Introduction to Deep Learning in Natural Language Processing

Dr. Ahmad El Sallab

Agenda

- What is NLP?
- Why NLP? Applications
- What are NLP tasks?
- Why NLP is hard?
- DL in NLP
- BoW model
- Text preprocessing pipeline
- Text features

What is NLP?

Natural Language => Means/**Media** of communication between Humans:

- Verbal → Speech
- Textual
- Sometimes: visual (Visual QA, **OCR**)

What is NLP?

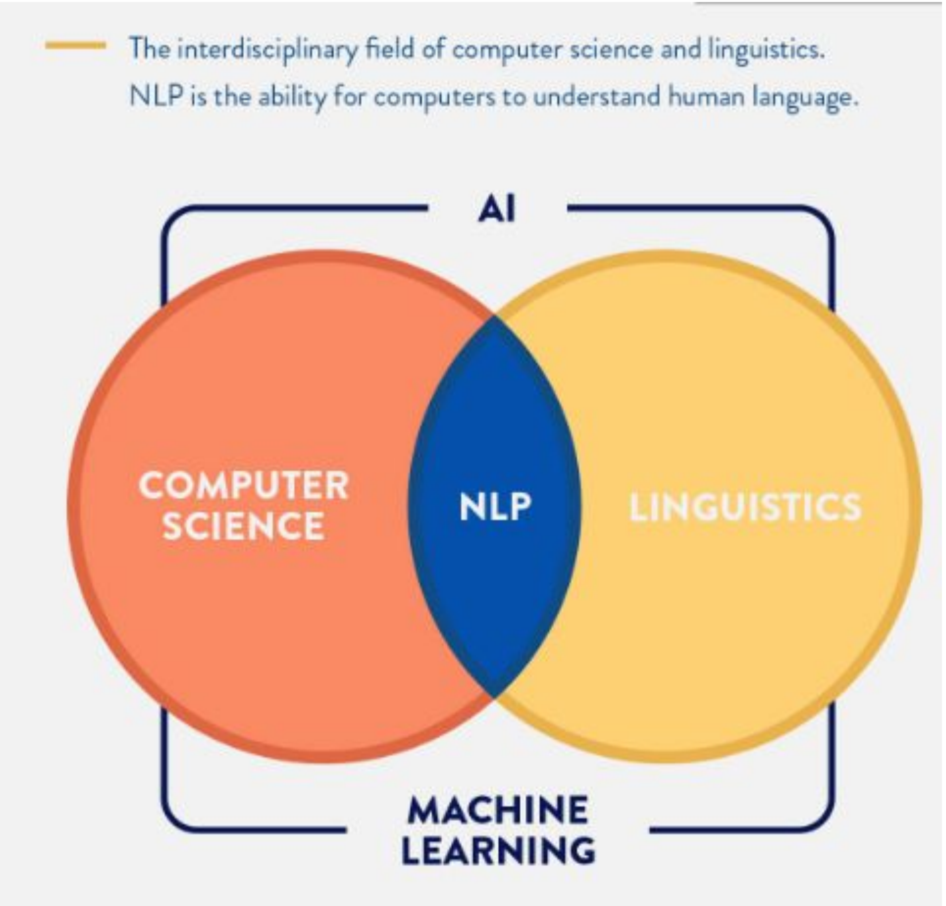
How to make language understandable to computers?

Then develop algorithms to operate on the input language:

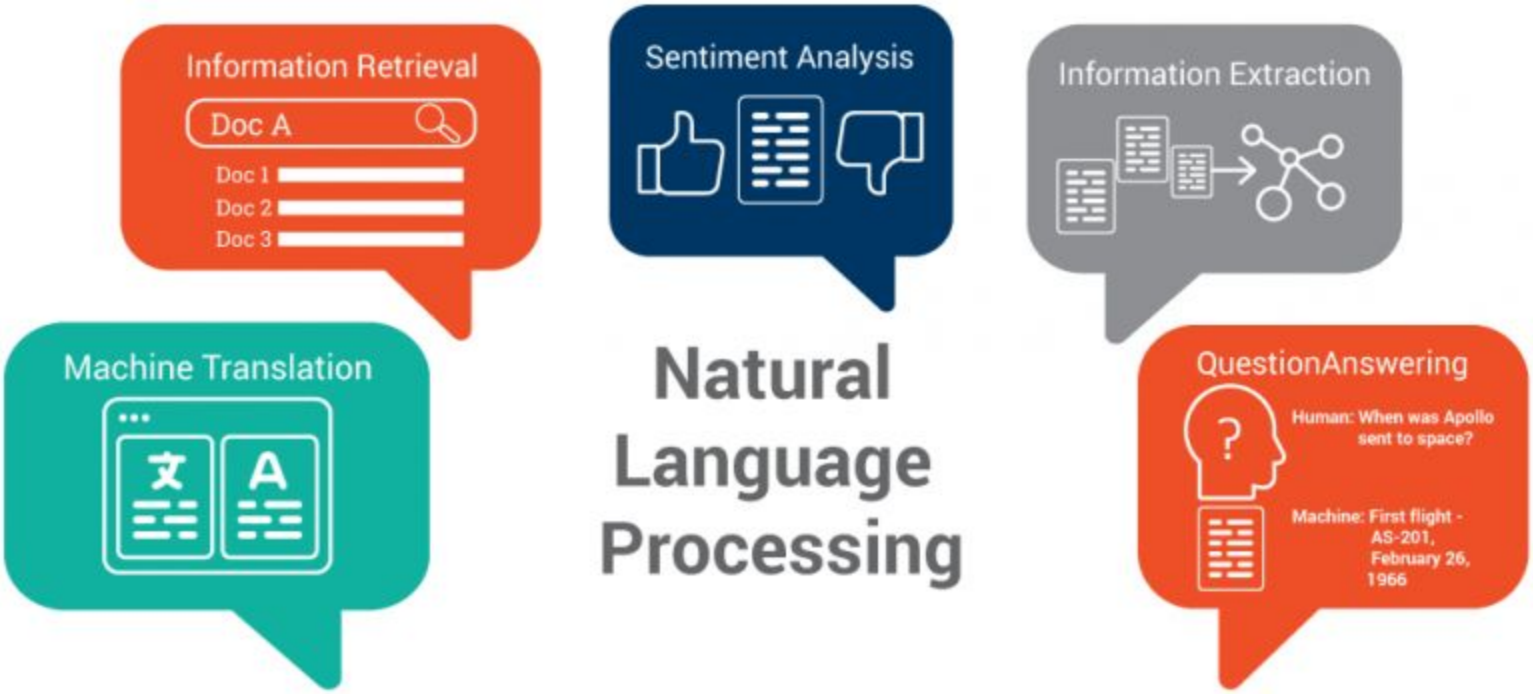
1- CS

2- AI

3- Linguistics

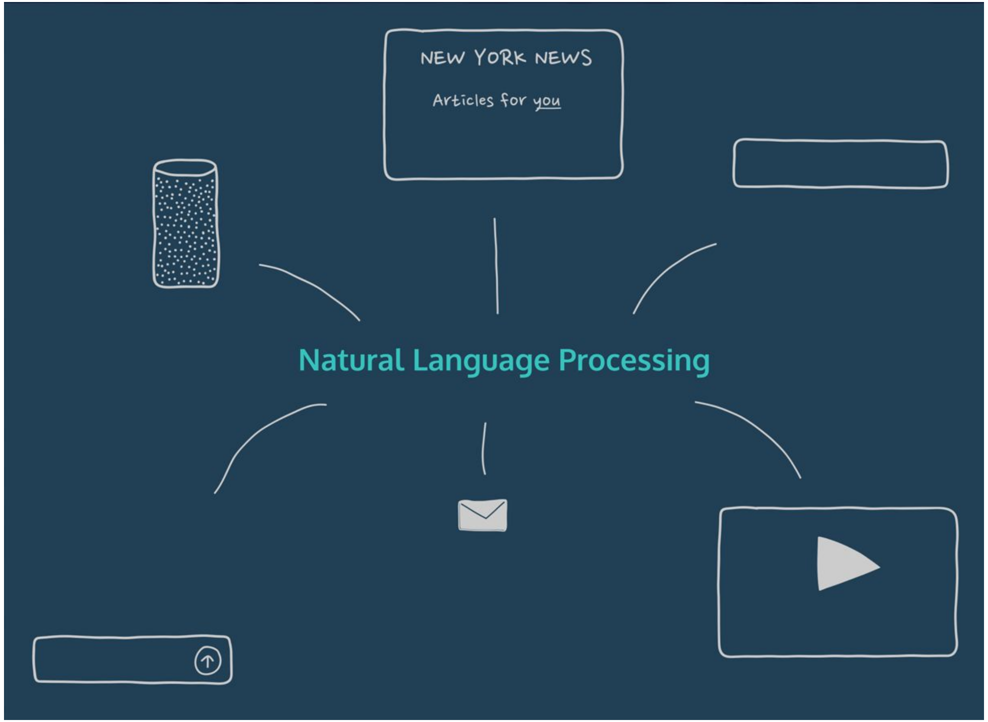


Why NLP?


