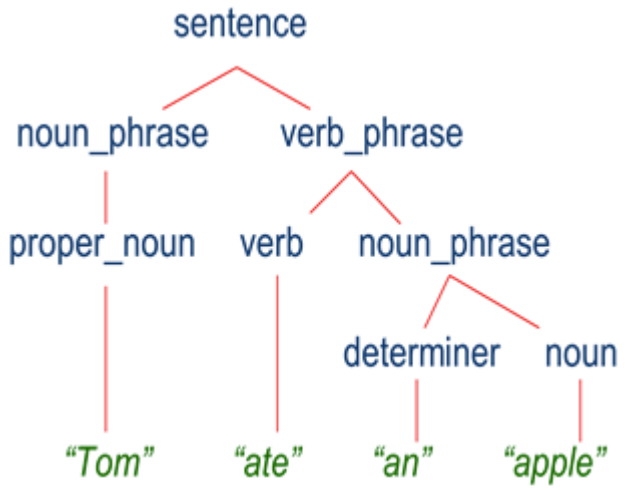
he? he?

Word sense disambiguation (WSD)

The new mouse looks so cool

?animal/computer

Parsing



Machine Translation

Je suis allé au centre commercial

I went to the mall

information extraction (IE)

I will see you tomorrow at 9:30

create new calendar entry

Hard

Question answering

How is the quantum computing and quantum physics related

Paraphrase

ABC acquired XYZ yesterday

XYZ has been taken over by ABC

Summarisation

The Dow Jones is up, The S&P500 jumped

economy is good

Dialog

When's the next Swan's game

xx/xx xx:xx at SCG

What makes Natural language understanding difficult

Non-standard english

U2

segmentation issues

the | New York|-New | Haven | Railroad

the | New| York-New | Haven | Railroad

idioms

get cold feet

neologisms

unfriend

world knowledge

Mary and Sue are sisters

Mary and Sue are mothers

tricky entity names

Let It Be was recorded in xxxx

Technique

- probabilistic models from language data
- neural network embedding from language data

Text processing
Regular expression

use expression to capture words

woodchuck

woodchucks

Woodchuck

Woodchucks

Word Tokenization

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.

Issues:

Finland's captial -> Finland Finlands Finland's ?

San Francisco -> one token or two ?

m.p.h., PhD. -> ??

other languages...

Word Normalization

U.S.A and USA

windows, window, Windows

Word Normalisation: Case folding

all lower case:

upper case in mid sentence

eg, US vs us

Lemmatization

to reduce inflections or variant forms to base form

am, are, is -> be

car, cars, car's -> car

Morphology

Morphemes:

The small meaningful units that make up words

Stems: The core meaning-bearing units

Affixes: Bits and pieces that adhere to stems

Often with grammatical functions

Stemming is crude chopping of affixes

eg reduce terms to their stems in information retrieval

automate(s), automatic, automation ->

automat

for example compressed and compression are both accepted as equivalent to compress ->

for exampl compress and compress ar both accept as equival to compress

Sentence Segmentation

!, ? relatively unambiguous

. ambiguous

- Sentence boundary
- Abbreviation
- Number decimals

Word representation one hot encoding

if man is word 5300th
it can be represented as
[0,0,0,…,1,…,0,0]
if woman is word 9000th
it can be represented as
[0,0,0,…,0,1,…,0]

- the disadvantage of this embedding is that the algorithm will treat each word as itself, doesn't generalise across words

for example

- if the algorithm learns
- I want a glass of orange juice
- it is hard for the algorithm to predict that the next word in the following sentence is juice
- I want a glass of apple ____

