Machine Learning for Natural Language Processing

Task Specific Modelling of Text

Session 3

Ḃenjamin Muller

INRIA Paris - ALMANACH
`benjamin.muller@inria.fr`

Course Outline

Lecture 2 recap

How to represent text into vectors ?

Lecture 2 recap

How to represent text into vectors ?

- Feature based approach for word representation (e.g. Wordnet)
- Distributional approach using co-occurrence statistics

Lecture 2 recap

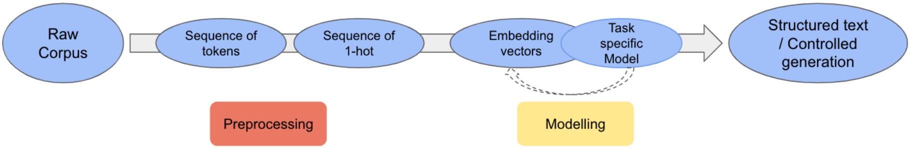
How to represent text into vectors ?

- Feature based approach for word representation (e.g. Wordnet)
- Distributional approach using co-occurrence statistics
- Continuous representation with the word2vec Skip-Gram Model
  - Embedding matrices
  - Predicting context words with focus words
  - Trained with Negative Sampling using Stochastic Gradient Descent
- Evaluating word vectors

Lecture Outline

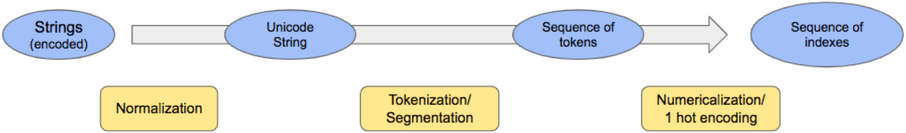
- Pre-processing (encoding, tokenization)
- Two sequence labelling tasks : POS tagging, NER
- Sequence Labelling: Sentiment Analysis
- Sequence Generation (lecture 5)

Standart NLP pipeline



Preprocessing

Preprocessing pipeline



Preprocessing

- Encoding
- Tokenization: Word and Sentence Segmentation

What is Encoding ?

At the beginning of any NLP problems is a collection of **strings**

NB : encoding is at play in every computer programs that handle strings !

What is Encoding ?

At the beginning of any NLP problems is a collection of **strings**

Reminder: a string is *data type* used to **represent** text

NB : encoding is at play in every computer programs that handle strings !

What is Encoding ?

At the beginning of any NLP problems is a collection of **strings**

Reminder: a string is *data type* used to **represent** text

Definition: The process of encoding converts information from a source into **symbols** for **communication** or **storage**.

NB : encoding is at play in every computer programs that handle strings !

What is Encoding ?

At the beginning of any NLP problems is a collection of **strings**

Reminder: a string is *data type* used to **represent** text

Definition: The process of encoding converts information from a source into **symbols** for **communication** or **storage**.

In other words, the way strings are stored in the memory of the computer is called the **encoding**

NB : encoding is at play in every computer programs that handle strings !

Encoding properties

A good encoding algorithm should have the following properties:

- Encode any string in memory (in bits)
- Universal
  e.g: *should be able to share string across computers and still have the same rendering*

UTF-8 : the current global encoding standard

- **Unicode**: a universal table that maps any character to a *code point*
  e.g : "Hello" → U+0048 U+0065 U+006C U+006C U+006F
- **Encoding**: a way to store any code-point to a sequence of bits.

UTF-8 : the current global encoding standard

- **Unicode**: a universal table that maps any character to a *code point*
  e.g : "Hello" → U+0048 U+0065 U+006C U+006C U+006F
- **Encoding**: a way to store any code-point to a sequence of bits.

Definition UTF-8 (8-bit Unicode Transformation Format) is a variable width character encoding capable of encoding all 1,112,064 valid code points in Unicode using one to four 8-bit bytes. It is the standard in the internet.
Guidline: Everytime you face some encoding problem/bugs, make sure everything is correctly encoded in UTF-8. If it is not, convert it to UTF-8.

Preprocessing

- Encoding
- Tokenization: Word and Sentence Segmentation

Text segmentation

Definition: Text Segmentation is the process of splitting raw (encoded) text (i.e. list of characters) into units of interest.

