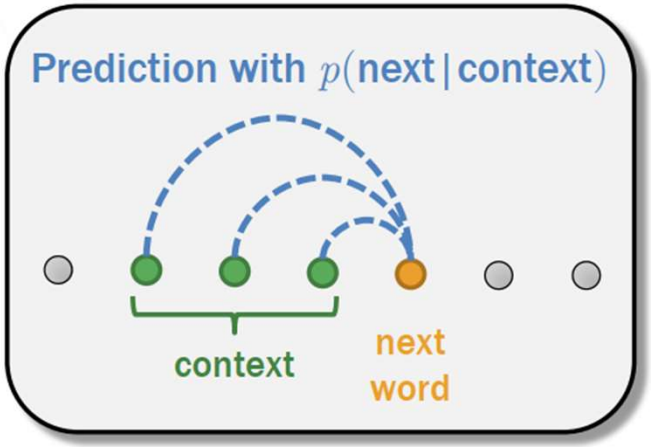
Week 5 – Ngrams LM and Sequence Tagging

EGCO467 Natural Language and Speech Processing

n-gram models

- Language models whose context is truncated to at most $n-1$ previous words

$$P(w_1 \dots w_N) = p(w_1) \prod_{k=2}^{N} p(\underbrace{w_k}_{\text{next word}} | \underbrace{w_{k-n+1} \dots w_{k-1}}_{n-1 \text{ prev words}})$$



  - **Unigram** $(n=1)$: $P(w_1 \dots w_N) = \prod_{k=1}^{N} p(w_k)$
  - **Bigram** $(n=2)$: $P(w_1 \dots w_N) = p(w_1) \prod_{k=2}^{N} p(w_k|w_{k-1})$
  - **Trigram** $(n=3)$: $P(w_1 \dots w_N) = p(w_1)p(w_2|w_1) \prod_{k=3}^{N} p(w_k|w_{k-2}, w_{k-1})$

Formal Definition of LM

- m words: w_1, w_2,..., w_m
- LM is probability $P(w_1, w_2, ..., w_m)$ for the whole text

- I like music <- high probability
- music like I <- low probability

Making a LM

- s1: a brown fox jumps over a lazy dog
- s2: the dog jumps over the fence

1. build vocab: {a, brown, fox, jumps, over, lazy, dog, the, over, fence}
2. P(w) = how many time w appear / how many word total in corpus

N = 14

P(a) = 2/14, P(dog) = 2/14, P(brown) = 1/14, ...

Unigram LM

- Probably of any sequence P(w1,w2,...,wn) = P(w1)P(w2)...P(wn)
- E.g. P(a,brown,fox) = P(a)P(brown)P(fox) = (2/14)*(1/14)*(1/14)

Unigram $(n=1)$: $P(w_1 \dots w_N) = \prod_{k=1}^{N} p(w_k)$

Bigram $(n=2)$: $P(w_1 \dots w_N) = p(w_1) \prod_{k=2}^{N} p(w_k|w_{k-1})$

Bigram LM

- E.g. let's say that in corpus "a" occurs 4 times in: "**a** brown fox", "**a** brown cookie", "**a** big apple", "**a** big book"
- P(brown|a) = 2/4. Because we have two instances of "a brown" out of all 4 instances where "a ..." appears
- So P(w2|w1) = count of w1 followed by w2 / count of w1 followed by x
  = count of w1 followed by w2 / count of w1
- Do this for all word pairs that occur in corpus.

Example

- s1: a brown fox jumps over a lazy dog
- s2: the dog jumps over the fence

- vocab: {a, brown, fox, jumps, over, lazy, dog, the, over, fence}
- P(brown|a) = |{ (a brown) }| / |{ (**a** brown), (**a** lazy) }| = 0.5
- P(jumps|fox) = |{ (fox jumps) }| / |{ (**fox** jumps) }| = 1.0
- P(over|the) = **the over** never occurs, so P = 0
- P(fox|brown) = 1.0