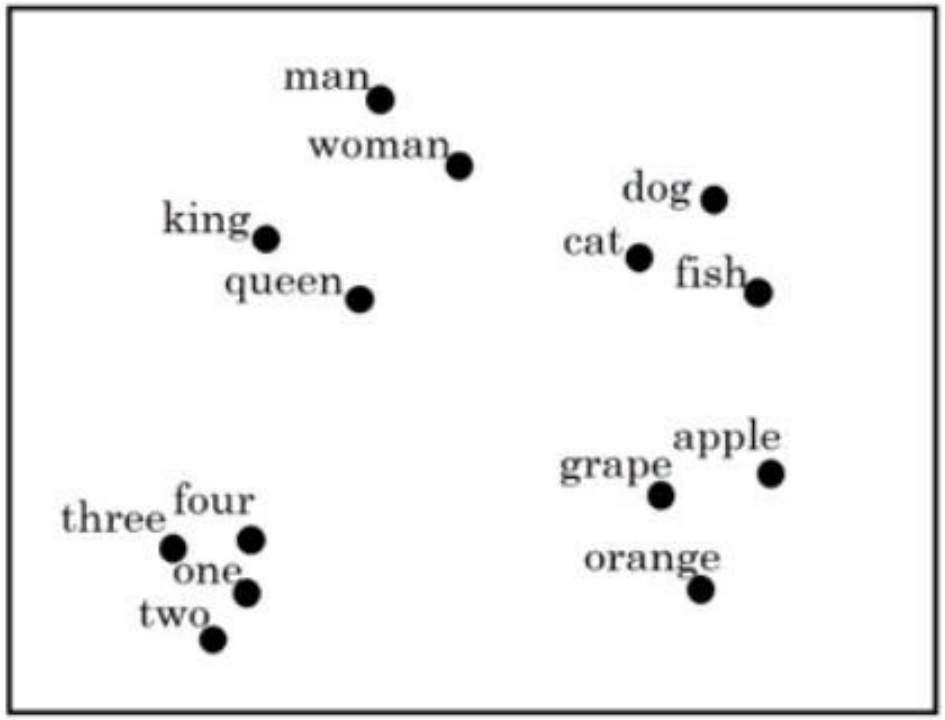
- That is because the any product between any two different one-hot vector is zero
- intuitively: distance between any word is the same

Featurised representation

	Man	Woman	Apple	Orange
Gender	-1	1	0	0
Food				
Age				
Quality				

- so that man is represented by a n-row features
- and we will notice that Apple and Orange has a similar values
- This increase odd the learning algorithm to put juice after apple

- If we learn the feature embedding then perform dimension reduction on the embedding and plot it
- we will notice that there exist group of words that are similar to each other
- eg usage NER



Transfer learning and word embeddings

- learn word embedding from large text corpus (1-100b words)

  (or download pretrain models online)
- transfer embedding to new task with smaller training set (100k words)
- Optional: Continue to finetune the word embeddings with new data

Embedding properties

- man -> woman
- king -> ?
- embed man - embed woman = [-2,0,0,0..]
- give king, find another vector that result in [-2,0,0,0..]

- cosine function calculates similarity

$$sim(u,v) = \frac{u^T v}{||u||_2 ||v||_2}$$

How do we learn word embedding

- Neural language model is able to use words around a word to predict unseen words, we use this to learn embedding

- Context word, e.g. 4 words on left & right
- Model: Context word vectors (one-hot encoded) ->
  Embedding Matrix ->
  Embedded vectors ->
  1 layer Neurons ->
  Softmax ->
  Target word class