Information Retrieval
Doc A
Doc 1
Doc 2
Doc 3
Sentiment Analysis
Information Extraction
Machine Translation
文
A
Natural Language Processing
QuestionAnswering
?
Human: When was Apollo sent to space?
Machine: First flight - AS-201, February 26, 1966

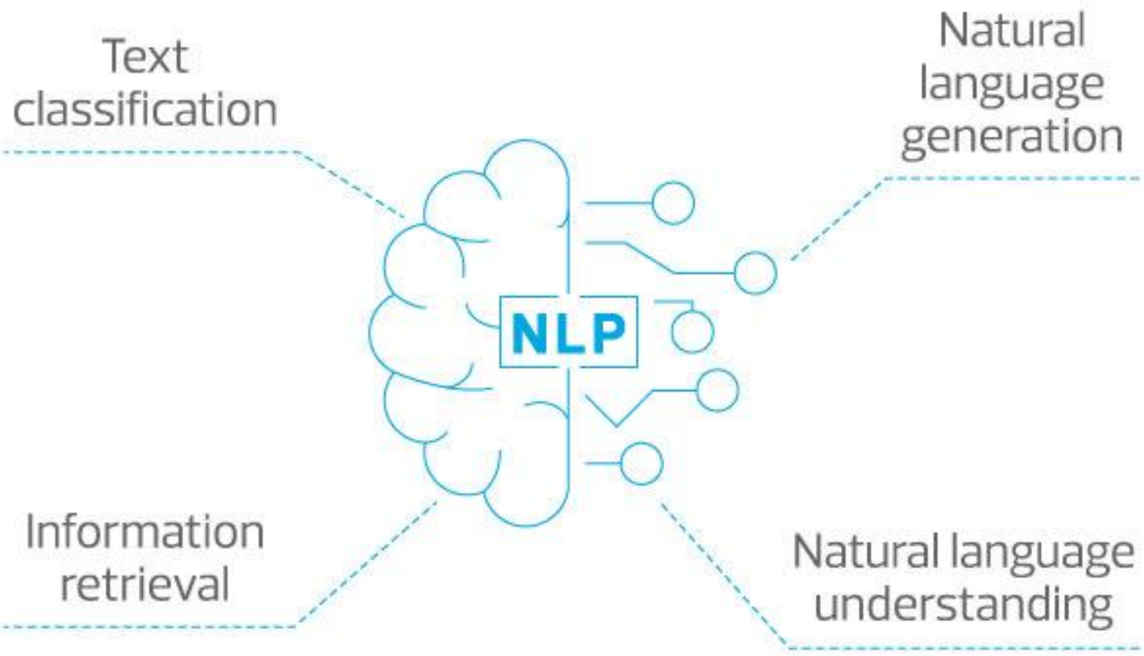
Why NLP?

Human-Machine Interactions



What are NLP tasks?

Sentiment classification
Toxic comments



QA
Chatbot
MT
Autocompletion
Summarization

Information extraction: Docs parsing → structured DB
Recommender systems

Intent classification

Why NLP is hard?

Human language is difficult

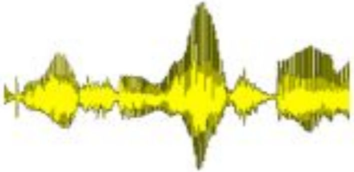
- Symbolic
- Implicit (Sarcasm!)
- Different Encodings for the same symbol.meaning:
    - Could include other signals (gestures, emoticons, speech)
- Sparse: large vocab
- Diverse: many languages, dialects, accents,..etc

What's special about human language?

The categorical symbols of a language can be encoded as a signal for communication in several ways:

- Sound
- Gesture
- Writing/Images

The symbol is invariant across different encodings!



CC BY 2.0 David Fulmer 2008

National Library of NZ, no known restrictions

What's special about human language?

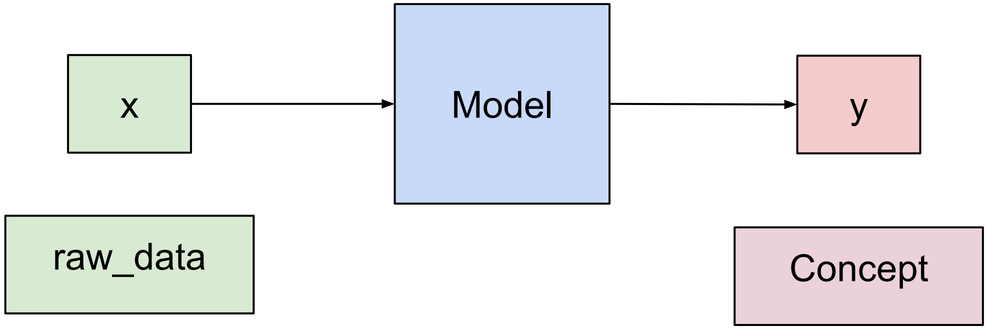
A human language is a system **specifically constructed to convey the speaker/writer's meaning**

- Not just an environmental signal, it's a deliberate communication
- Using an encoding which little kids can quickly learn (**amazingly!**) and which changes

A human language is mostly a **discrete/symbolic/categorical signaling system**

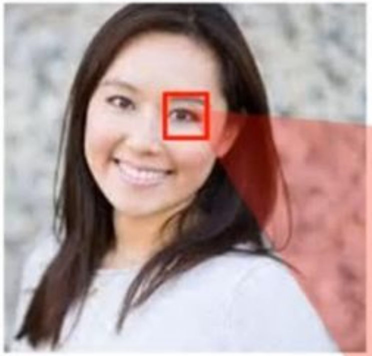
- rocket = 🚀; violin = 🎻
- Presumably because of greater signaling reliability
- Symbols are not just an invention of logic / classical AI!

Semantic gap in ML



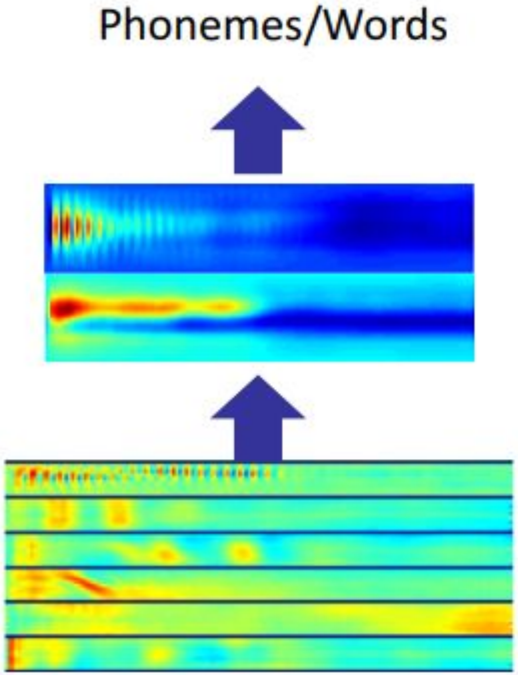
Semantic gap in ML

CV: what the computer can see?



