



CIS 522: Lecture 11

Introduction to Natural Language Processing
02/20/20, @kordinglab

Feedback

If you did not take 519/520. Go read up on what you are missing. See the TAs. Go to recitations. Go to office hours. Watch the recitation recordings / go to recitation (very few people are watching the recordings)!

Want to see implementations!

Go to the recitations. All of them.

Want more math!

March 3 and March 5 will backfill a lot of that

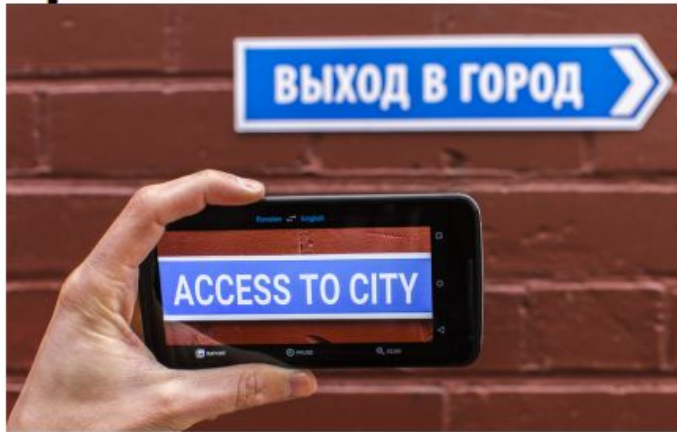
HW feedback

Pollev

Outline

- Vectors for words -> meaning
 - Defining similarity: From word vectors to meaning vectors
 - Many flavors
 - Handling Variable Length Vectors
- Using vectors to solve NLP problems

Why Natural Language Processing?



To make machines comprehend human languages and communicate with us.

About 1,240,000,000 results (1.23 seconds)

You are not **required** to know **Machine Learning** for **learning Deep Learning** but I strongly recommend to first start with **Machine Learning** and then **deep** dive into **Machine Learning**. ... So in case you **do** not have that complex problem or computing resources then I **will** recommend to use **Machine Learning** algorithms first.

[www.quora.com › Should-I-learn-machine-learning-or-deep-learning](https://www.quora.com/Should-I-learn-machine-learning-or-deep-learning)[Should I learn machine learning or deep learning? - Quora](https://www.quora.com/Should-I-learn-machine-learning-or-deep-learning)

About Featured Snippets



Feedback

People also ask

Can you learn deep learning without machine learning? How do I start learning deep learning? Should I learn machine learning or deep learning first? Is deep learning easy to learn? [Feedback](#)[towardsdatascience.com › dont-learn-deep-learning-d23485e4c1c4](https://towardsdatascience.com/dont-learn-deep-learning-d23485e4c1c4) 

Don't Learn Deep Learning - Towards Data Science

Jun 23, 2019 - **Deep Learning** is one of the biggest breakthroughs in **machine learning** in the ...However, what **needs** to be realised is that **Deep Learning** will, ...



The central intuitions and historical NLP



Representing Words as Vectors

I hot Word Representation

How to represent a word in a computer?

before the deep learning era, we tended to treat each word as an individual symbol, totally independent of others;

E.g. tw words: “hotel, motel”, in **one-hot encoding**

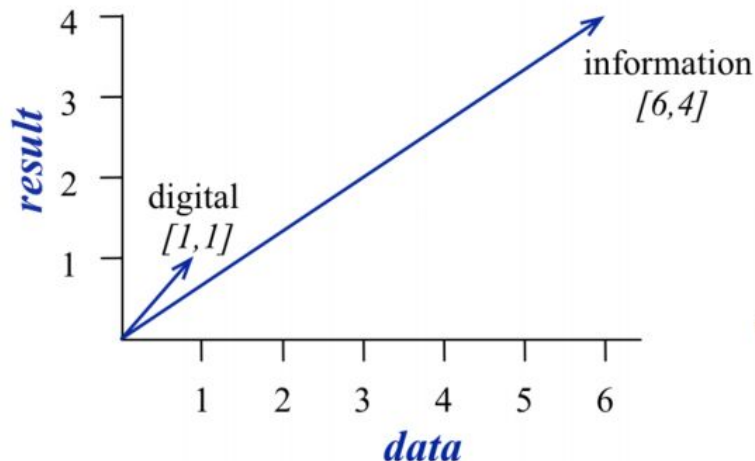
Hotel = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Motel = [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]

But there is no way to measure how similar these two vectors are.

How can we know that two words (or sentences) mean similar things

Ideally we would have a meaningful metric



$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^V u_i v_i}{\sqrt{\sum_{i=1}^V u_i^2} \sqrt{\sum_{i=1}^V v_i^2}}$$

The similarity between two words (cosine distance).

Outlier insensitive

What we want of a good similarity

Similar meaning should translate into close distance.

But watch out:

- Maybe we rather want structure

- Or form

- Or style

So be mindful of how you construct a metric

How to construct good similarity (in meaning) metric - PolIEV

Similarity by Context

C1: A bottle of ____ is on the table.

C2: Everybody likes ____.

C3: Don't have ____ before you drive.

C4: We make ____ out of corn.

	C1	C2	C3	C4
tejuino	1	1	1	1
loud	0	0	0	0
motor-oil	1	0	0	0
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

An important assumption: words that occur in similar context tends to have similar meanings. --- somewhat infeasible

E.g. as local co-occurrence

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

(Word-Word co-occurrence matrix)

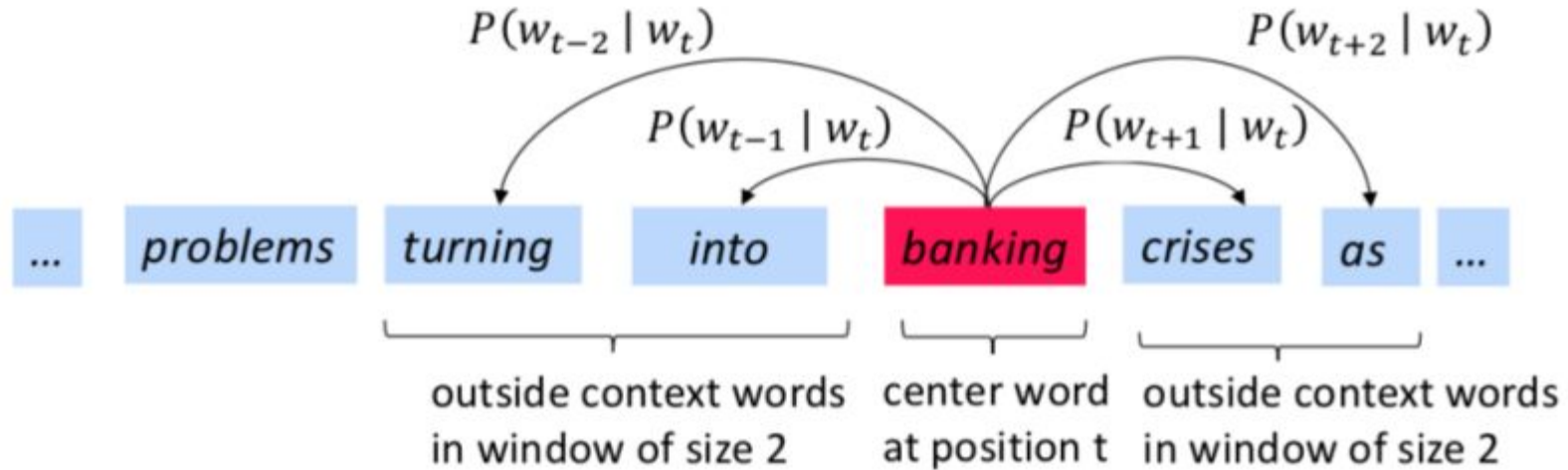
To transform a word into a vector by context: the meaning of a word is represented by the words in the neighbor, i.e. the same sentence, paragraph.