Example

- P(a,brown,fox) = P(a)*P(brown|a)*P(fox|brown) = (2/14)*0.5*1.0

Unigram $(n{=}1)$: $P(w_1 \ldots w_N) = \prod_{k=1}^{N} p(w_k)$

Bigram $(n{=}2)$: $P(w_1 \ldots w_N) = p(w_1) \prod_{k=2}^{N} p(w_k|w_{k-1})$

Trigram LM

- P(w3|w2,w1) = count of sequence (w1,w2,w3) in corpus / count of sequence (w1,w2) in corpus
- Do this for all possible values of w1,w2,w3

n-grams

- $p(w_n|w_{n-1}, w_{n-2},...,w_1)$
- The number of conditions grows with n
- Longer condition -> less chance to appear in corpus -> count = 0
- Maximum used in practice is only trigram
- For n > 3, we use NN to estimate the probability

LSTM Language Model

- At time k, predict kth word, give the previous k-1 words (context).
- Append the predicted word to the context.
- Go the next time step.

- k=3: *The quick* **brown**
- k=4: *The quick brown* **fox**
- k=5: *The quick brown fox* **jumps**
- **....**

perplexity

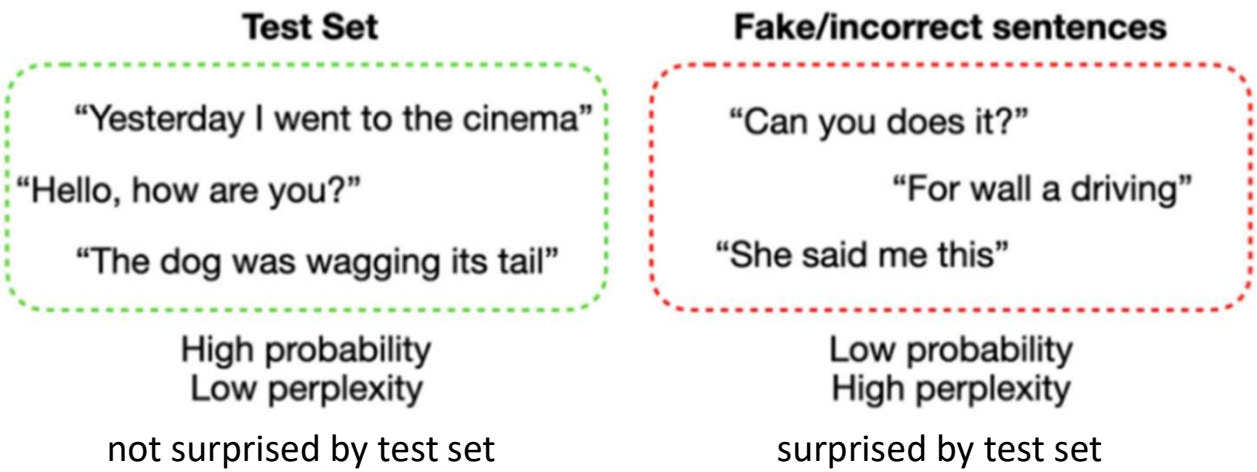
- perplexity = *inverse normalized probability of test set*
- lower is better

$$PP(W) = \frac{1}{P(w_1, w_2, \ldots, w_N)^{1/N}}$$

- W = test set
- w_1 to w_N = all words in the test set
- N = number of words in the test set

https://towardsdatascience.com/perplexity-in-language-models-87a196019a94

Probability of test set



Perplexity

- inverse normalized probability of test set

$$PP(W) = \frac{1}{P(w_1, w_2, \ldots, w_N)^{\frac{1}{N}}}$$

$$= \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$

Example - trigram

- 05 - trigram NLTK.ipynb

Exercise

- Given the same corpus from the word2vec (03 - word2vec basic.ipynb)
- Count how many bigrams there are the corpus
- Calculate the probability of “of a"
- Modify the .ipynb in the previous slide to do this exercise.
- (ignore punctuation marks and lower/upper case)

We are about to study the idea of a computational process.
Computational processes are abstract beings that inhabit computers.
As they evolve, processes manipulate other abstract things called data.
The evolution of a process is directed by a pattern of rules
called a program. People create programs to direct processes. In effect,
we conjure the spirits of the computer with our spells.