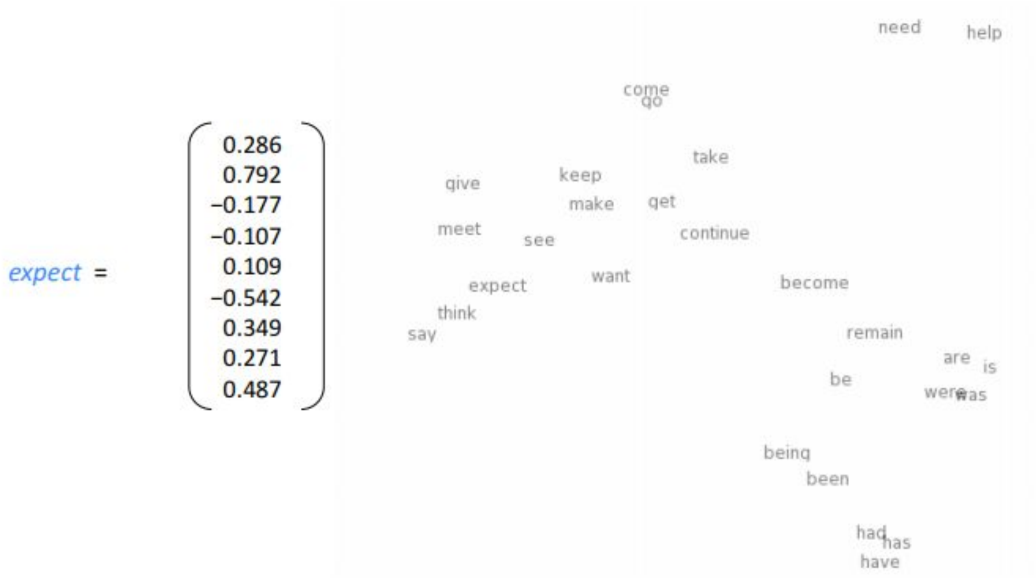
30	32	22	12	10	10	12	33	35	30
12	11	12	234	170	176	13	15	12	12
234	222	220	230	200	222	230	234	56	78
190	220	186	112	110	110	112	180	30	32
49	250	250	250	4	2	254	200	44	6
55	250	250	250	3	1	250	245	25	3
189	195	199	150	110	110	182	190	199	55
200	202	218	222	203	200	200	208	215	222
219	215	220	220	222	214	215	210	220	220
220	220	220	220	221	220	221	220	220	222

Semantic gap in Speech



Semantic gap in Text

How to even transform/digitize a sequence of words into numbers?

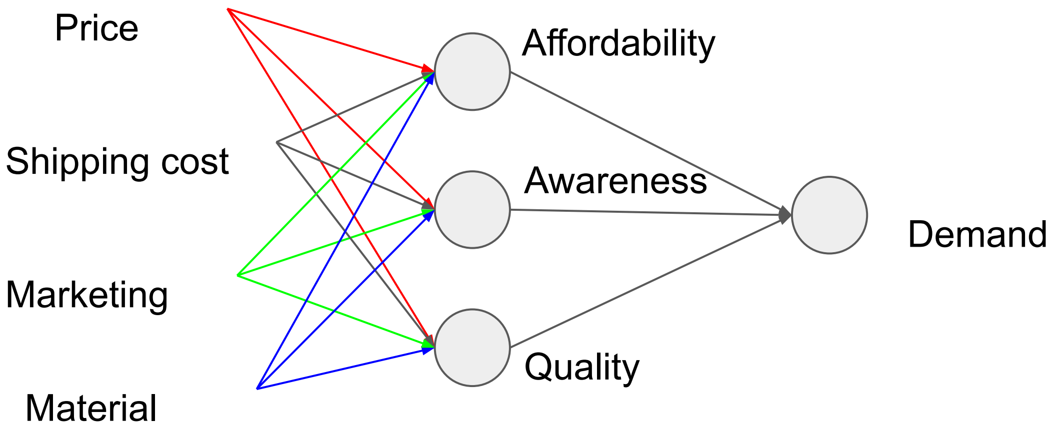


Let’s code

How computer reads?

https://colab.research.google.com/drive/1IwhBkAvgBG0QiBwGPJb2FbOXtz_sQyBW

DL way of thinking → Hierarchical refinement of concepts

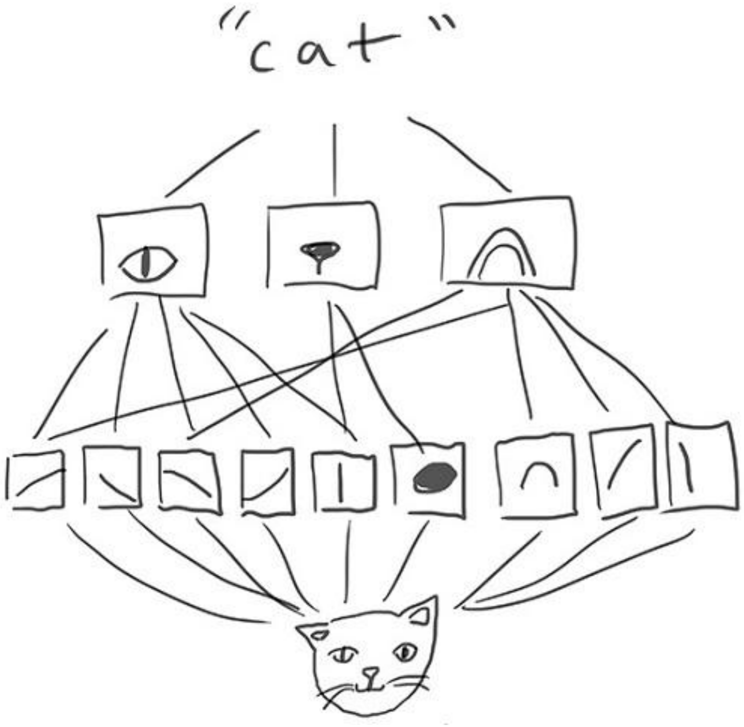


Price	Shipping cost	Marketing	Material	demand

Price	Shipping cost	Marketin g	Material	Affordab ility	Awaren ess	Quality	demand

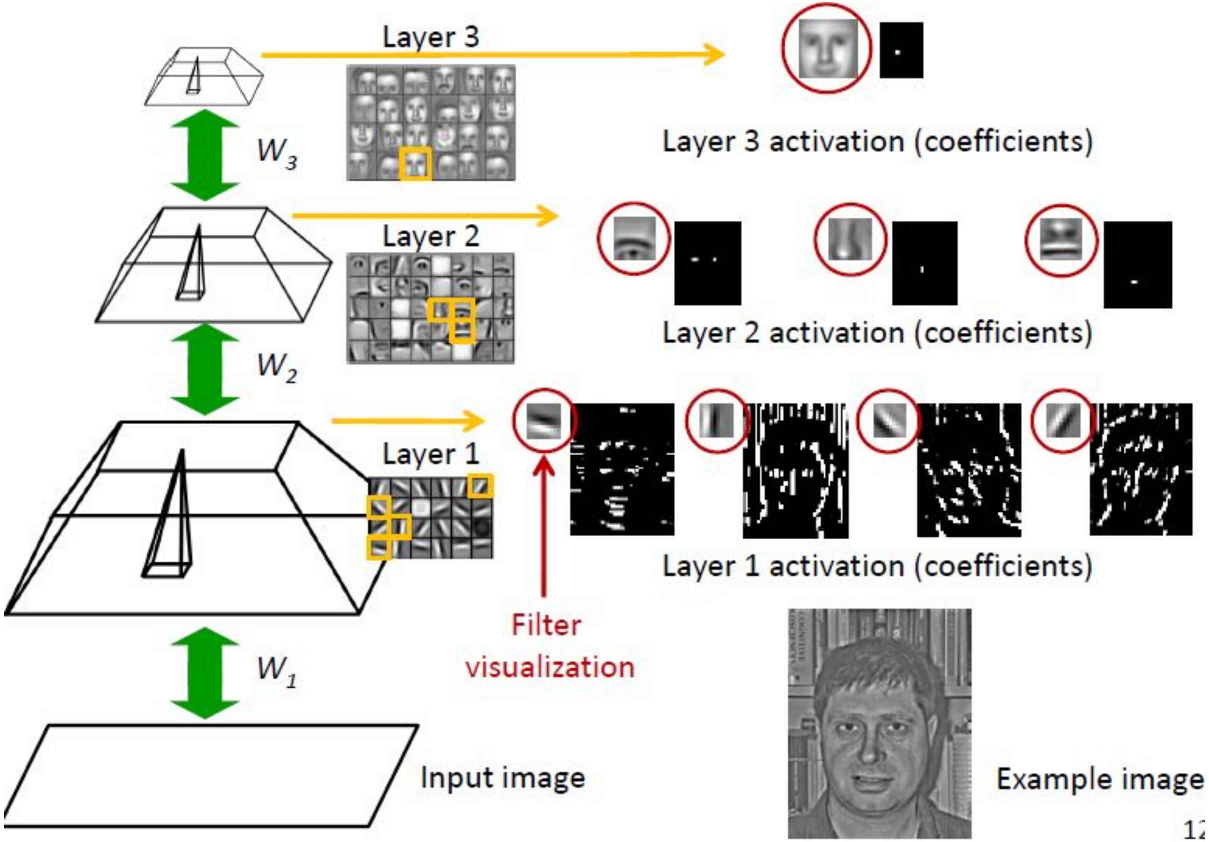
New features!