Similarity by predictability

Intro to what it means to have good similarity

More meaningful than just context

Local correlations



The essence: to predict any word by the context: a fixed-size window

Historical Word2Vec

The objective function for skip-gram: to predict all the words given the context.

$$\mathcal{L}(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$$

all the parameters to be optimized

The core problem then is about how to define the conditional probability $P(w)$.

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

“softmax”

$\mathbf{u}_i \in \mathbb{R}^d$: embedding for target word i

$\mathbf{v}_{i'} \in \mathbb{R}^d$: embedding for context word i'

Learning Word2Vec

We calculate the gradients of negative log-likelihood with respect to the parameters: in this case, the embedding vectors of each word.

$$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$$

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t; \theta)$$

Then, we just simply run SGD.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} J(\theta)$$

Word2Vec Visualization

$$\text{employees} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 10.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



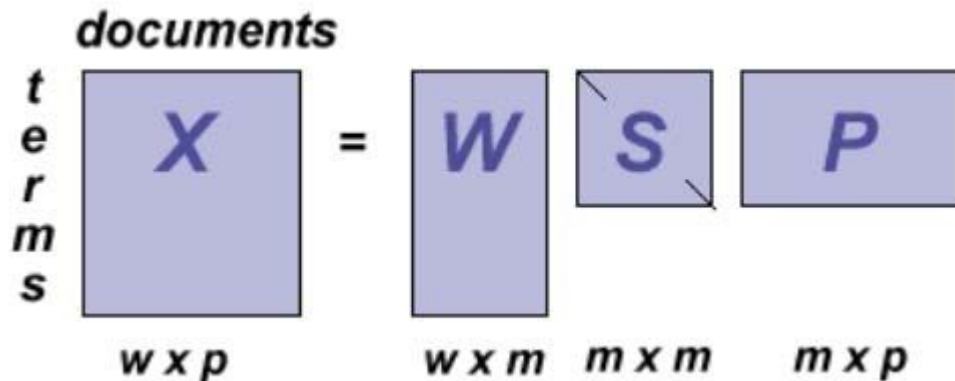
A two dimensional visualization of the embedding of similar words.

How to make such analyses? LSA

Latent semantic analysis

Project into lower-d space.

In a way that leaves distances as similar as possible



Not all words are equally important

E.g. predicting that after a word comes “the” is less important than predicting that after a word comes “variational”

TF-IDF (some words are more important than others)

How to measure the relative importance of a word?

$$\text{Term frequency} = \frac{\text{number of times the word appears in a document}}{\text{Total number of words in the document}}$$

$$\text{Inverse data frequency} = \log \frac{\text{Total number of documents}}{\text{Number of documents containing the word}}$$

$$\text{TF-IDF score} = \text{Term frequency} \times \text{Inverse data frequency}$$

Penn-wide calendar (<http://35.160.123.103/>)

[Home](#)[Selected Events](#)[Recommendations](#)[Usage](#)

Search Results



Search results for Neuroscience

Wed, Feb 19, 2020

04:00 PM **Physics Colloquium**

05:00 PM unknown



Department of Physics and Astronomy



04:00 PM **Mahoney Institute for Neurosciences Seminar - Quan Yuan**

05:00 PM Room 140, 3620 Hamilton Walk



Mahoney Institute for Neuroscience (MINS)



Wed, Feb 26, 2020

04:00 PM **Mahoney Institute for Neurosciences Seminar - Christopher Doe**

05:00 PM Room 140, 3620 Hamilton Walk



Mahoney Institute for Neuroscience (MINS)



Recommendations

 Fri, Feb 21, 2020

12:00 PM

MindCORE Seminar: Ueli Rutishauser  

01:30 PM

 SAIL Room, 425 S. University Avenue, Philadelphia, PA 19104 111 Levin Building



MindCORE



 Mon, Feb 24, 2020

03:30 PM

Psychology Colloquium: Lila Davachi  

04:30 PM

 Levin Auditorium, 425 S. University Avenue, Philadelphia, PA 19104



MindCORE



 Fri, Feb 28, 2020

12:00 PM

MindCORE Seminar: Todd Rogers  

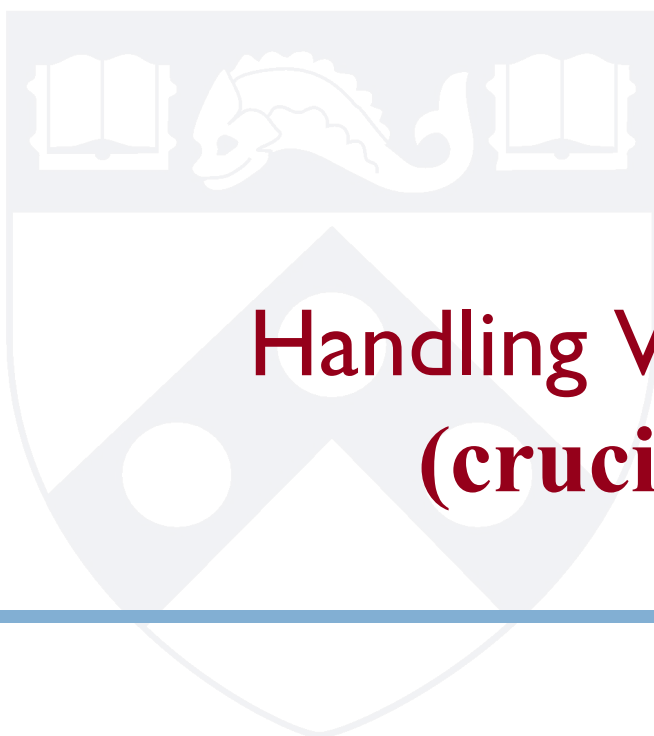
01:30 PM

 SAIL Room, 425 S. University Avenue, Philadelphia, PA 19104 111 Levin Building

Two very similar ways of reading a sentence

Siri: “Tell my wife I love her”

All the simple ways of preprocessing text can not handle this



Handling Variable Length Vectors? (crucial in modern NLP)

Handling Variable Length Vectors

It is often the case that sentences or documents are not of a fixed length.

Working with variable length vectors reduces the ability to process data as a batch

Increases the computational overhead as the size of the documents might vary from a few words to millions of words

How to deal with Variable Length

Some of the common practices to handle variable length vectors are:

- Truncation: chop off the vector.
- Padding: padding the vectors with zeros.
- Bag-of-words: turn the vector into a count-dict.
- Convolution: convolute over a fixed-length filter and aggregate the results.
- Recurrence: make your forward pass variable-length.