Word2vec

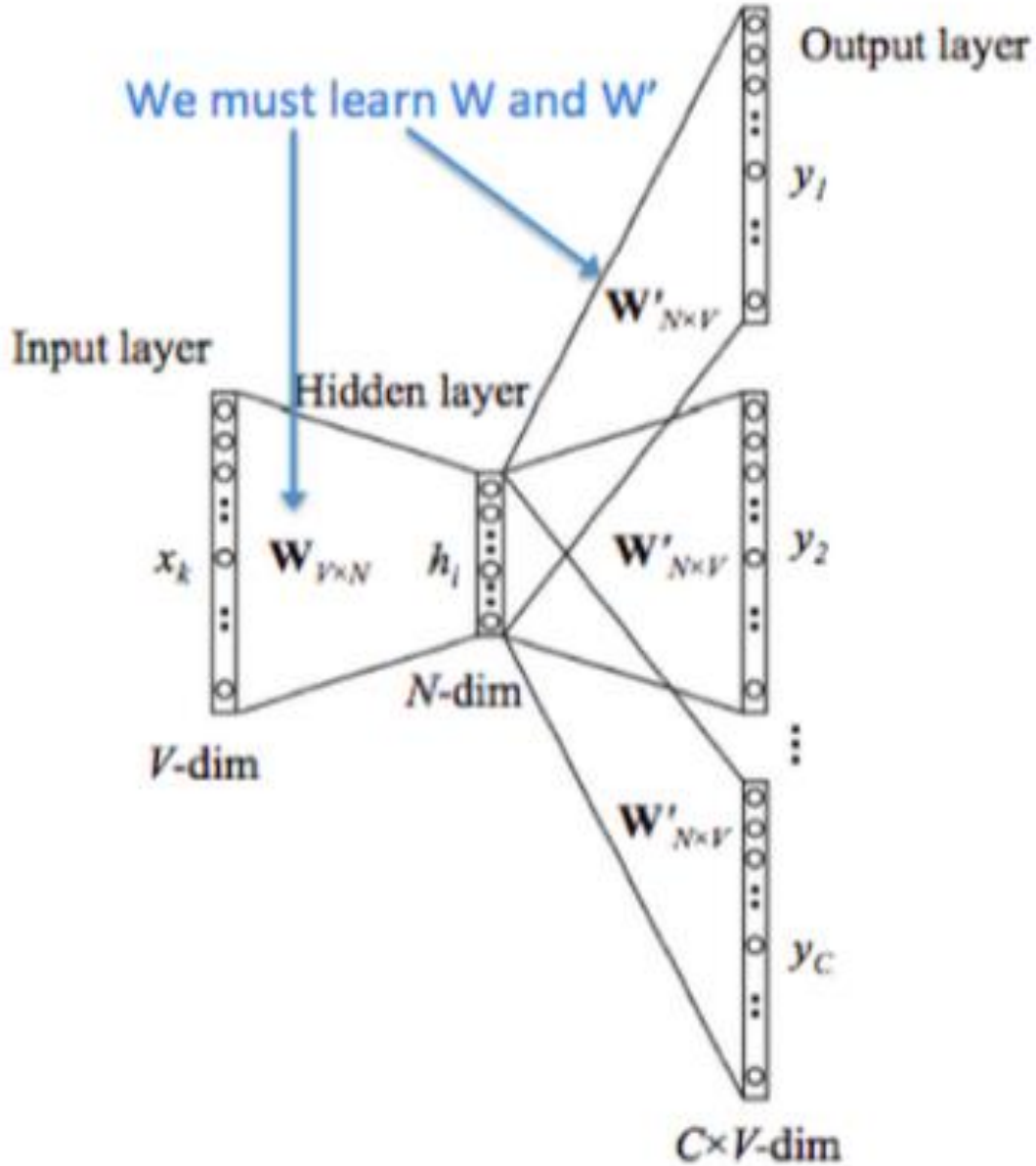
- Skip gram
- Continuous Bag of words (CBOW)

Skip-gram

- more efficient than the vanilla neural language model
- use target to predict context
- to create Target to Context model to train supervised learning

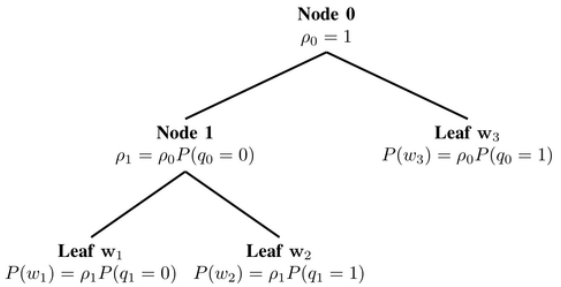
- eg, I want to pass the interview tomorrow
- For context Instead of picking pre4words or post4words immediately after
- say we randomly pick a word
- pass
- The target word is a random word within a window (say 2)
- say want/to/the/interview

- the goal is not do well on prediction but to learn the embedding vector


We must learn W and W'
Output layer
y_1
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
y_2
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
y_C
C×V-dim