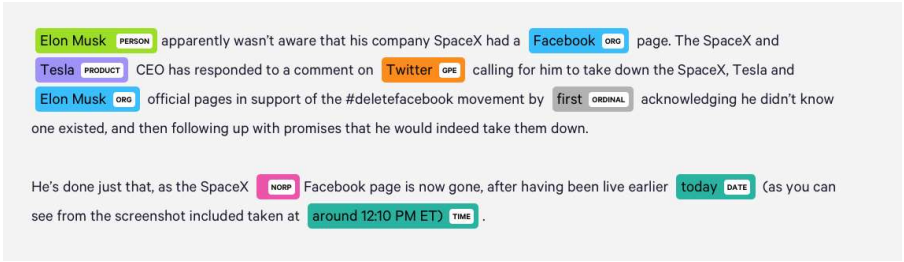
Dealing with OOV words

- If corpus has: I am a police
- Test has: I am a farmer
- p(farmer | a) = 0
- p(I am a farmer) = 0
- perplexity = inf
- Use <unk> for unknown words (don't have in training data)

Sequence tagging problem

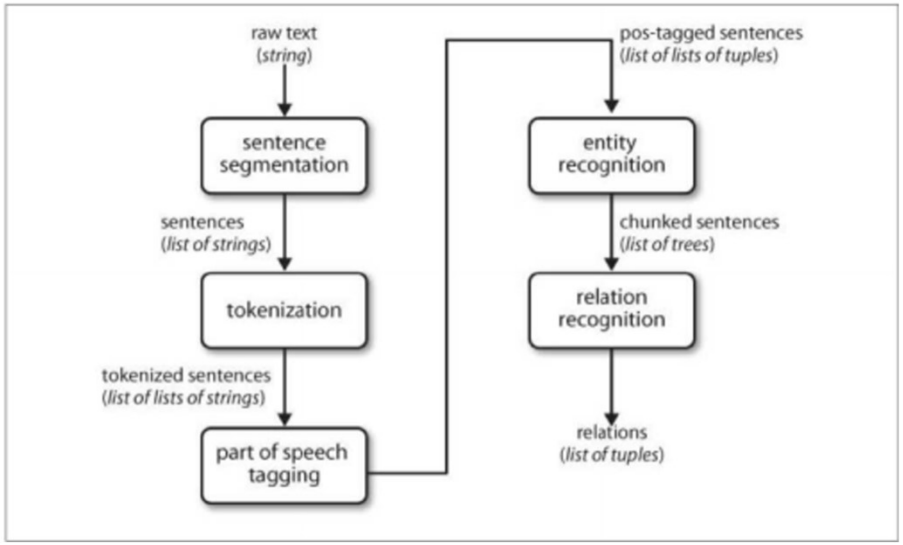
- Label each token, not the entire sequence
- Label whole sequence is classification
- Label each token is sequence tagging