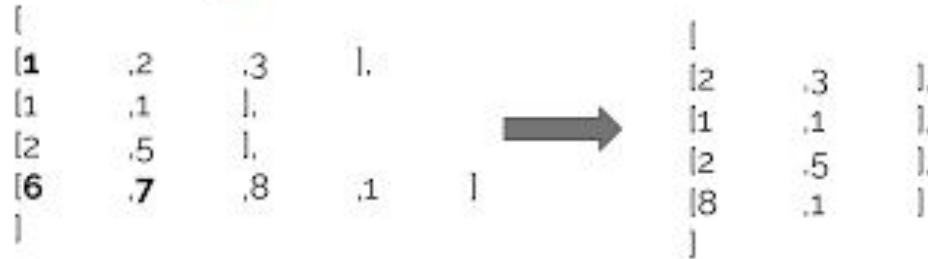
Truncation

As the name suggests, the sentence vectors are truncated to the desired length

Pre Truncation

The sentence vectors are truncated from the beginning

Pre Sequence Truncation

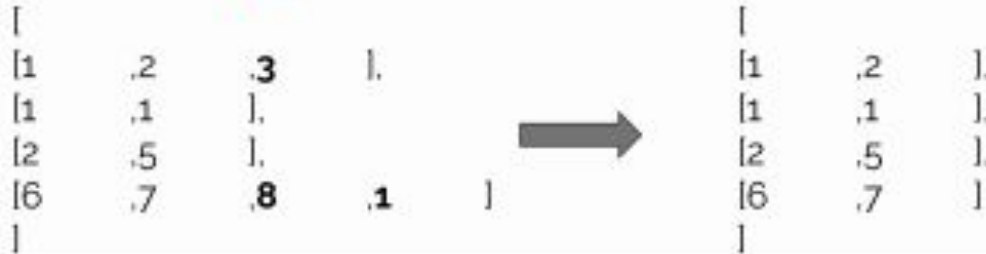


Truncation

Post Truncation

The sentence vectors are truncate from the end

Post Sequence Truncation

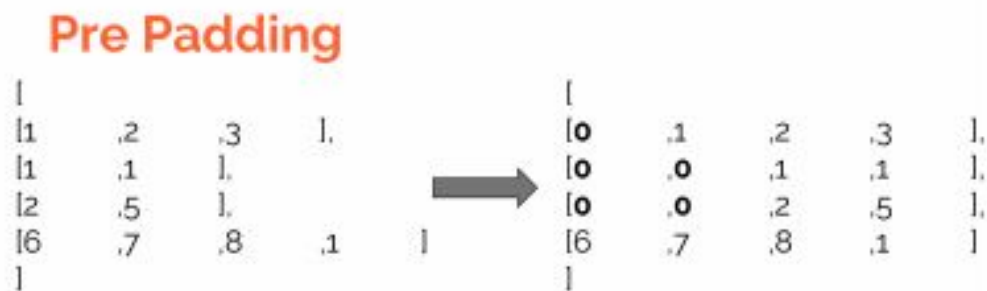


Padding

Padding a sentence vector would pad the sequence with zeros to a desirable length. Similar to padding an image during convolution

Pre Padding

Pad the sequence towards the beginning



Padding

Post Padding

Pad the slides towards the end



Padding

Pack Padded Sequence

Padding results in unnecessary computations due to addition of zeros to the variable length sentence vectors.

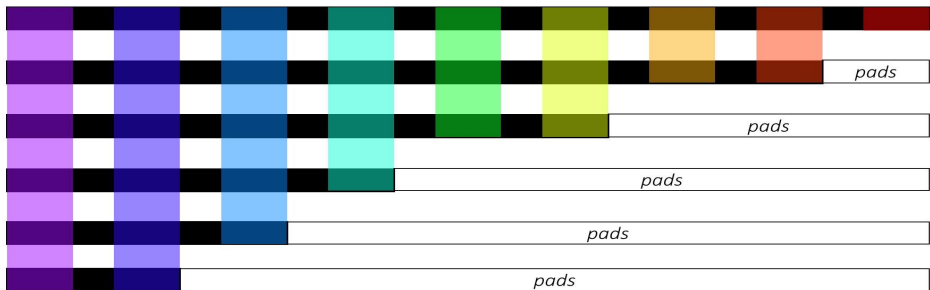
Hence it is useful to maintain a list of non-zero elements interleaved by timesteps, and the size of the batch.

Keep track of batch at each time step to remove computation over the padding value

Sparse matrices

Pack Padded Sequence

Padded sequences sorted by decreasing lengths



Packed sequences

`pack_padded_sequence()` flattens sorted sequences by timestep, keeping track of the effective batch size at each timestep



Bag-of-words

Convert the words in the sentence to a dictionary of frequencies.

Is used for document classification tasks.

Maintain a global list of all the words in the data and convert each document to a list of frequency vectors based on the global list

Bag of words

« The dog started to run after the cat, but the cat jumped over the fence. »
« The cat's food was eaten by the dog in a few seconds. »
« The cat attacked the bird the other day. »



[« dog », « start », « run », « cat », « cat », « jump », « fence »]
[« cat », « food », « eat », « dog », « second »]
[« cat », « attack », « bird », « day »]



Bag-Of-Words representation

	Dog	Start	Run	Cat	Jump	Fence	Food	Eat	Second	attack	Bird	Day
1	1	1	1	2	1	1	0	0	0	0	0	0
2	1	0	0	1	0	0	1	1	1	0	0	0
3	0	0	0	1	0	0	0	0	0	1	1	1

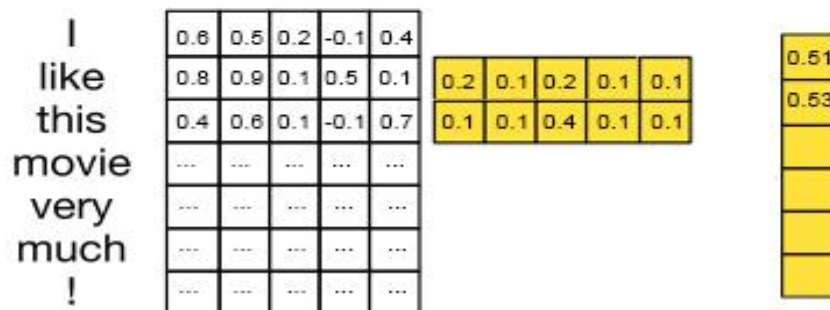
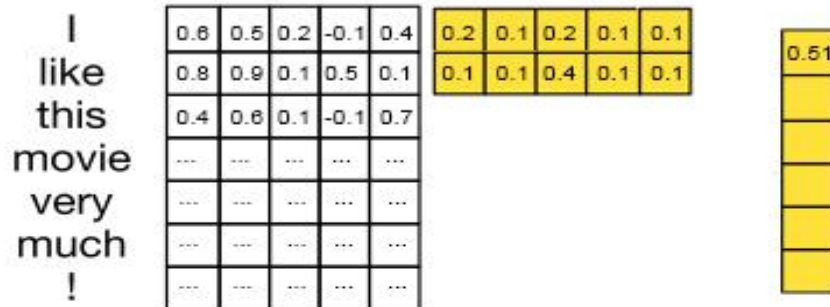
Convolution

Extract sentence representations by convolving over the sentence embedding

Apply convolutions of fixed size windows

Useful for n-gram models

Convolution



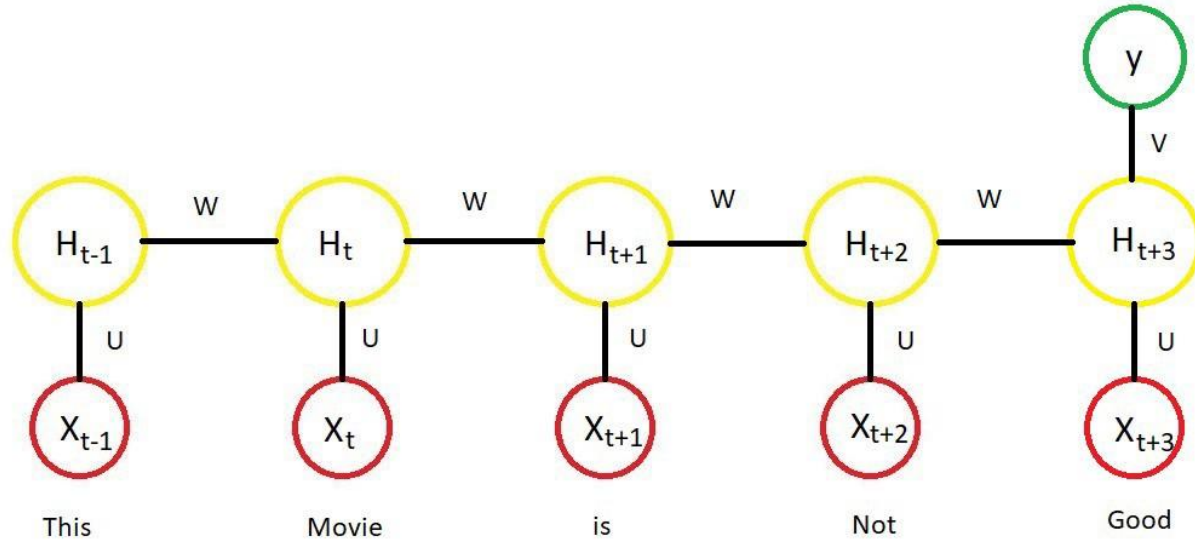
Recurrence

Useful when the order of the words matter.