DL way of thinking → Hierarchical refinement of concepts



https://dpzbhybb2pdcj.cloudfront.net/chollet/v-6/Figures/visual_hierarchy_rsd.jpg

CV is spatial

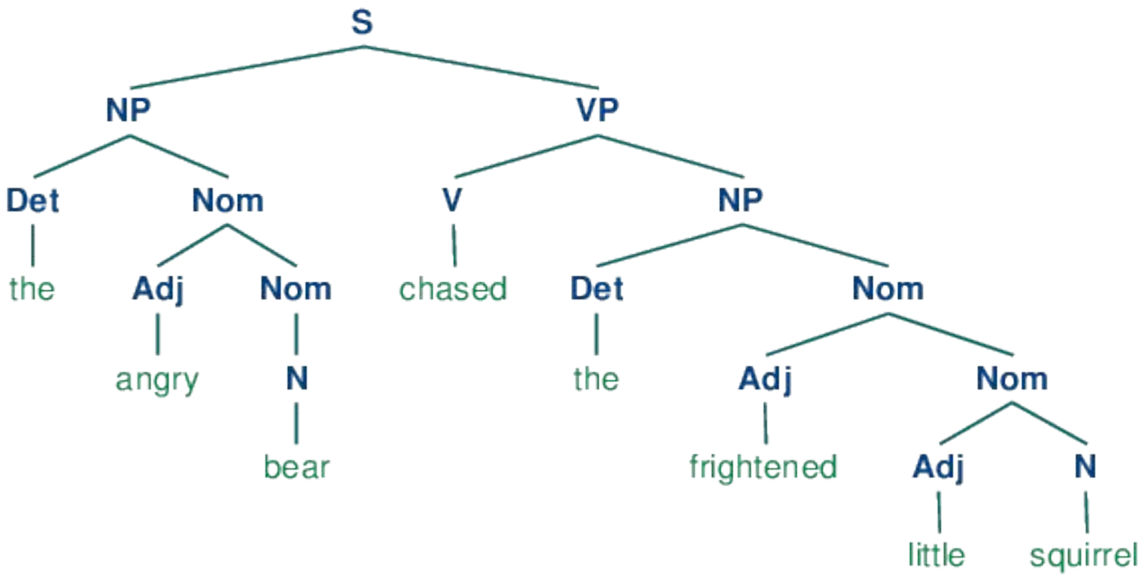


NLP is sequential

the clouds are in the

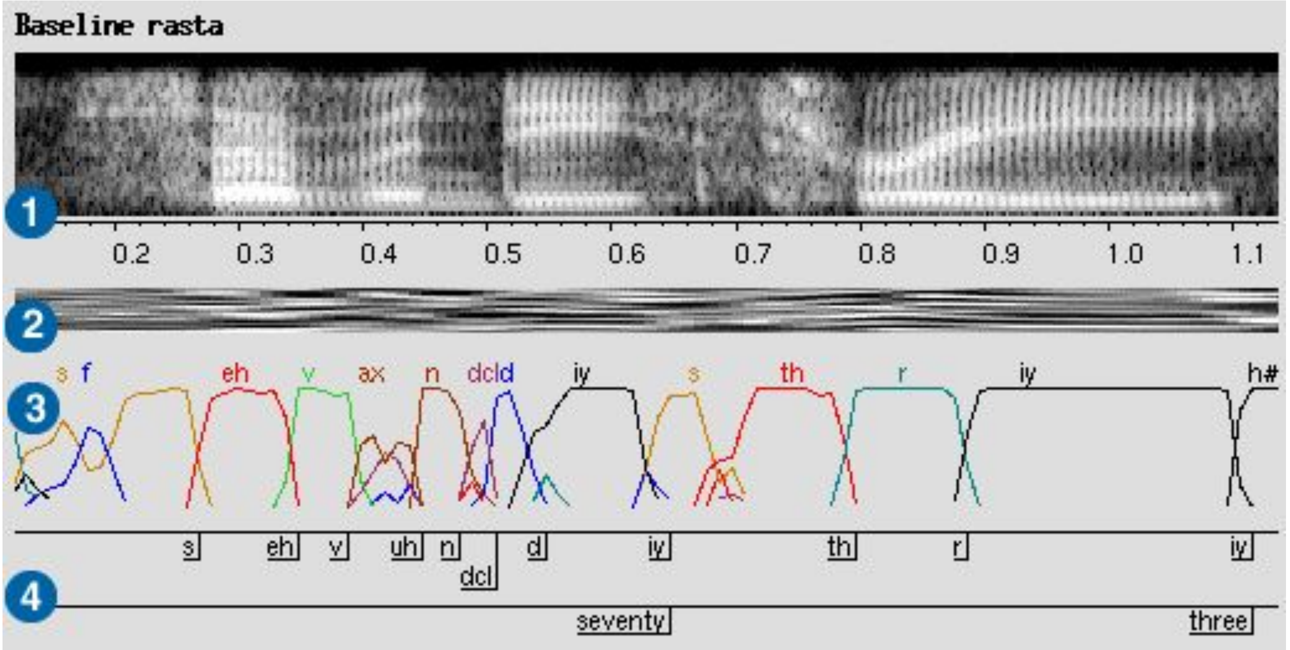
Ground
Garage
Sky
Airport
Oven

NLP is hierarchical



https://www.nltk.org/book/tree_images/ch08-tree-6.png

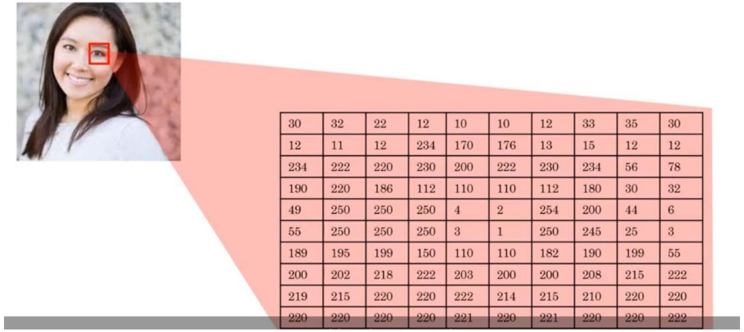
NLP is sequential



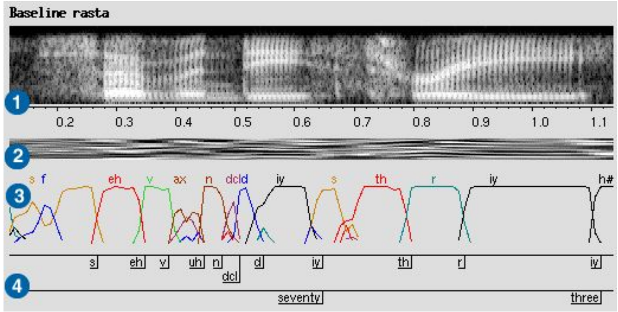
CV is continuous -- NLP is categorical

CV Alphabet are pixels values (0..255)
NLP Alphabet/Vocabulary are symbols (indices in vocab)

- CV → pixels → meaning=intensity
  - Already numerical and continuous
  - Near values means near intensity
  - No further processing needed
  - What remains is how to understand the **spatial relations/features**?



- NLP → words/phonemes → symbols=?
  - How to encode/digitize the symbols for the computer?
  - Encoding must incorporate similarity operations!
  - *Similar words should have similar representations*
  - What remains is how to understand the **sequential relations/features**?