Two main types of sequence tagging



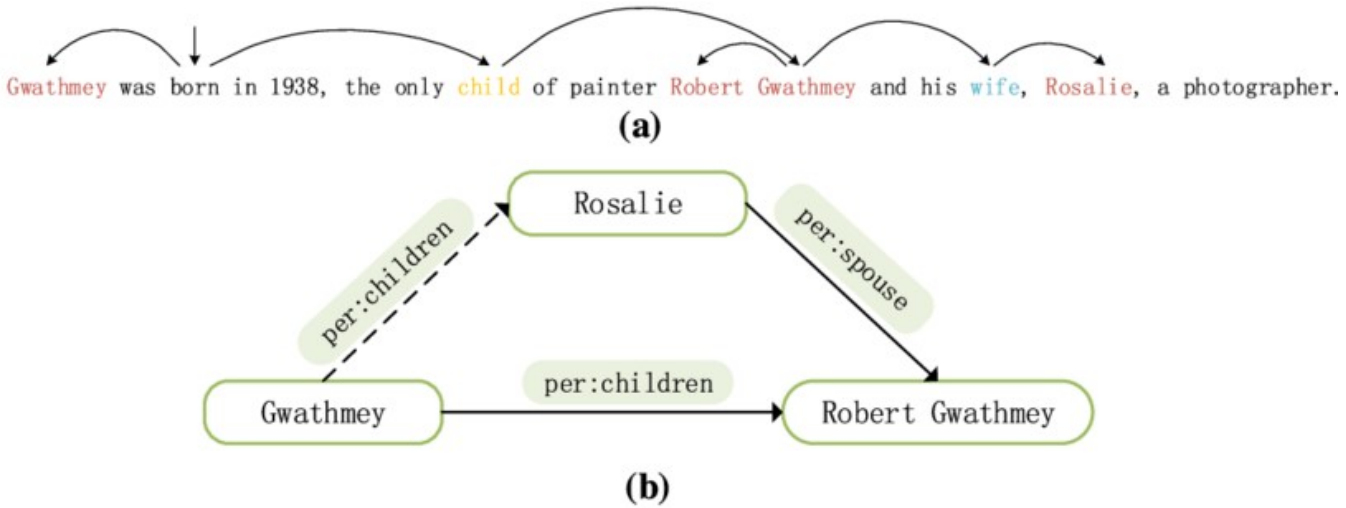
Why do NER/POS

- Information extraction
- Relation extraction

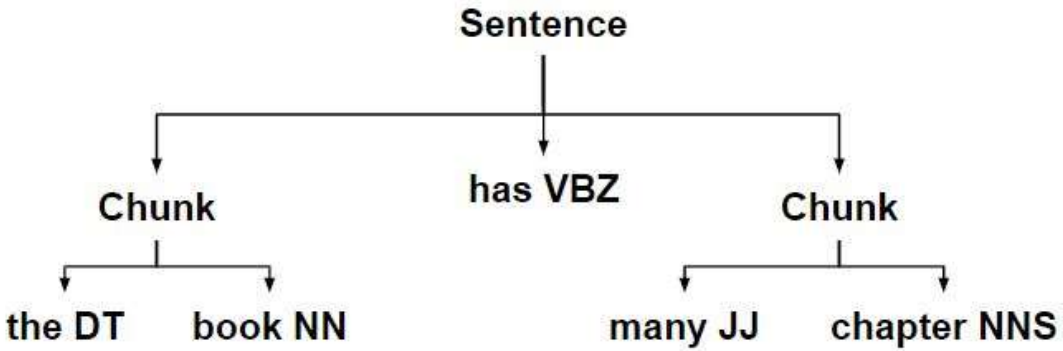


Relation Extraction

- extract relationship between each entity (nouns) in the text