Make the forward pass variable length and compute value of the current word based on the previous words.

Used in LSTM and RNN

Recurrence



We will talk about how to make RNNs work next week



**Now that we have input representation:
how to model the probability of a
sentence?**

The Probability of Occurrence

Language Models predict the probability of the next occurring word

More formally,

Given a sequence of words $x(1) \dots x(t)$, compute the probability of the occurrence of $x(t+1)$

$$P(\mathbf{x}^{(t+1)} \mid \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})$$

where $\mathbf{x}^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \dots, w_{|V|}\}$

The Probability of Occurrence


Language models also assign probability to the entire text.

Suppose the text contains words $\mathbf{x}^{(1)} \dots \mathbf{x}^{(t)}$. Then the probability of that text is

$$\begin{aligned} P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}) &= P(\mathbf{x}^{(1)}) \times P(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}) \times \dots \times P(\mathbf{x}^{(T)} | \mathbf{x}^{(T-1)}, \dots, \mathbf{x}^{(1)}) \\ &= \prod_{t=1}^T P(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) \end{aligned}$$

The probability of what we will type



san f 

- san francisco weather
- san francisco
- san francisco giants
- san fernando valley
- san francisco state university
- san francisco hotels
- san francisco 49ers
- san fernando
- san fernando mission
- san francisco zip code

The local content

N-Grams

N-grams refer to chunk of consecutive n words in the sentences

Can also be referred to as shingles.

Examples of n-grams:

uni-grams: “the”, “students”, “opened”, “their”

bi-grams: “the students”, “students opened”, “opened their”

tri-grams: “the students opened”, “students opened their”

4-grams: “the students opened their”

N-Grams

To calculate probability using N-grams we assume markov conditions, i.e each word depends on the preceding n-1 words

Easier to compute probability with markov assumptions

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(t+1)} | \overbrace{\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}}^{n-1 \text{ words}})$$

Computing the Conditional Probability

The conditional probability of the n-gram model can be computed by counting the number of occurrences in a large corpus of text

$$\approx \frac{\text{count}(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}{\text{count}(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})} \quad \text{(statistical approximation)}$$

Sparsity Problems

N- grams derived from the statistical approximation deal with the sparsity problem.

Due to the huge amounts of text, the probability of occurrence of some events might be zero. This results in zero probability of occurrence of those predicted words.

This occurs since the count if those sequences are zero.



Syntactical Analysis



Linguistics 101

Syntactical Analysis (Parsing)

Involves analysis and extracting meanings from the words

Syntax analysis checks the text for meaningfulness comparing to the rules of formal grammar

Parse Tree

Parse tree analyzes the sentence based on grammatical rules.

A particular parse tree indicates how a particular grammar analyzes the syntactic structure of a particular sentence.

Context Free Grammar

A context-free grammar are rules which define well formed english sentences. Each rule has a left-hand side, which identifies a syntactic category, and a right-hand side, which defines its alternative component parts, reading from left to right.

E.g., the rule $s \rightarrow np\ vp$ means that "a sentence is defined as a noun phrase followed by a verb phrase."

Context Free Grammar

Syntactic categories

np - noun phrase

vp - verb phrase

s - sentence

det - determiner (article)

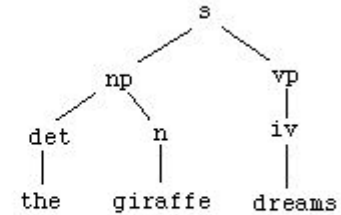
n - noun

tv - transitive verb (takes an object)

iv - intransitive verb

adj - adjective

s	→	np vp
np	→	det n
vp	→	tv np
	→	iv
det	→	the
	→	a
	→	an
n	→	giraffe
	→	apple
iv	→	dreams
tv	→	eats
	→	dreams



Dependency grammar

Dependency grammar is a grammar where words, are connected to each other by dependencies (directed links).

Dependency grammar describes the syntactic structure of a sentence in terms of the words and an associated set of directed binary grammatical relations that hold among the words.

Dependency grammar

Universal Dependencies

tmod: temporal modifier

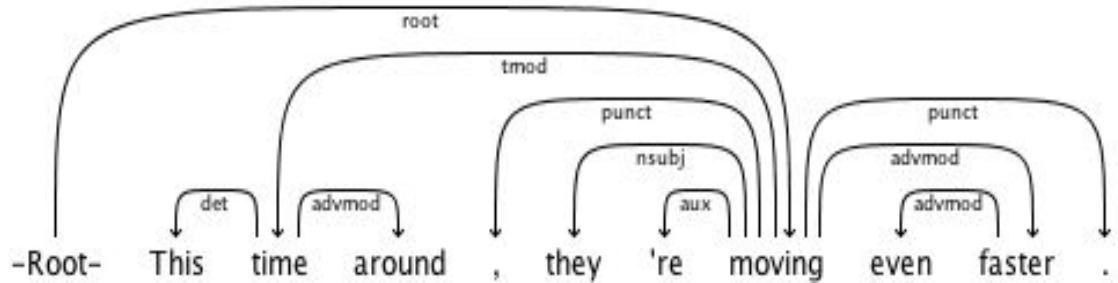
nsubj: nominal subject

advmod: adverbial modifier

punct: punctuation

det: determiner

aux: auxiliary



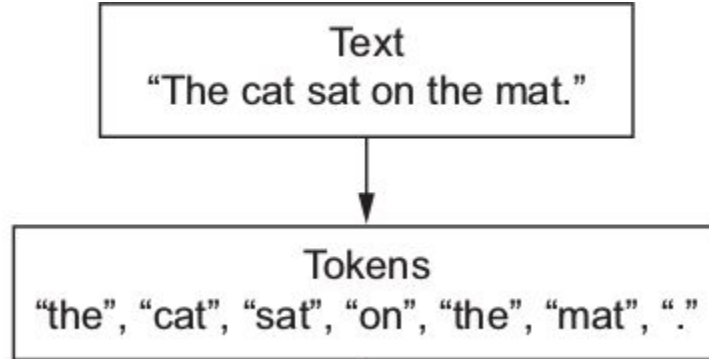


Text Processing

Tokenization

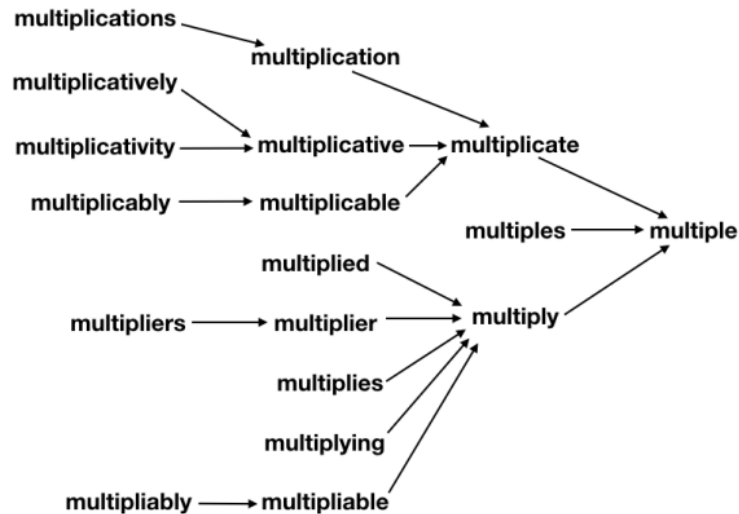
Tokenization involves splitting the sentence into pieces called tokens.