Analogy CV vs NLP - conclusion

Raw data:

CV is numeric - NLP is symbolic (categorical)

Context:

CV is spatial - NLP is sequential

NLP pipeline

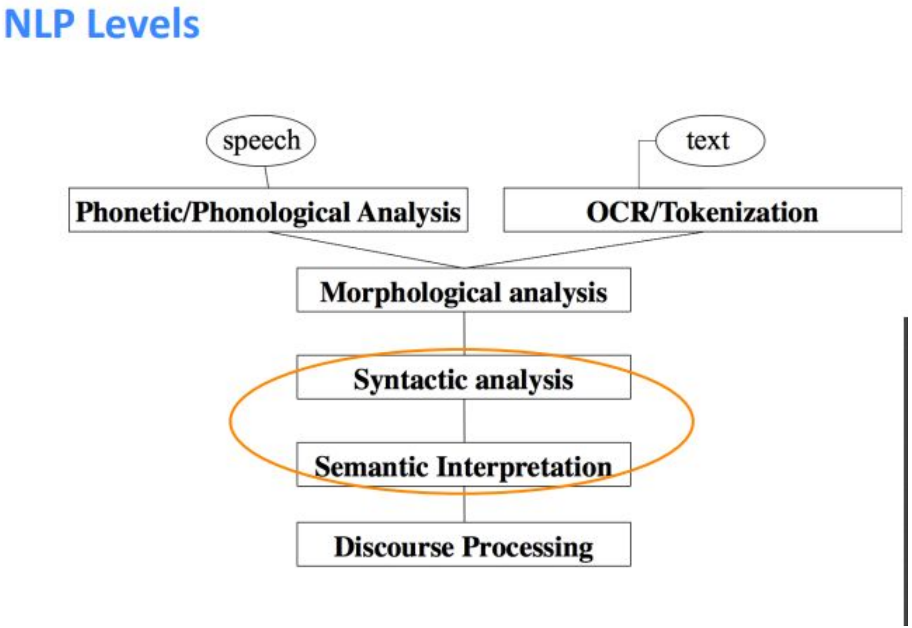
Low level tasks → Medium dependent (speech, text,..etc)

Morphology: Normalize symbols variations (synonyms, phonemes,...etc)

Syntax: Parse symbols according to some template → “grammar”

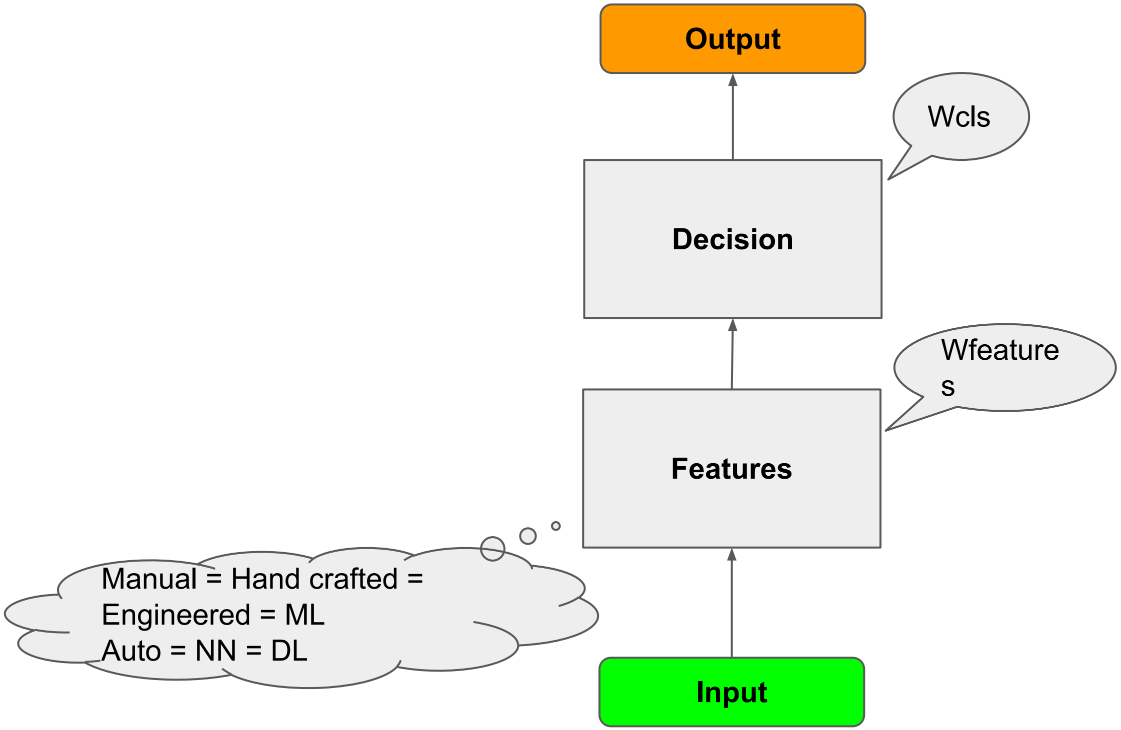
Semantics: Convert symbols to meaningful structures

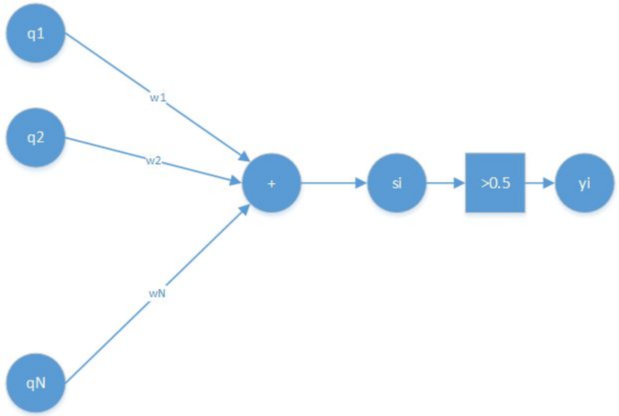
Processing: Perform the task!



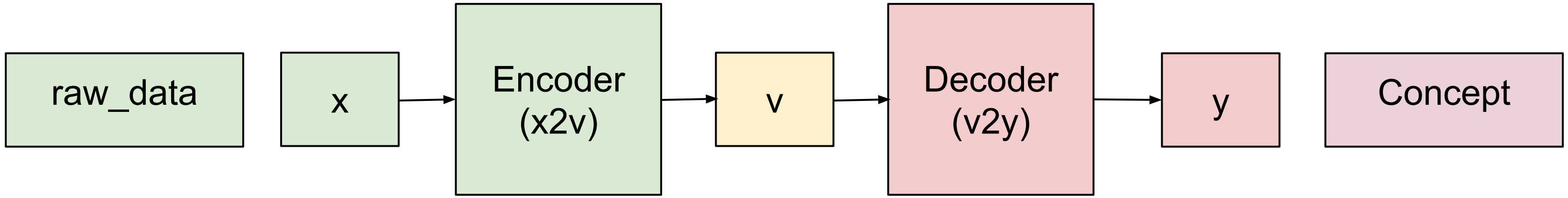
DL in NLP

Supervised Learning Model Design Pattern

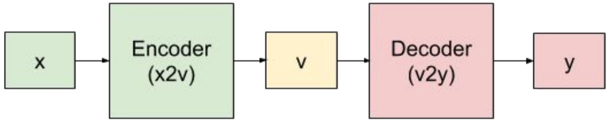




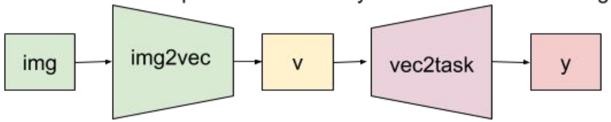
Encoder-Decoder to close the semantic gap in ML



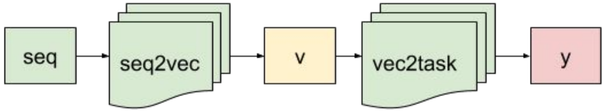
Encoder-Decoder pattern in NLP



(a) Encoder- Decoder Pattern in Deep Learning to close the semantic gap = x2y
The Encoder produces an “Embedding” vector “v” that encoder x → x2v.
The Decoder produces the task y based on the embedding → v2y.