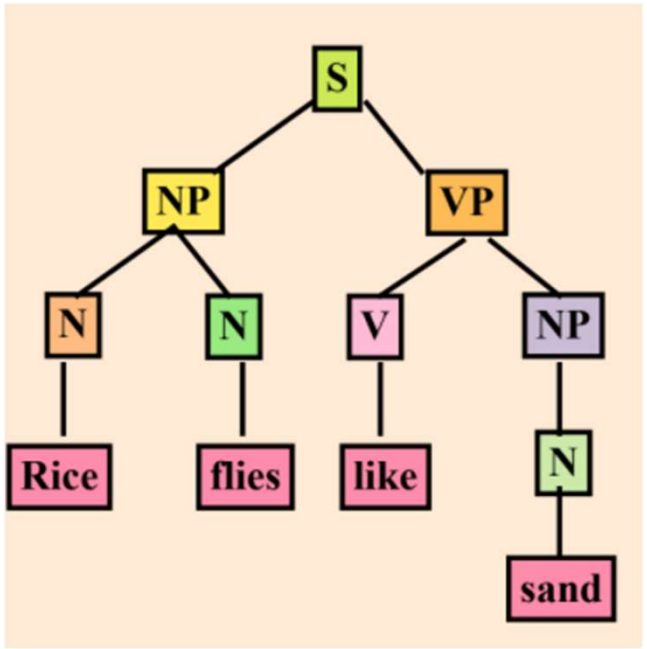
Sentence chunking



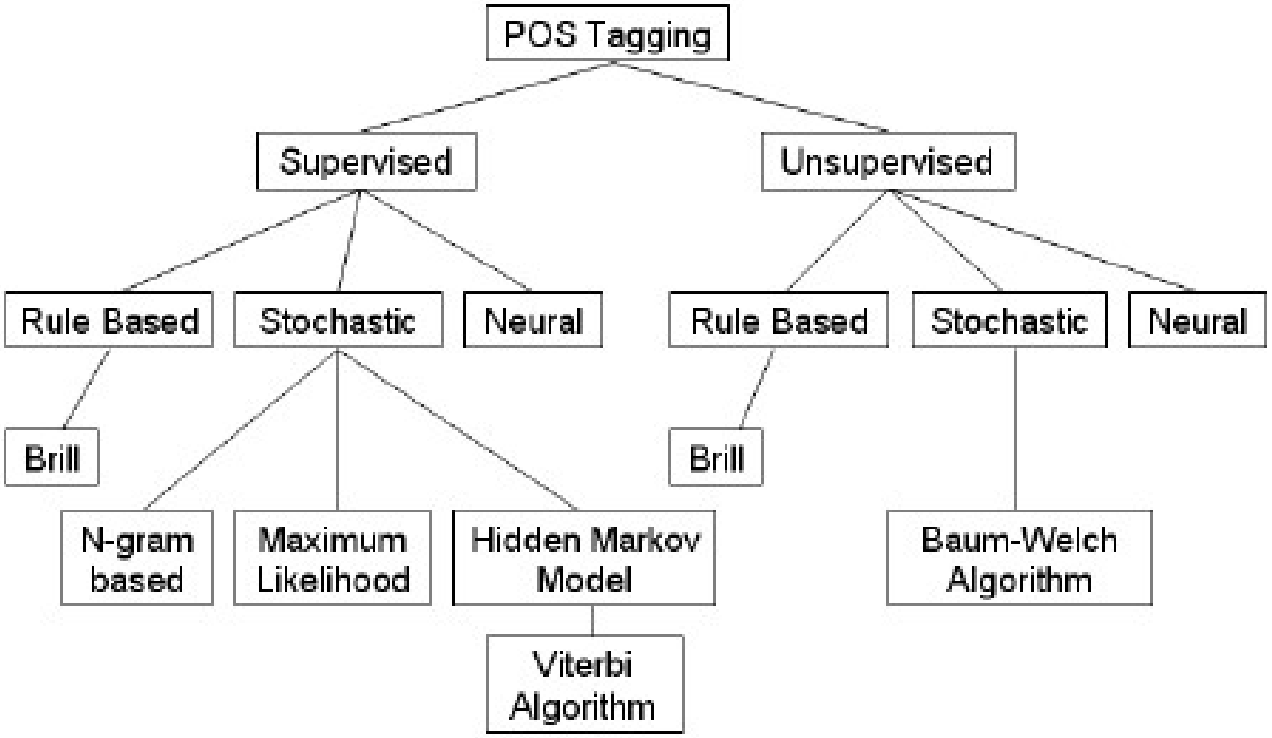
Sentence Parse Tree

Grammar

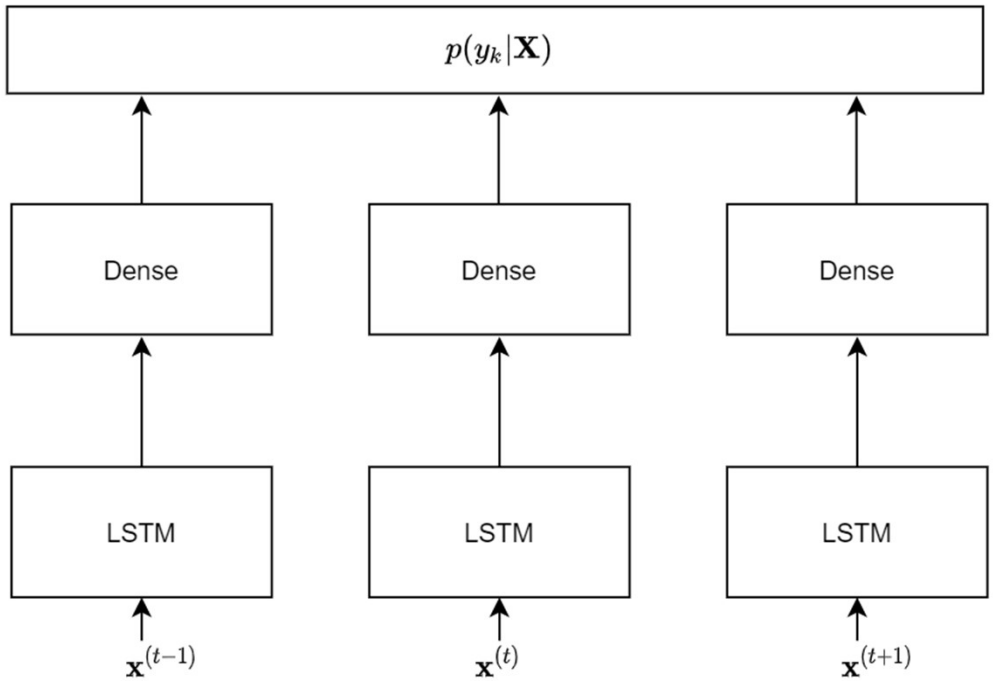
S → NP VP	N → Rice
VP → V NP	N → flies
VP → V	N → sand
NP → N N	V → like
NP → N	P → like
PP → P NP	