Tokenizations might also involve removal of characters such as punctuations while splitting sentences to tokens



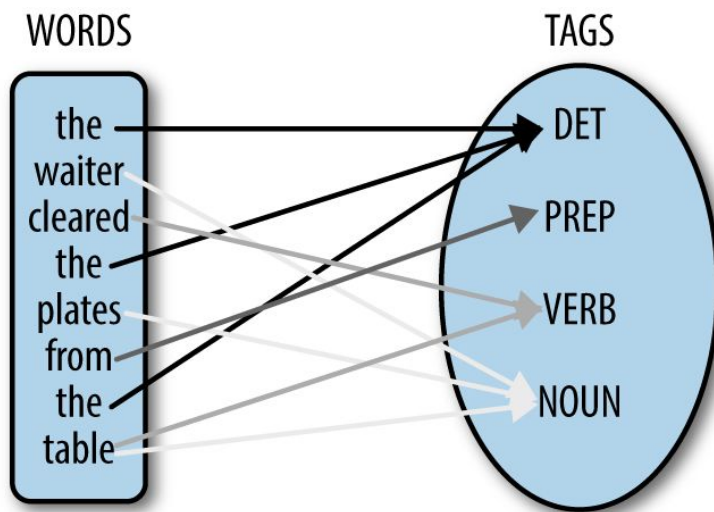
Lemmatization

Lemmatization involves removal of the inflectional endings only and to return the base or dictionary form of a word.



POS (parts of speech) Tagging

POS tagging is the process of marking the words in the corpus to its corresponding part of a speech tag (noun, verb, adjective, etc), based on its context and definition.



Named entity recognition (solved by convolution and recurrence)

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported **ORG** byF.B.I. Agent Peter Strzok **PERSON** ,
Who Criticized Trump **PERSON** in Texts, Is FiredImagePeter Strzok, a top F.B.I. **GPE** counterintelligence agent who was taken off the special counsel
investigation after his disparaging texts about President Trump **PERSON** were uncovered, was fired. CreditT.J. Kirkpatrick **PERSON** for The New York
TimesBy Adam Goldman **ORG** and Michael S. SchmidtAug **PERSON** . 13 **CARDINAL** , 2018WASHINGTON **CARDINAL** — Peter Strzok
PERSON , the F.B.I. **GPE** senior counterintelligence agent who disparaged President Trump **PERSON** in inflammatory text messages and helped
oversee the Hillary Clinton **PERSON** email and Russia **GPE** investigations, has been fired for violating bureau policies, Mr. Strzok **PERSON** 's lawyer
said Monday **DATE** .Mr. Trump and his allies seized on the texts — exchanged during the 2016 **DATE** campaign with a former F.B.I. **GPE** lawyer,
Lisa Page — in **PERSON** assailing the Russia **GPE** investigation as an illegitimate “witch hunt.” Mr. Strzok **PERSON** , who rose over 20 years
DATE at the F.B.I. **GPE** to become one of its most experienced counterintelligence agents, was a key figure in the early months **DATE** of the
inquiry.Along with writing the texts, Mr. Strzok **PERSON** was accused of sending a highly sensitive search warrant to his personal email account.The
F.B.I. **GPE** had been under immense political pressure by Mr. Trump **PERSON** to dismiss Mr. Strzok **PERSON** , who was removed last summer
DATE from the staff of the special counsel, Robert S. Mueller III **PERSON** . The president has repeatedly denounced Mr. Strzok **PERSON** in posts on

<https://arxiv.org/abs/1511.08308>

Sentiment analysis (solved by convolution or recurrence)

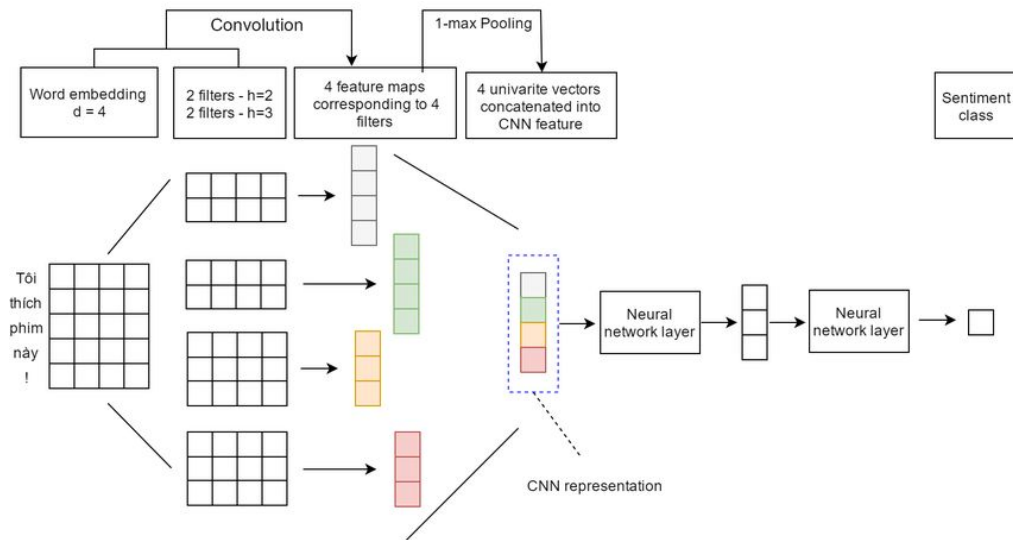


Table 1. Small dataset classification accuracies

METHOD	MR	CR	SUBJ	MPQA
NBSVM [49]	79.4	81.8	93.2	86.3
SKIPTHOUGHT [23]	77.3	81.8	92.6	87.9
SKIPTHOUGHT(LN)	79.5	83.1	93.7	89.3
SDAE [12]	74.6	78.0	90.8	86.9
CNN [21]	81.5	85.0	93.4	89.6
ADASENT [56]	83.1	86.3	95.5	93.3
BYTE MLSTM	86.9	91.4	94.6	88.5

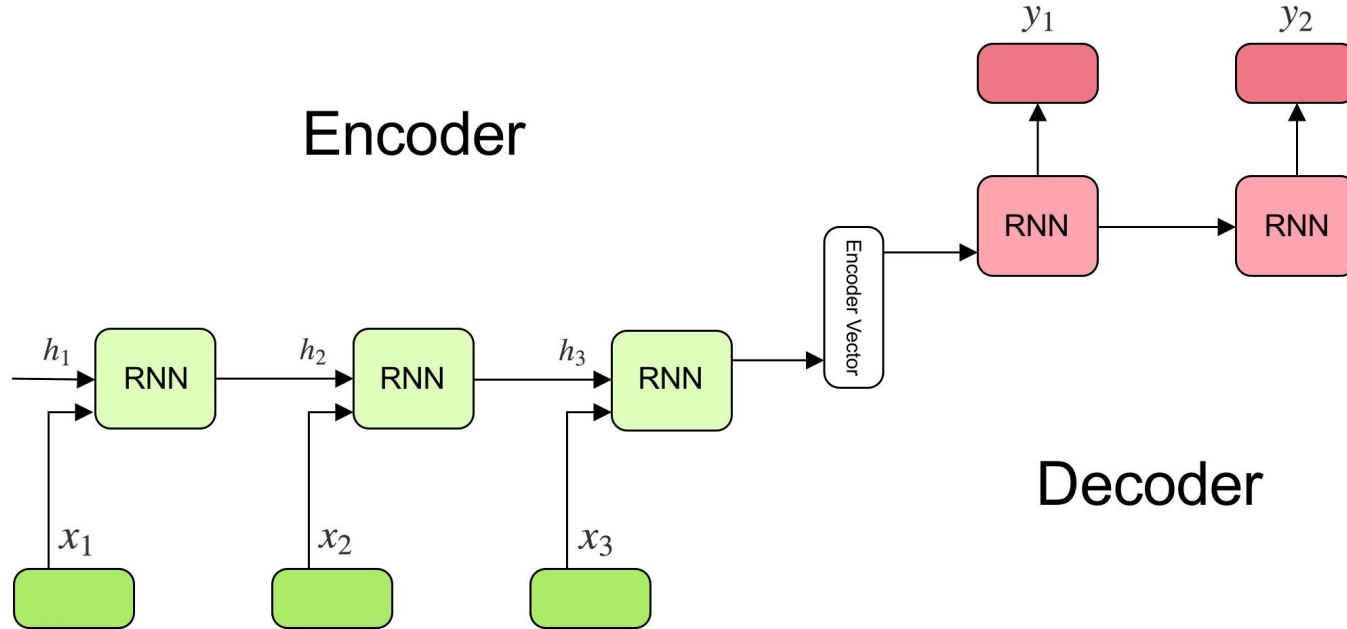
<https://cloud.google.com/natural-language/docs/sentiment-tutorial>