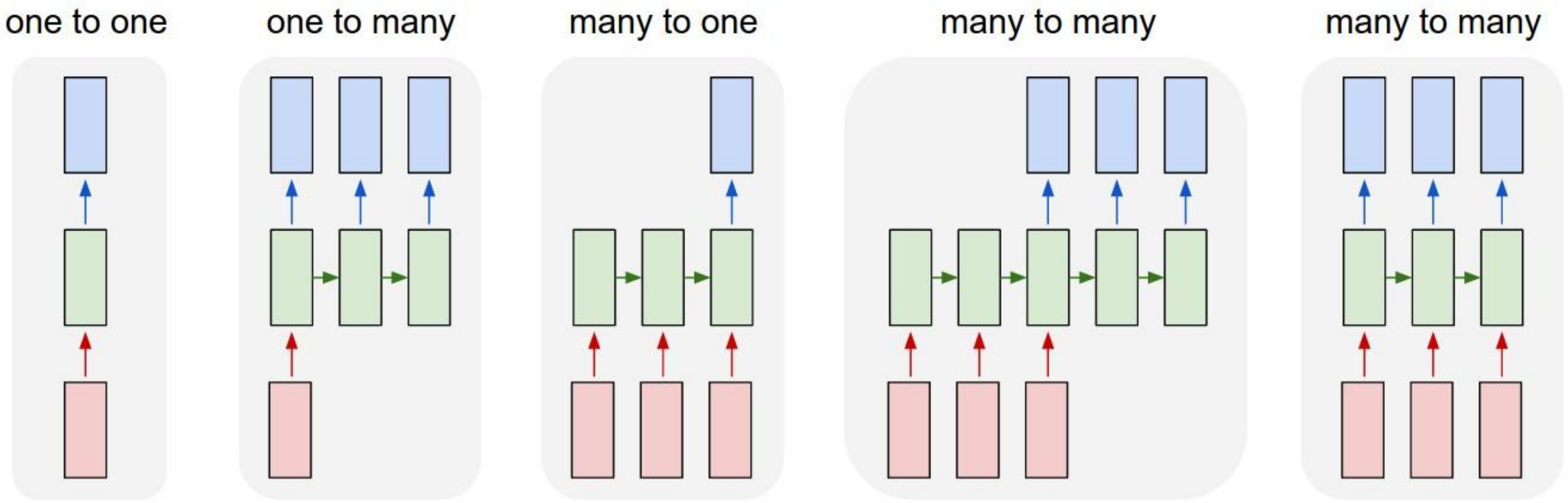


(b) Spatial image case
The Encoder = ConvNet.
Embedding v is a spatial feature map that encodes spatial features in a grid representing regions of input image.
The Decoder depends on the task y (img2cls, img2img, img2box)



(c) Sequential language case
The Encoder = Sequence model (RNN, Transformer,..etc)
Embedding v is a summarization of the sequence.
The Decoder depends on the task y (seq2cls, seq2seq)

Encoder-Decoder pattern



Examples

- One-One: Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification).

- One-many: Sequence output (e.g. image captioning takes an image and outputs a sentence of words).

- Many-one: Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

- Many-Many: Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).

- Synced sequence input and output: NER

What is the best representation of language?

This question summarizes all NLP efforts!

For what?

- Classification
  - Many-to-one: Seq2Class
  - Sentiment analysis, Toxicity detection (JIGSAW), Real or not? Disaster tweets
- Dialogue:
  - Many-to-many: Seq2Seq → Unaligned case → More on that later
  - MT, Spelling correction, Speech, OCR,...etc

NLP models meta-architectures

Just like in CV: **Encoder-Decoder**

- **Seq2Class:**
  - Encoder = words vectors aggregation **(How?)**
  - Decoder = None (just classifer=softmax)
  - Analogy to CV: Encoder-Softmax (AlexNet, VGG,...etc)
- **Seq2Seq:**
  - Encoder = words vectors aggregation **(How?)**
  - Decoder = multiple words generation **(How?)**
  - Analogy to CV: Encoder-Decoder in semantic segmentation. But in SS, we have aligned many2many, while in NLP, we have unaligned sequences→ challenge in annotation, model, when to stop, position encoding...etc
- **Word vectors are the input to all the above meta-architectures:**
  - Unlike in CV, where pixels are already digitized
  - Also, in NLP **order matters!** = **Context**

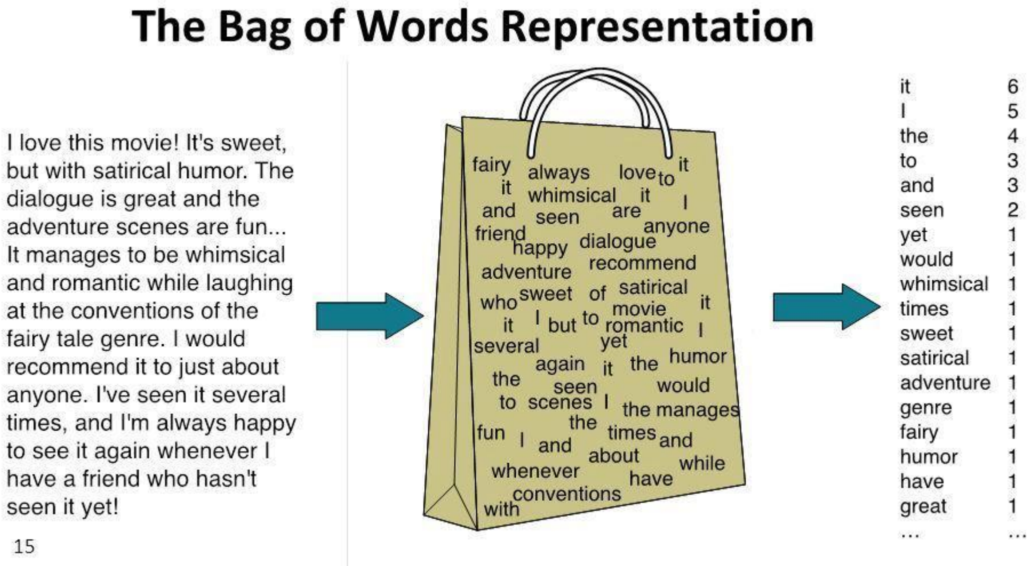
How to represent words to NN?

If we use Dense layers, we need fixed-sized input vectors.

Words = Symbols

The alphabet of symbols s the **vocabulary**

For each piece of text → Put in the bag what is in the text



How to represent words to NN?

Bag-of-Words model

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Let’s code

https://colab.research.google.com/drive/1pzfmjWjTCmnBSR0He7LP2LXyWYTvMhL1?usp=sharing

Text pre-processing pipeline

Two phases

A - Text preprocessing

Text-in→ Text-out

1- **Data sequencing (splitting/tokenization):** each sentence --> sequence (list) of words

2- **Data cleaning:** This step varies from task to task. For some tasks it's better to remove special characters and punctuations, for other they are critical (emotiocons). Good for performance. Sometimes bad! Ex: CAPITALIZED words in sentiment.

3- **Text normalization:** in general text morphology is a big issue in NLP. Upper and lower cases, stemming and lemmatization, ...etc. Again it's task dependent.

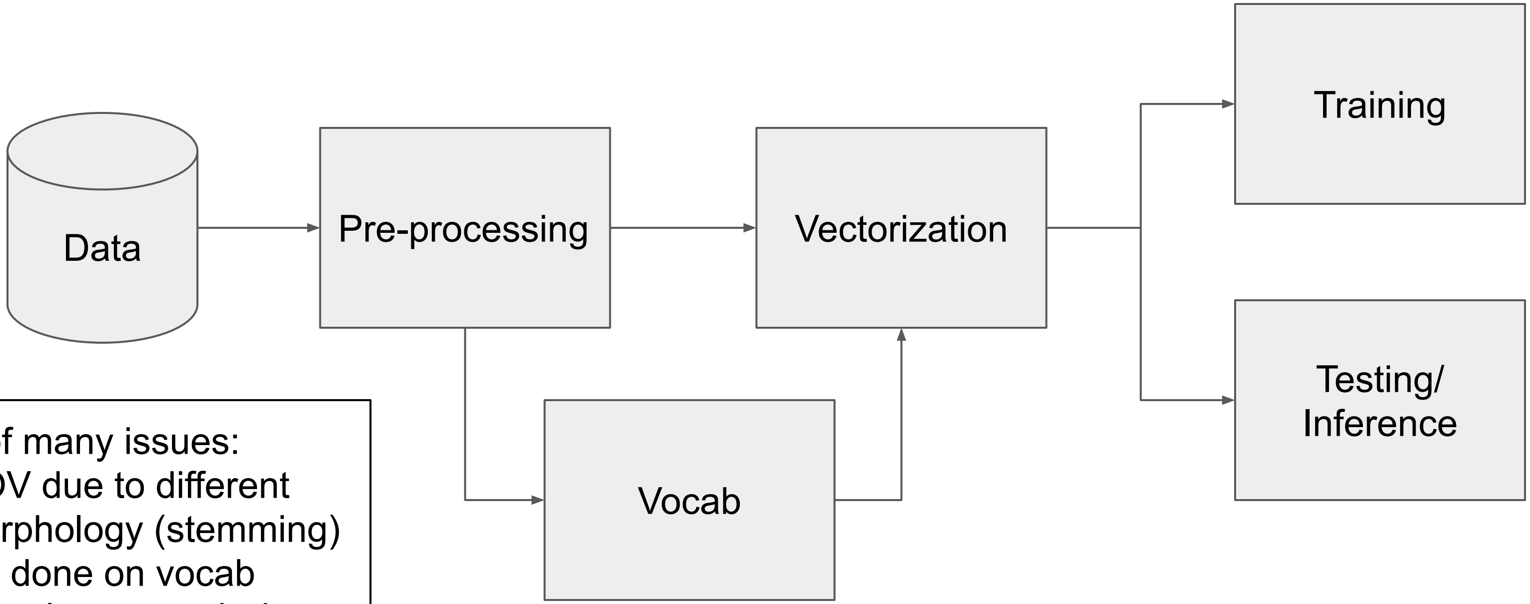
4- **Padding (model dependent):** Dense and CNN. RNN can skip this step.