How to do POS?

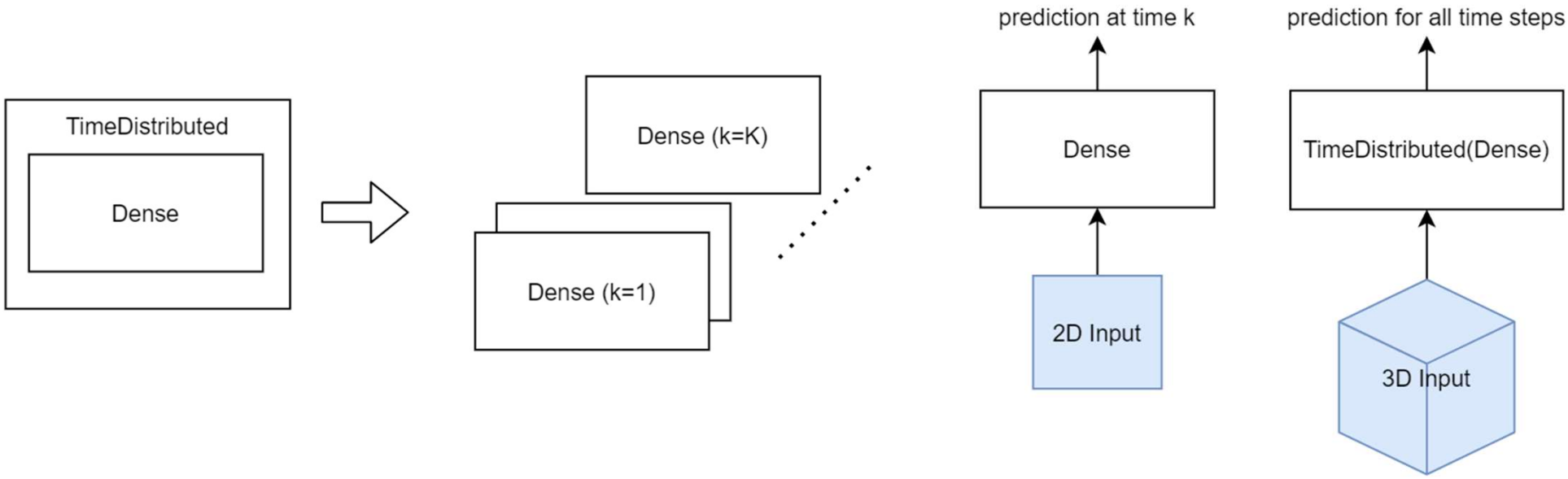

POS Tagging
Supervised
Unsupervised
Rule Based
Stochastic
Neural
Rule Based
Stochastic
Neural
Brill
Brill
N-gram based
Maximum Likelihood
Hidden Markov Model
Baum-Welch Algorithm
Viterbi Algorithm

One way to tag tokens



TimeDistributed

TimeDistributed(Dense(n_tags, activation="softmax"))