Text segmentation

Definition: Text Segmentation is the process of splitting raw (encoded) text (i.e. list of characters) into units of interest.

Two level of segmentation (usually) required :

- Split raw text into **modelling units** (ex: sentence, paragraph, 1000 characters...)
- Split modelling units into sequence of **basic units** (referred as **tokens**) (e.g: words, word-pieces, characters, ...)

Text segmentation

Definition: Text Segmentation is the process of splitting raw (encoded) text (i.e. list of characters) into units of interest.

Two level of segmentation (usually) required :

- Split raw text into **modelling units** (ex: sentence, paragraph, 1000 characters...)
- Split modelling units into sequence of **basic units** (referred as **tokens**) (e.g: words, word-pieces, characters, ...)

Two distinct approaches:

- Linguistically informed
  e.g. word, sentence segmentation...
- Statistically informed
  e.g. frequent sub-words (wordpieces, sentencepieces...)

Text segmentation

Definition: Text Segmentation is the process of splitting raw (encoded) text (i.e. list of characters) into units of interest.

Two level of segmentation (usually) required :

- Split raw text into **modelling units** (ex: sentence, paragraph, 1000 characters...)
- Split modelling units into sequence of **basic units** (referred as **tokens**) (e.g: words, word-pieces, characters, ...)

Two distinct approaches:

- Linguistically informed
  e.g. word, sentence segmentation...
- Statistically informed
  e.g. frequent sub-words (wordpieces, sentencepieces...)

NB: Text Segmentation algorithms are task and model dependant

Tokenization

Tokenization is a broad (vague) term which simply refers to the process of splitting raw text into sequences of tokens.

Therefore, It can refer to word segmentation, wordpieces segmentation, character splitting...

NB: In its most usual meaning, tokenization refers to word segmentation.

Preprocessing : Word Segmentation

- First step for most NLP application: **text segmentation**
- e.g: Given a string $\mathbf{x}_{1:T}$:

  predict for each position if end of word and sentence

- Can be framed as a character level task

  input: `une industrie métallurgique existait.`
  output: `IIEIIIIIIIIIEIIIIIIIIIIIIIIEIIIIIIIIIEE`

- Easy task for most languages and not too noisy text
- Can be very complex in some cases (Chinese, User-Generated Content...)

Segmentation: Annotated data

- Can be trained/evaluated on Universal Dependency data (60+ languages)[1]:

```
 text =...  une industrie métallurgique existait.
18  une            DET    19  det    _
19  industrie      NOUN   21  nsubj  _
20  métallurgique  ADJ    19  amod   _
21  existait       VERB   4   ccomp  SpaceAfter=No
22  .              PUNCT  4   punct  _
```

[1] https://universaldependencies.org/

Word Segmentation: a complex task (UGC[2] example)

- How to segment UGC text? e.g :

```
text = J'vais regarder teen wolf :))  @kino13...!!!
```

1 J'
2 vais
3 regarder
4 teen
5 wolf
15 :))
16 @kino13
17 ...
18 !!!

[2] User Generated Content

Sentence Segmentation: a complex task

How to segment transcripted French speech ?

e.g : *euh il y avait donc une euh jeune fille qui regardait dans une boutique apparemment une pâtisserie qui semblait avoir faim qui a profité de ce que le livreur s' éloigne pour euh voler un une baguette euh a rencontré donc Charlot à ce moment -là lui est rentrée dedans euh dans la confusion donc une euh une passante a dénoncé la jeune fille au livreur qui a couru après la jeune fille euh les policiers sont arrivés en raison du du du vacarme je p je pense* (Gerdes)

Sentence Segmentation: a complex task (speech text example)

How to segment transcripted French speech ?

e.g : *euh il y avait donc une euh jeune fille qui regardait dans une boutique apparemment une pâtisserie qui semblait avoir faim qui a profité de ce que le livreur s' éloigne pour euh voler un une baguette euh a rencontré donc Charlot à ce moment -là lui est rentrée dedans* **END** *euh dans la confusion donc une euh une passante a dénoncé la jeune fille au livreur qui a couru après la jeune fille euh* **END** *les policiers sont arrivés en raison du du du vacarme je p je pense* (Gerdes)