B- Text preparation

Text-in→ Numbers-out

5- **Binarization/vectorization/digitization:** transform words into numbers according to a vocab index.

Same pre-processing in training and inference



Source of many issues:

- OOV due to different morphology (stemming) not done on vocab
- Extra large vocab due to non-normalization or non-clean text
- OOV test

As a rule of thumb, perform the same preprocessing on the vocab (or its training corpus), as that on the input text

Sequencing and Tokenization

- Simple tokenization → split by space
  - Punctuations, Prefixes, Suffixes
  - Missing spaces by mistake
  - Unnecessarily increase vocab size: Hello, Hello!, Hello,...etc
- Handle with regex
- Use special tokenizer
  - NLTK
  - Keras
  - spaCy

```
'being',
'absurd.
',
 '<br',
'/><br',
'/>Yet,'
,
 'one',
 'must',
'admit',
```

Normalization

- Case-normalization
    - Less vocab → Hello, hello, HELLO → hello
    - Hence, reduce OOV
    - Could remove some meaning → WHAT!
- Remove stop words
    - Again reduces vocab
    - But could remove some meaning → with context and co-reference → No need with sequence models and DL

Stemming

Another source of redundancy and highly variable/unexpected morphology are the prefixes and suffixes

The most basic ones trims prefixes and suffixes, known in the language

Note that: stemming might produce **meaningless words** sometimes!

Also, notice how stemming automatically reduce to lower case.

If we are going to preprocess with stemming, we must do the same on the text we use for building our vocab!

As a rule of thumb, perform the same preprocessing on the vocab (or its training corpus), as that on the input text

```
[ ] s = 'The little girl'
    stemmed = [porter.stem(word) for word in s.split()]
    stemmed
```

```
['the', 'littl', 'girl']
```

Lemmatization

Unlike stemming, lemmatization understand the root of the word in the language:

(am, is, are → be)

So not only the morphology is considered, but also the root. This has more importance in languages like Arabic (requires special lemmatizers and stemmers).

Either stem or lemmatize

There's no need to do both.

Actually stemming might make lemmatization not working.

PoS Tags

In NLP, Part-of-Speech refers to the different classes a word can belong to: noun, verb, adjective, ...etc. The different tags/classes are called tagset, and there's no common standard. They usually encode grammar + tense.

The task of PoS tagging resembles the task of semantic segmentation in CV; assign a class for every word.

```
s = 'He is going on a journey fishing on ships'
tagged = nltk.pos_tag(s.split())
tagged
```

```
[('He', 'PRP'),
 ('is', 'VBZ'),
 ('going', 'VBG'),
 ('on', 'IN'),
 ('a', 'DT'),
 ('journey', 'NN'),
 ('fishing', 'NN'),
 ('on', 'IN'),
 ('ships', 'NNS')]
```

Extra clean-up

We might encounter non ascii codes. In this case we need to decode Unicode characters into a normalized form, such as UTF8.

```
import unicodedata
text = 'some text'
unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8', 'ignore')
```

Extra clean-up

We might encounter non ascii codes. In this case we need to decode Unicode characters into a normalized form, such as UTF8.

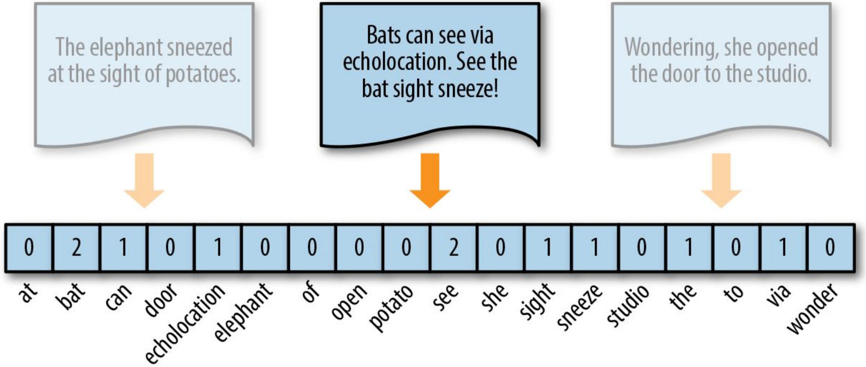
Remove special characters

```
import re
re1 = re.compile(r'  +')
import html

def fixup(x):
    x = x.replace('#39;', "'").replace('amp;', '&').replace('#146;', "'").replace(
        'nbsp;', ' ').replace('#36;', '$').replace('\\n', "\n").replace('quot;', "'").replace(
        '<br />', "\n").replace('\\"', '"').replace('<unk>','u_n').replace(' @.@ ','.').replace(
        ' @-@ ','-').replace('\\', ' \\ ')
    return re1.sub(' ', html.unescape(x))
```

Vectorization

- Build vocab
- Register the index of the word from the vocab
- Not in vocab? → store as UNK
- Need to pad? → encode as PAD
- All special tokens must NOT be used for other symbols
- Example: if PAD=0, don't use 0 as a word index (known mistake)
- Manual or Tokenizer
    - Most tokenizers do both: splitting + vectorization

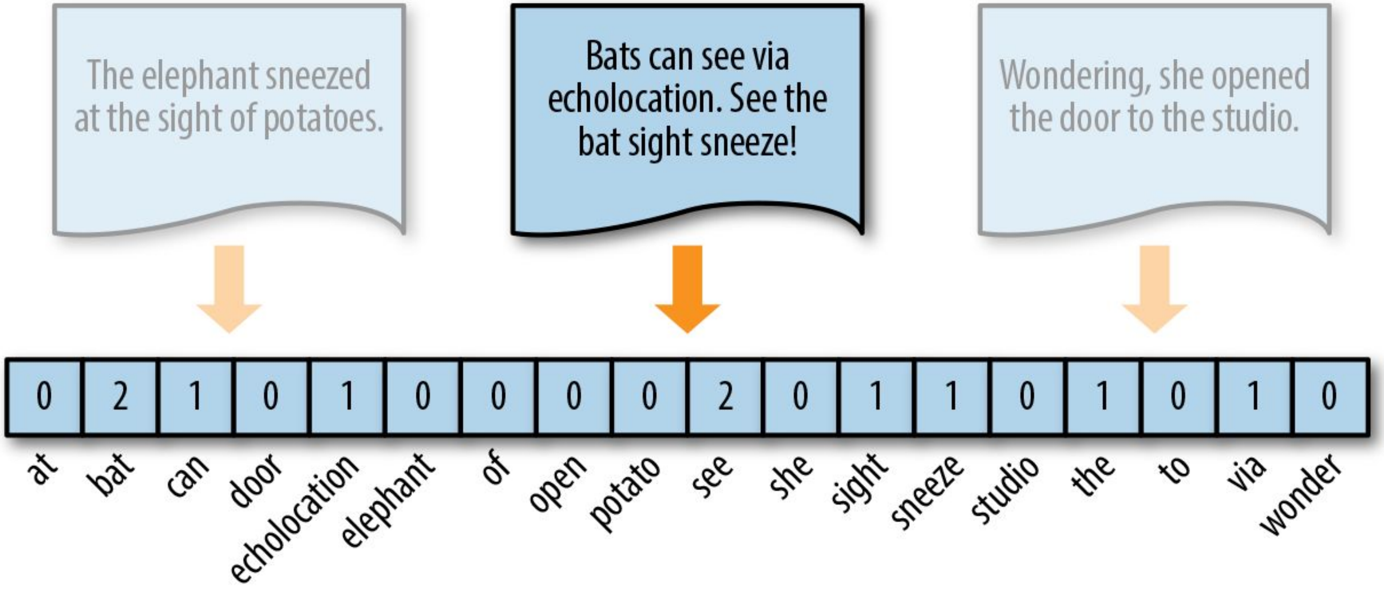


Let’s code

https://colab.research.google.com/drive/1pzfmjWjTCmnBSR0He7LP2LXyWYTvMhL1?usp=sharing