Time Distributed

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 50)]	0
embedding (Embedding)	(None, 50, 50)	1758950
dropout (Dropout)	(None, 50, 50)	0
bidirectional (Bidirectional	(None, 50, 200)	120800
time_distributed (TimeDistri	(None, 50, 17)	3417

note: the number of classes = 17
and sequence length = 50

Loss of sequence tagging

- L(t) = loss at time t
- Sequence tagging is doing classification for each time step
- The loss L(t) is just regular cross-entropy like classification

$$L = \frac{1}{T} \sum_{t=0}^{T} L(t)$$

Example - POS tagging

- 05 - LSTM sequence tagging.ipynb

Orchid Dataset

```
<corpus>
<document TPublisher="ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ, กระทรวงวิทยาศาสตร์ เทคโนโลยีและการพลังงาน" EPublisher="National
<paragraph id="1" line_num="12">
<sentence id="1" line_num = "13" raw_txt = "การประชุมทางวิชาการ ครั้งที่ 1">
<word surface="การ" pos="FIXN"/>
<word surface="ประชุม" pos="VACT"/>
<word surface="ทาง" pos="NCMN"/>
<word surface="วิชาการ" pos="NCMN"/>
<word surface="&lt;space&gt;" pos="PUNC"/>
<word surface="ครั้ง" pos="CFQC"/>
<word surface="ที่ 1" pos="DONM"/>
</sentence>
<sentence id="2" line_num = "23" raw_txt = "โครงการวิจัยและพัฒนาอิเล็กทรอนิกส์และคอมพิวเตอร์">
<word surface="โครงการวิจัยและพัฒนา" pos="NCMN"/>
<word surface="อิเล็กทรอนิกส์" pos="NCMN"/>
<word surface="และ" pos="JCRG"/>
<word surface="คอมพิวเตอร์" pos="NCMN"/>
</sentence>
<sentence id="3" line_num = "30" raw_txt = "ปีงบประมาณ 2531">
<word surface="ปีงบประมาณ" pos="NCMN"/>
<word surface="&lt;space&gt;" pos="PUNC"/>
<word surface="2531" pos="NCNM"/>
</sentence>
<sentence id="4" line_num = "36" raw_txt = "เล่ม 1">
<word surface="เล่ม" pos="CNIT"/>
<word surface="&lt;space&gt;" pos="PUNC"/>
<word surface="1" pos="DONM"/>
</sentence>
</paragraph>
```

Virach Sornlertlamvanich, Naoto Takahashi and Hitoshi Isahara. Building a Thai Part-Of-Speech Tagged Corpus (ORCHID). The Journal of the Acoustical Society of Japan (E), Vol.20, No.3, pp 189-198, May 1999./p>

Orchid POS system

The following table shows POS tags as used in ORCHID:

Abbreviation	Part-of-Speech tag	Examples
NPRP	Proper noun	วินโดวส์ 95, โคโรน่า, โค้ก
NCNM	Cardinal number	หนึ่ง, สอง, สาม, 1, 2, 10
NONM	Ordinal number	ที่หนึ่ง, ที่สอง, ที่สาม, ที่1, ที่2
NLBL	Label noun	1, 2, 3, 4, ก, ข, a, b
NCMN	Common noun	หนังสือ, อาหาร, อาคาร, คน
NTTL	Title noun	ครู, พลเอก
PPRS	Personal pronoun	คุณ, เขา, ฉัน
PDMN	Demonstrative pronoun	นี่, นั้น, ที่นั่น, ที่นี่
PNTR	Interrogative pronoun	ใคร, อะไร, อย่างไร
PREL	Relative pronoun	ที่, ซึ่ง, อัน, ผู้
VACT	Active verb	ทำงาน, ร้องเพลง, กิน
VSTA	Stative verb	เห็น, รู้, คือ

Homework 5

- Parse the Orchid data (xmlchid.xml) using Python's xml library https://docs.python.org/3/library/xml.etree.elementtree.html
- Extract the POS tag of each word
- Store data in a list like below
- Save the list to a .pickle file

```
<sentence id="1" line_num = "13" raw_txt = "การประชุมทางวิชาการ ครั้งที่ 1">
<word surface="การ" pos="FIXN"/>
<word surface="ประชุม" pos="VACT"/>
<word surface="ทาง" pos="NCMN"/>
<word surface="วิชาการ" pos="NCMN"/>
<word surface="&lt;space&gt;" pos="PUNC"/>
<word surface="ครั้ง" pos="CFQC"/>
<word surface="ที่ 1" pos="DONM"/>
</sentence>
```

```
[
(["การ", "ประชุม", "ทาง", "วิชาการ", " ", "ครั้ง", "ที่ 1"], ["FIXN", "VACT", "NCMN", "NCMN", "PUNC", "CFQC", "DONM"]),
(next sentence),
...
]
```