<http://www.nltk.org/howto/sentiment.html>

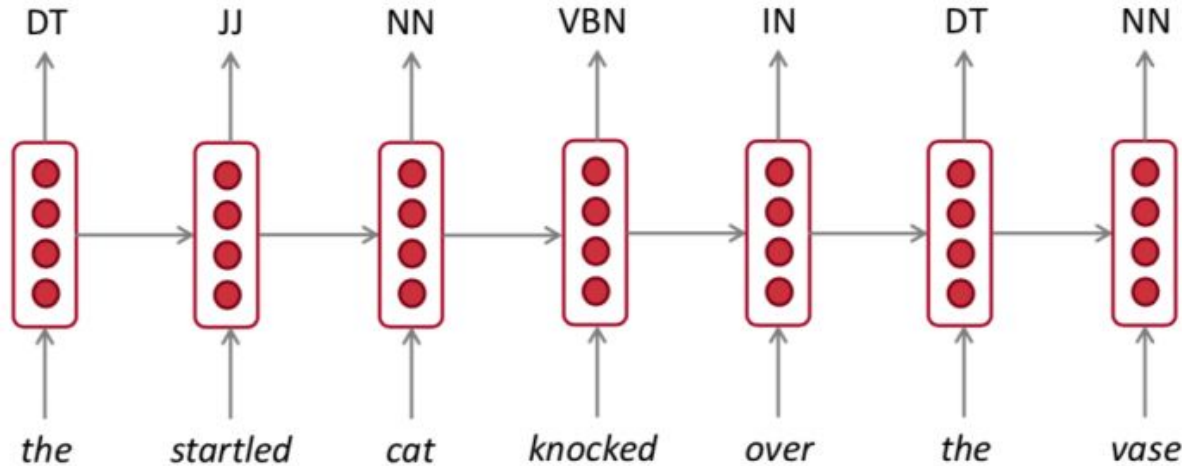


Sequence to Sequence Learning

Sequence-to-Sequence (Seq2Seq) Learning



Application: Sequence Tagging



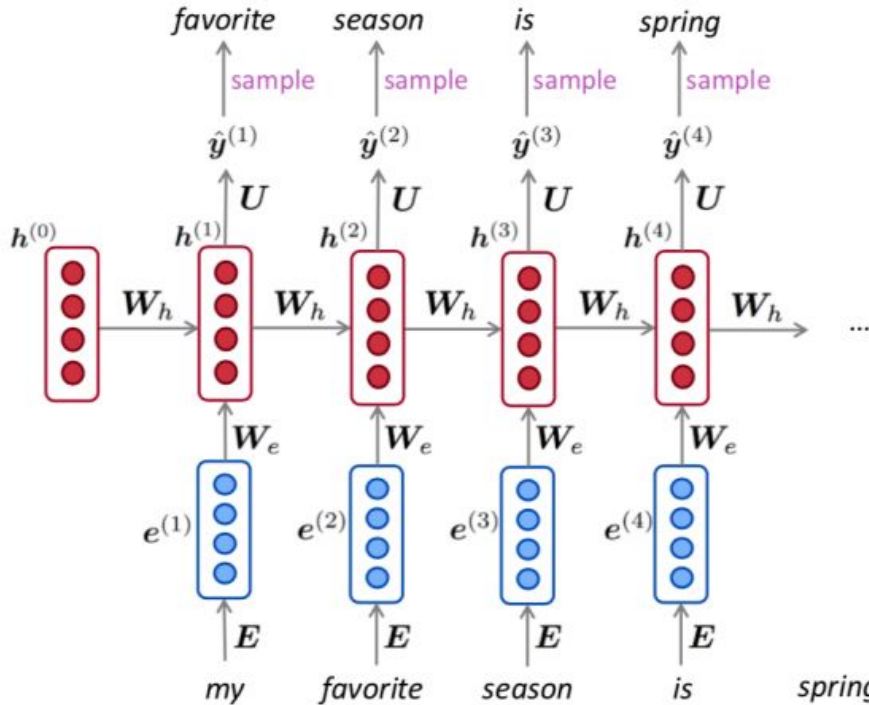
Input: A sequence of n words:

Output: tags

$$P(y_i = k) = \text{softmax}_k(\mathbf{W}_o \mathbf{h}_i) \quad \mathbf{W}_o \in \mathbb{R}^{C \times d}$$

$$L = -\frac{1}{n} \sum_{i=1}^n \log P(y_i = k)$$

Application: Text Generation



Generating text by repeatedly sampling the next step's output by the last output.

Application: Text Generation

Good afternoon. God bless you.

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretcks of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.

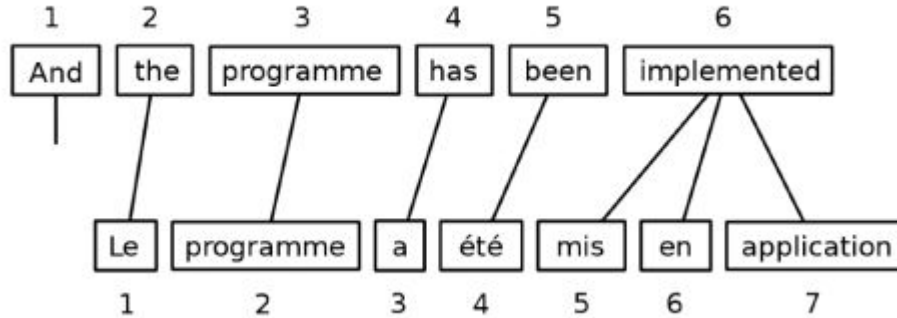
```
\begin{proof}
We may assume that  $\mathcal{I}$  is an abelian sheaf on  $\mathcal{C}$ .
\item Given a morphism  $\Delta : \mathcal{F} \rightarrow \mathcal{I}$ 
is an injective and let  $\mathfrak{q}$  be an abelian sheaf on  $\mathcal{X}$ .
Let  $\mathcal{F}$  be a fibered complex. Let  $\mathcal{F}$  be a category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\label{lemma-characterize-quasi-finite}
Let  $\mathcal{F}$  be an abelian quasi-coherent sheaf on  $\mathcal{C}$ .
Let  $\mathcal{F}$  be a coherent  $\mathcal{O}_X$ -module. Then
 $\mathcal{F}$  is an abelian catenary over  $\mathcal{C}$ .
\item The following are equivalent
\begin{enumerate}
\item  $\mathcal{F}$  is an  $\mathcal{O}_X$ -module.
\end{enumerate}
\end{lemma}
```