Text features for BoW

Vectorization



Binary features

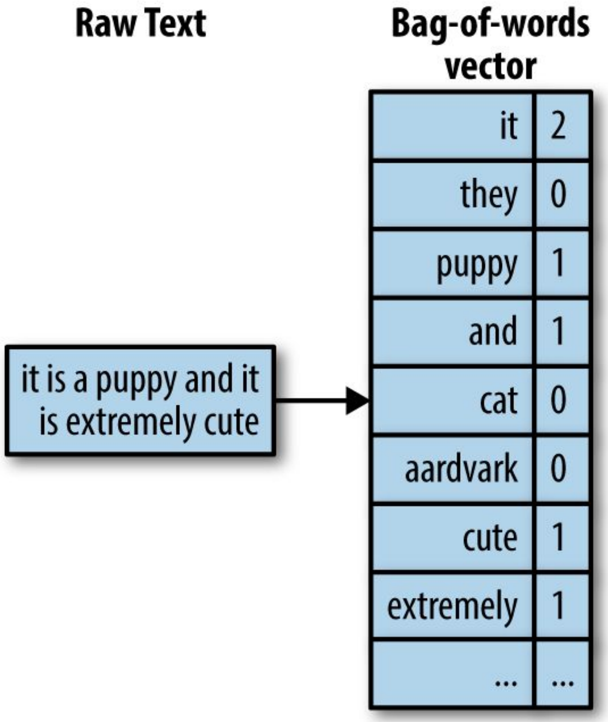
	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

How to encode text data?

First build a vocabulary for your data!

Many ways to encode a review

- BoW: Each sentence will have |V| numbers
  - **Binary:** Put 1/0 if the word is present/absent in a review
  - **Count:** Put the number of times the word is mentioned in a review
  - **Freq:** Normalized counts by total words counts
  - **TF-IDF:** Normalize by the total mentions in all reviews (frequent words are not important) = TF(in rev) x IDF (in ALL revs)
- Sequence (Advanced)



TF-IDF

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Log → df = 0,
log(inf) = 1

N = normalization factor

Bag-of-words (TF)			
loved	movie	great	awful
1	1	3	0
1	1	0	1
0	1	0	0
x			
Weights (IDF)			
loved	movie	great	awful
log(1/.01)	log(1/.33)	log(1/.01)	0
0	log(1/.33)	0	log(1/.01)
0	log(1/.33)	0	
x			
Term-Document Matrix (TF-IDF Weighted)			
loved	movie	great	awful
4.61	1.11	13.82	0
0	1.11	0	4.61
0	1.11	0	0

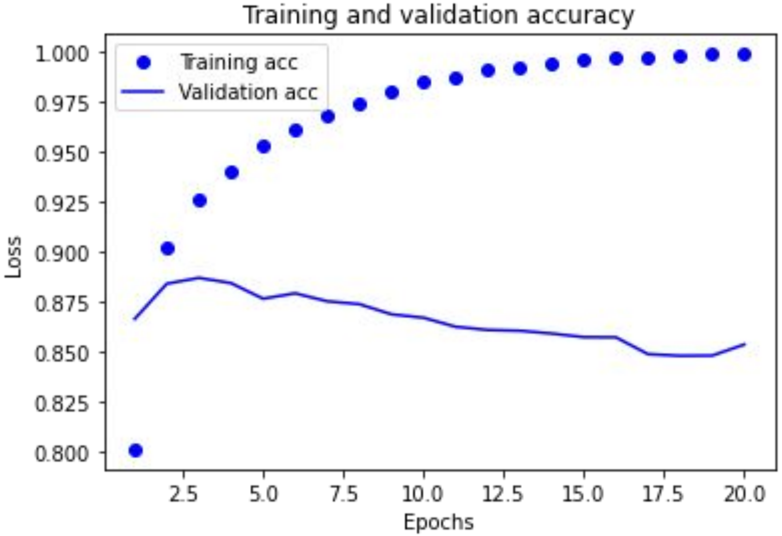
Let’s code

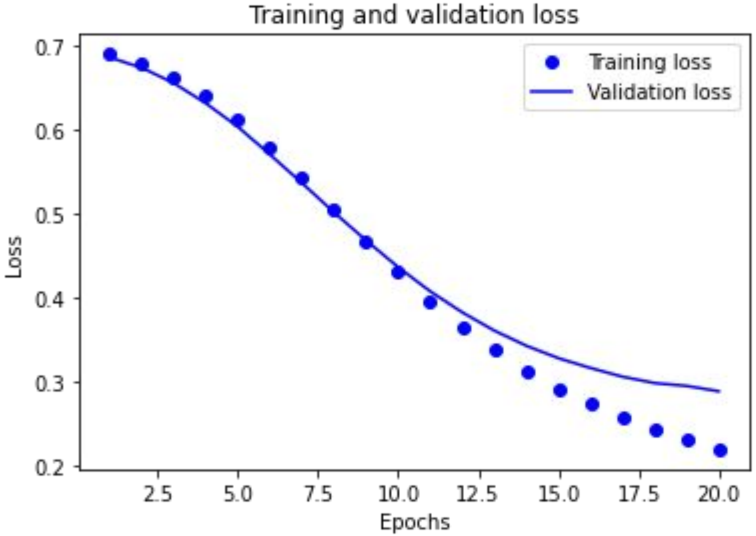
https://colab.research.google.com/drive/1pzfmjWjTCmnBSR0He7LP2LXyWYTvMhL1?usp=sharing

Normalization: Count vs. Freq

As you can see above, the counts are not normalized values, which is not good for the NN (neurons activations prefer normalized values, however, for Dense and Relu it makes no big difference).

However, making just unnormalized counts is the same as binary BoW, since high frequency words will dominate the vector, specially that it's very sparse, causing quick overfitting. In other words, important low freq words are discarded. This can be treated with TF-IDF, or at least by normalizing, so the network might understand that low frequencies have special importance.





Pros and Cons of BoW

All words are normalized, regardless of their index in the vocab	Sparsity = inefficiency = A lot of useless features = confusion of the model = unnecessary big model = more overfitting
Fast inference → Dense layers	Need to pad to max (again inefficiency)
	No context = no sequence = hard to model co-reference, sarcasm, negation...etc: Example Such BoW model has no clue to differentiate the following cases: ● This movie is good --> + ● This movie is bad --> - ● This movie is not good --> - ● This movie is not bad --> + Simple because it cannot link the context of the words "good" and "bad" to the negatition or affirmation context "not"

What we need?

- A model that encode sequence → sequence models
- But if we use the sent actual word + pad, we end up with features=word index! High and low values + not reflecting the word meaning + no similarity encoding
- So we need:
- A mapping of symbolic words into a space where similarity is encoded → Word Embeddings → Word Vectors

References

http://web.stanford.edu/class/cs224n/

http://jalammar.github.io/illustrated-word2vec/

http://jalammar.github.io/illustrated-transformer/

http://jalammar.github.io/illustrated-bert/

http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and

- Deep Learning with Python, Fchollet (Keras)
  - Notebooks
- Hands-On Machine Learning with Scikit-Learn and TensorFlow
  - Notebooks
- Ian Goodfellow DeepLearning Book

References

- http://jalammar.github.io/illustrated-gpt2/
- https://arxiv.org/abs/1906.08237
- https://arxiv.org/abs/1801.06146
- https://arxiv.org/abs/1706.03762
- https://arxiv.org/abs/1906.08237
- https://arxiv.org/abs/1901.02860
- https://arxiv.org/abs/1910.07370
- https://openai.com/blog/better-language-models/
- https://github.com/openai/gpt-3