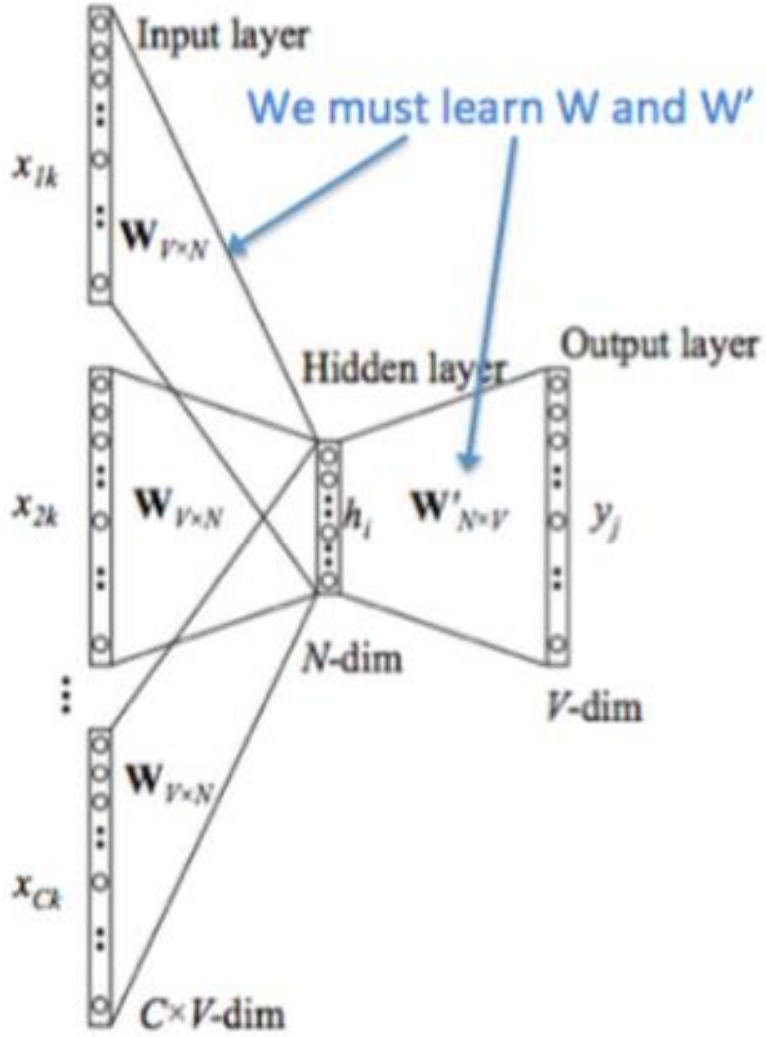
- **problem**
- computational speed
- softmax is very slow (summation denominator)
- **solution:**
- hierarchical softmax
- classifier1: first 5000 words, second 5000 words
- classifier2: first 2500, second 2500 words
- building a tree
- computation cost become log(number of words)
- usually not symmetric tree, asymmetric with common words on top (leaf)

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$



Continuous Bag of words (CBOW)

- reverse skip-gram
- use context to predict target



- **Skip-gram:** works well with small amount of the training data, represents well even rare words or phrases.
- **CBOW:** several times faster to train than the skip-gram, slightly better accuracy for the frequent words

Negative sampling

- modify learning problem, that does task similar to skip gram but much more efficient
- given two words, predict if they are a context target pair
- eg, orange, juice ->1
- eg, orange, man ->0
- first pick a target word that is actual pair
- next random pick k words
- Model: onehot -> E matrix -> embed vector -> n(corpus size) logistic classification problem (is the word the target word or not)
- each iteration we only train 5(k) logistic instead of n

Glove (global vectors for word representation)

$$J = \sum_{i=1}^{N} \sum_{j=1}^{N} f(X_{ij})(\theta_i' e_j + b_i + b_j - log(X_{ij}))^2$$

- X_{ij} : number of times i appears in context j
- θ_i : i vector
- e_j : j vector
- b_i b_j: bias for word i, j
- $f(X_{ij})$: =0 when X_{ij} = 0 to avoid $\log(X_{ij})$ error or weight in normal situation