Obama speeches

Latex generation

Andrej Karpathy “The Unreasonable Effectiveness of Recurrent Neural Networks”

Machine Translation



Classical statistical
machine translation:
Word-based,
Phrase-based

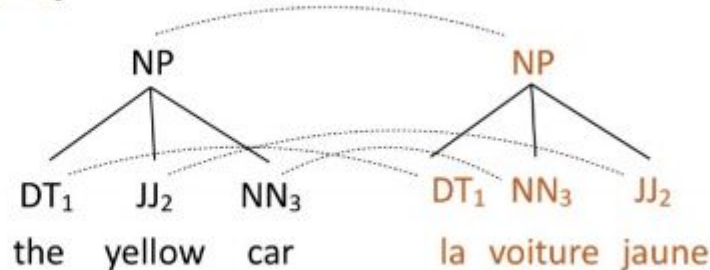
NP → [DT₁ JJ₂ NN₃; DT₁ NN₃ JJ₂]

DT → [the, la]

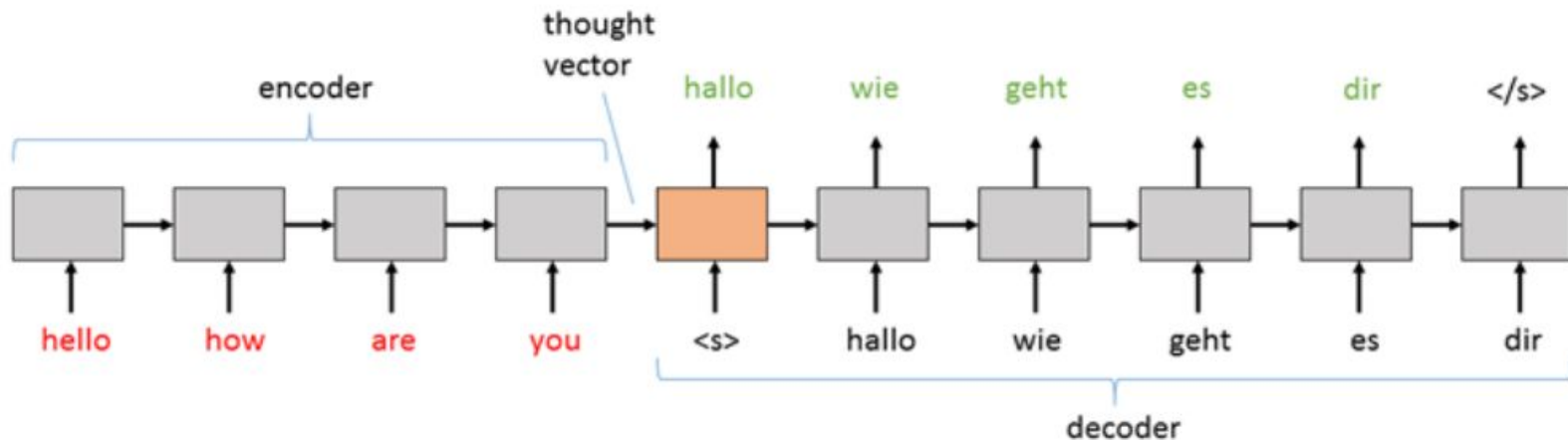
DT → [the, le]

NN → [car, voiture]

JJ → [yellow, jaune]



Neural Machine Translation



The encoder transform the input sequence into a vector.

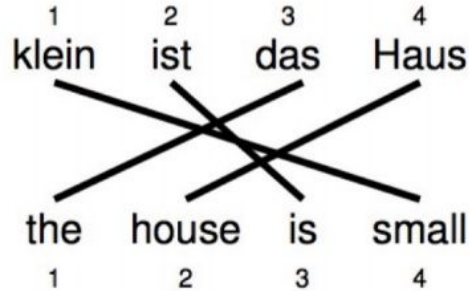
Decoder samples from the vector word by word.

Beam search for better inference.

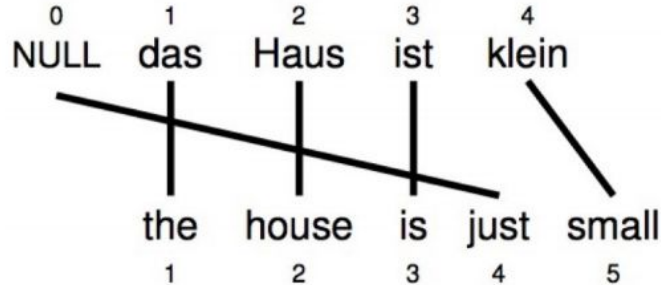
Learning is prone to vanishing/exploding gradients.

Attention Models

Motivation: In statistical machine translation, the algorithm needs to calculate the “alignment” between source and target sequence, i.e. which word corresponds to each word in the source:



$$\mathbf{a} = (3, 4, 2, 1)^\top$$



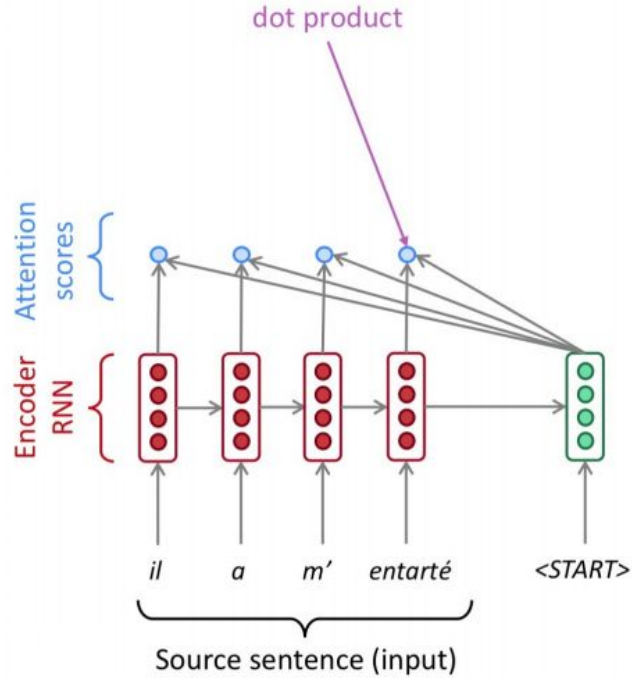
$$\mathbf{a} = (1, 2, 3, 0, 4)^\top$$

Attention Models

The attention model is a neural network equivalency of the alignment in the context of machine translation.