Probability language model

- Goal: compute probability of a sentence/sequence of words

Backgrounds on probability

probability measures the likelihood an event will occur

$$P(A)$$

conditional probability measures the likelihood an event will occur given (by assumption) that another event have occurred

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

probability of both A and B occur

Bayes' theorem
that is express as follows

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{\frac{P(B \cap A)}{P(A)} \cdot P(A)}{P(B)}$$

Probability language model

- Goal: compute probability of a sentence/sequence of words
- $P(W) = P(w_1,w_2,w_3,...,w_n)$
- or
- $P(w_4|w_1,w_2,w_3)$

How to estimate probabilities

- $P(interview \mid I\ want\ to\ pass\ the)$
- $= \frac{Count(I\ want\ to\ pass\ the\ interview)}{Count(I\ want\ to\ pass\ the)}$
- doesn't work, too many possibilities

- **Markov assumption**
- $P(interview \mid I\ want\ to\ pass\ the) \approx P(interview \mid the)$
- or
- $P(interview \mid I\ want\ to\ pass\ the) \approx P(interview \mid pass\ the)$

Markov Assumption

$$P(w_1, w_2, \ldots, w_n) \approx \prod_i P(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

We can approximate each word in the product as

$$P(w_{\mathrm{i}} \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

Unigram, Bigram, N-gram

- Unigram: the simplest case of markov model
- $P(w_1, w_2, \ldots, w_n) \approx \prod_i P(w_i)$
- this doesn't generate meaningful sentence

- Bigram: condition on previous word
- $P(w_\mathrm{i} \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-1})$

- N-gram
- $P(w_\mathrm{i} \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-n}, \ldots, w_{i-1})$

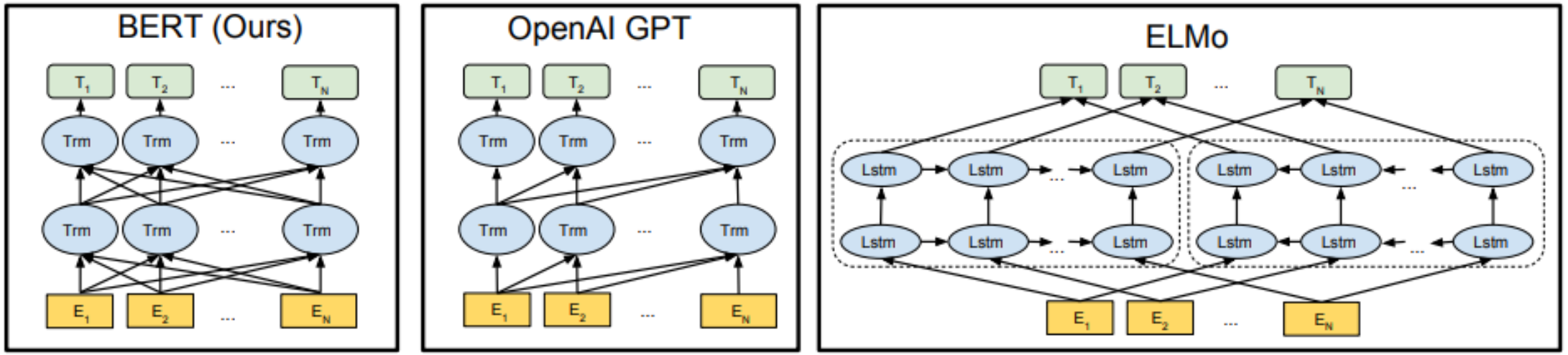
FastText

- was developed by the team of Tomas Mikolov who proposed the w2v in 2013
- compare to word2vec, which treats each word in the corpus as an atomic entity and generates vector for each word, fasttext uses n-gram
- eg. tri-gram for apple is app, ppl, ple, the word embedding vector for apple will be the sum of all these n-grams
- After training the Neural Network, we will have word embeddings for all the n-grams given the training dataset.

- Rare words can now be properly represented since it is highly likely that some of their n-grams also appears in other words, therefore can be reconstruct

State of the art (word embedding)

- BERT Deep contextualized word representations (few days ago)
- ELMo (Embeddings from Language Models)

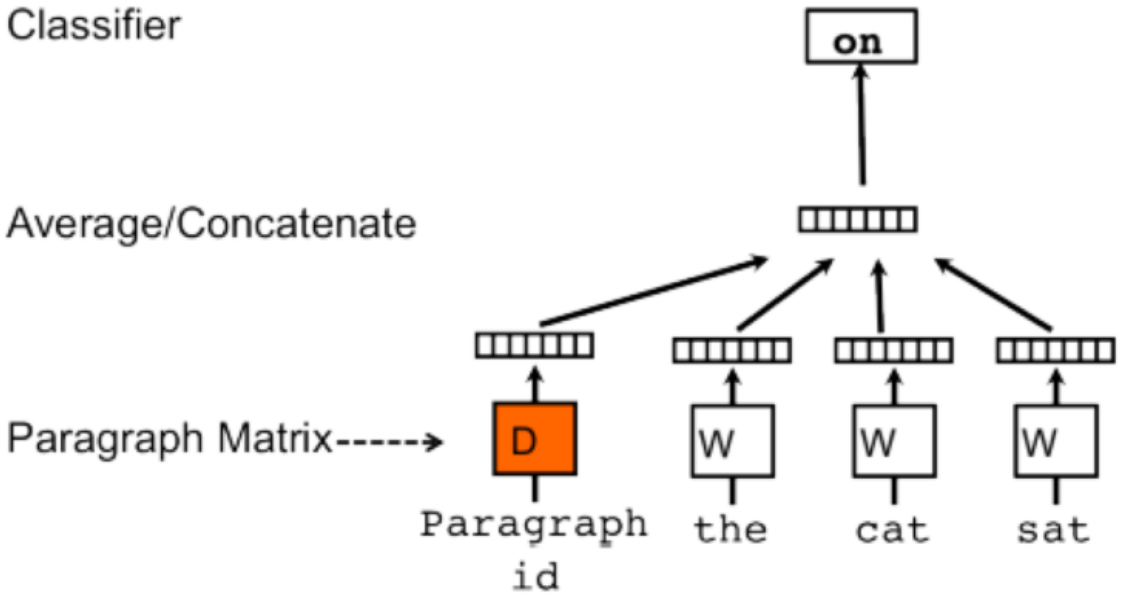


Sentence Embedding

- State of the art
- Unsupervised
  - Skip-Thoughts
  - Quick-Thoughts
- Supervised
  - InferSent
  - Multi-task learning
  - MILA/MSR's General Purpose Sent
  - Google's Universal Sentence Enc.

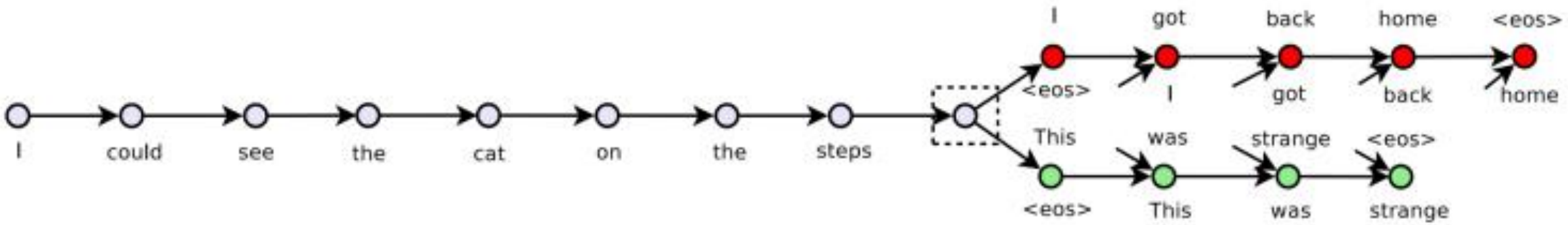
doc2vec

- an extension of word2vec
- Paragraph/sentence vectors extend this by training paragraph vectors as **auxiliary information** during this prediction task.



Skip-Thought Vectors

- word2vec for sentences
- predicts surrounding sentences from a given sentence
- using encoder-decoder framework



Models

- Conditional Random Field (CRF)
- LSTM variation