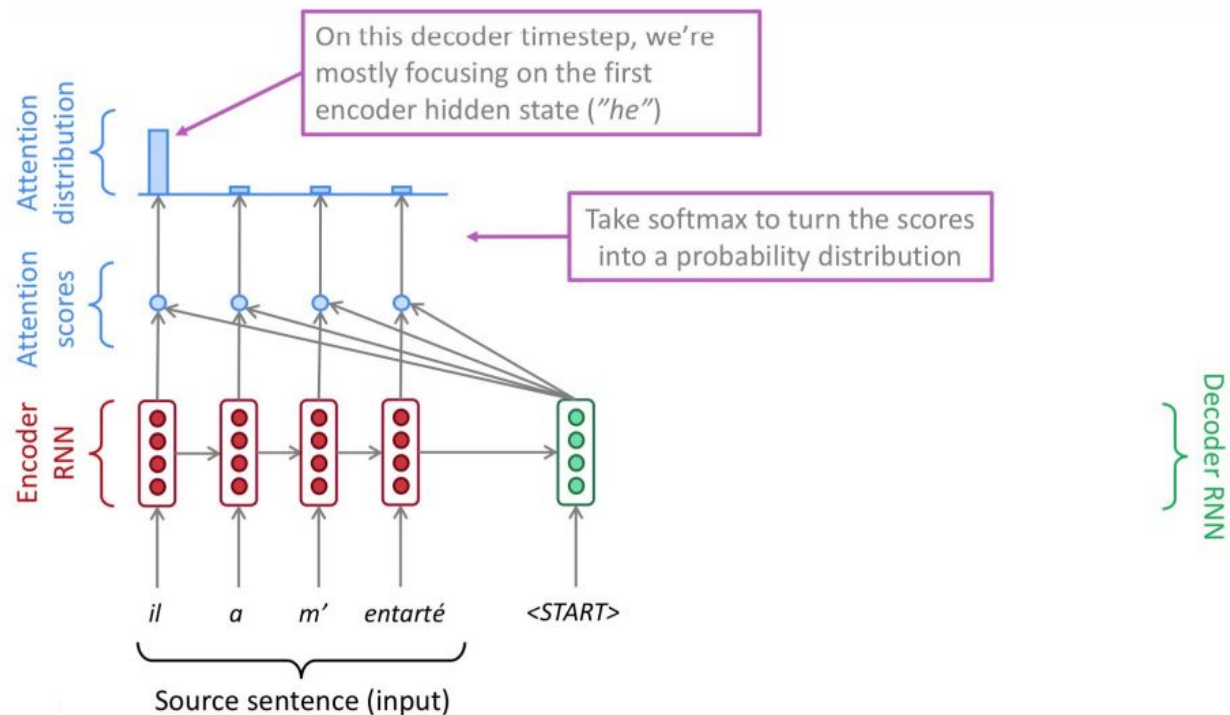
Essence: while doing decoding, the network focus on a small part of the source sentence instead of the whole sentence.

Seq2Seq with Attention

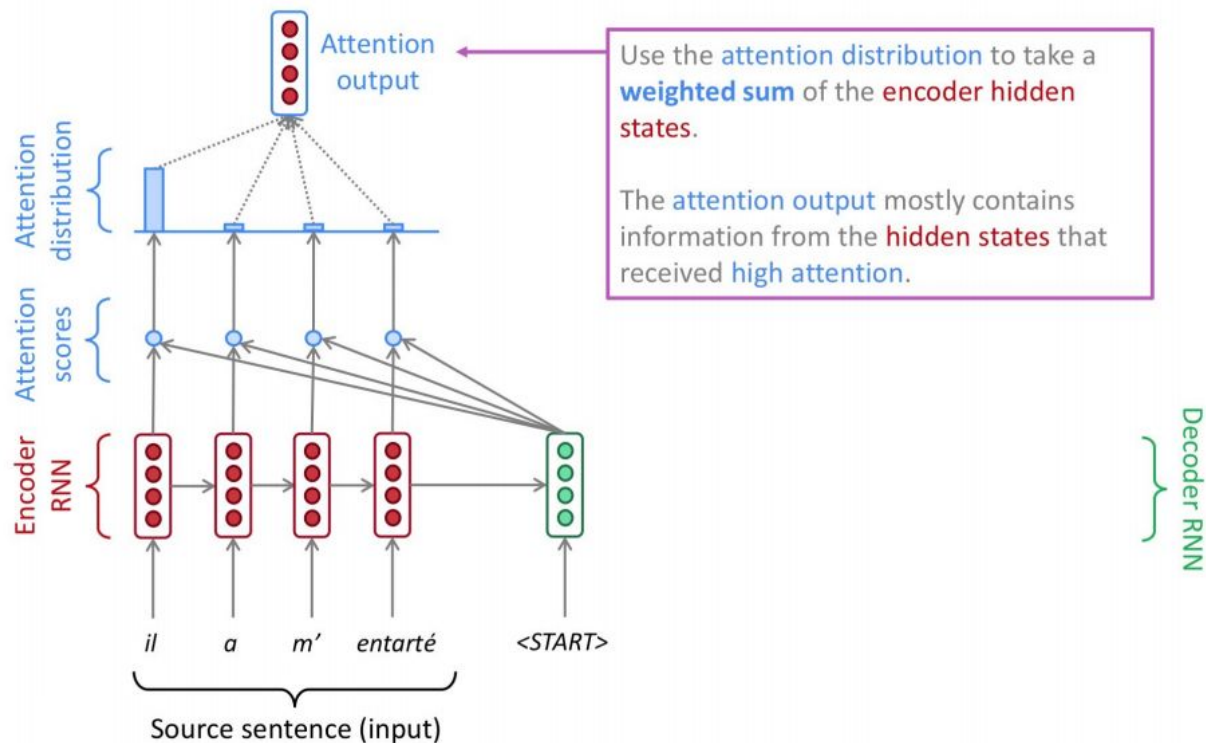


Decoder RNN

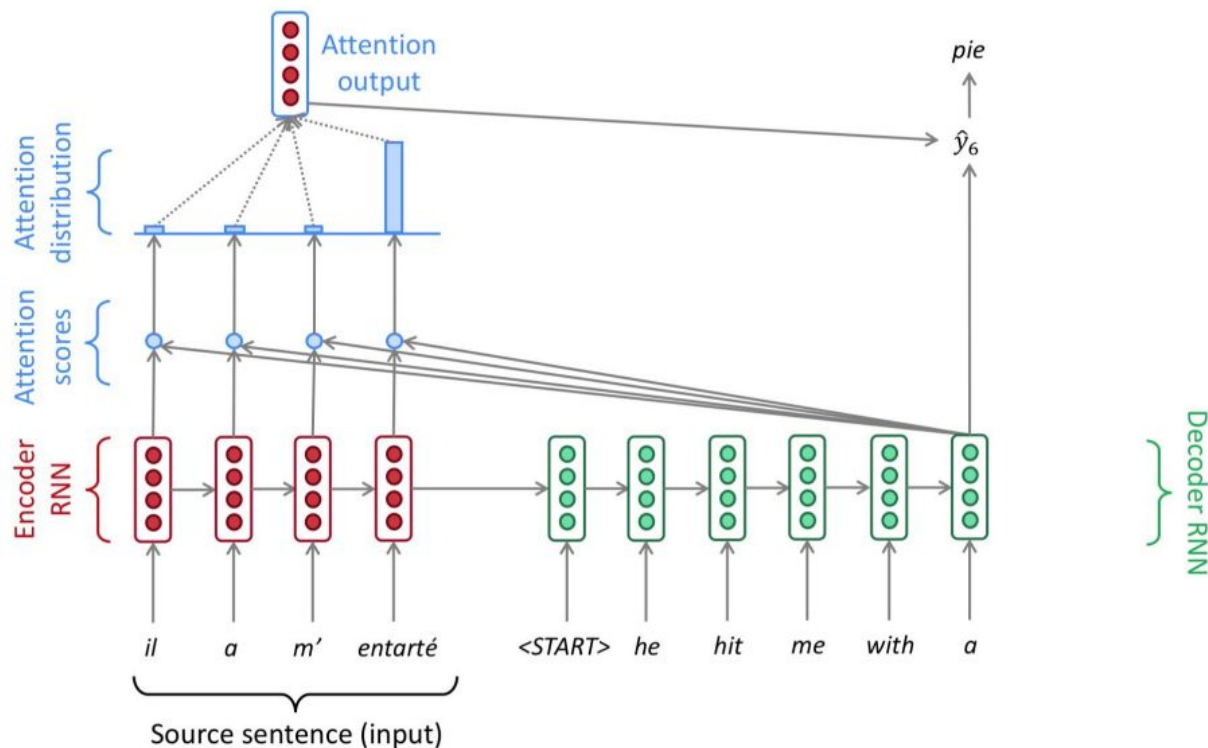
Seq2Seq with Attention



Seq2Seq with Attention



Seq2Seq with Attention



PolIEV

What we learned

- Big ideas
 - Defining similarity: From word vectors to meaning vectors
 - Handling Variable Length Vectors
 - Syntactical Analysis
- Semantic Analysis
- Sequence to Sequence