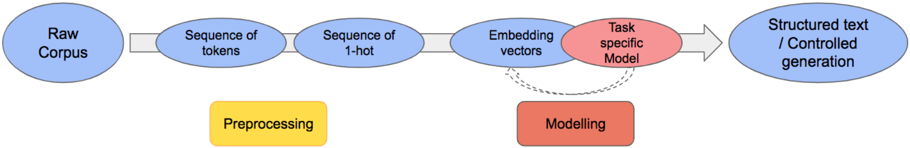
Segmentation

How to approach segmentation ?

- Easy cases: Define set of rules (e.g. using regex)
- Complex cases: Build a character-level sequence labelling model using annotated data

Modelling

Standart NLP pipeline



Sequence Modelling

Modelling Framework

- Rule-based approach (e.g. defining regex for extraction and rules for handling ambiguity)
- **Statistical approach**
- Neural approach (lectures 4 and 5)

Type of tasks

- Sequence Labelling (lecture 3)
- Sequence Classification (lecture 3)
- Sequence Generation (lecture 5)
- Topic Models (lecture 6)

Sequence Labelling

Definition: Sequence Labelling is a type of task that involves the assignment of **a label/tag** to each element of a sequence.

We will focus on two sequence labelling tasks:

- Part-of-Speech Tagging (POS)
- Name Entity Recognition (NER)

Part of Speech Tagging (POS tagging)

Input: *Cette exposition nous apprend que dès le XIIe siècle , à Dammarie-sur-Saulx , entre autres sites , une industrie métallurgique existait .*

Output: Cette`/DET` exposition`/NOUN` nous`/PRON` apprend`/VERB` que`/SCONJ` dès`/ADP` le`/DET` XIIe`/ADJ` siècle`/NOUN` ,`/PUNCT` à`/ADP` Dammarie-sur-Saulx`/PROPN` ,`/PUNCT` entre`/ADP` autres`/ADJ` sites`/NOUN` ,`/PUNCT` une`/DET` industrie`/NOUN` métallurgique`/ADJ` existait`/VERB` .`/PUNCT`

Part of speeches

- Part of speech: morphosyntactic categories (not semantic)
  - Syntactic function: how words are a composed together
  - Morphological properties: how words are formed (e.g. prefix/suffix)
- Divided into two broad categories:
- Closed class: new words rarely appear. e.g. conjunction or pronouns
- Open class: new words often appear. e.g. nouns or verbs

Part of speeches: open classes

- **Noun:** word to describe thing, people, concept, entity, etc. Can occur with adjective and determiners, can be the subject of sentence, can be marked for plural with *-s*

  dog, apple, robot

- **Verb:** word to describe action, process, state of being, etc. Can be inflected to mark tense, voice, or aspect

  eat, run, drive

- **Adjective:** word to describe property or quality, etc. Modify a noun or noun phrase.

  big, blue, good

- **Adverb:** word to express place, time, frequency, etc. Diverse class, modify verbs, adjectives, adverbs, whole phrases or sentences.

  quickly, often, sometimes

Part of speeches: closed classes

- **Auxiliary verbs:** add information to a clause or sentence, such as tense, voice, aspect, etc. Usually modify a verb.

  be, have

- **Conjunction:** join two nouns, phrases, or clauses.

  and, or, if

- **Pronouns:** shorthand to refer to noun phrase, person, entity, etc.

  she, him, who

- **Determiners:** add information to a noun or noun phrase. Often appear at the beginning of noun phrase

  a, the, any

Part of speech tagging

- Why hard? ambiguity:

Time flies like an arrow

- `[N V Adp Det N]` or `[N N V Det N]`?
- But also easy? most frequent tag baseline: $\sim$ 88% for English

- Want to capture local information (*like* can be a verb or preposition) and context information (`V` often follows `N`).

Named Entity Recognition (NER)

- Detect and classify named entities in text.

Input: During World War II, Turing worked for the Government Code and Cypher School at Bletchley Park.

Output: During World War II, [Turing]$_{\texttt{PER}}$ worked for the [Government Code and Cypher School]$_{\texttt{ORG}}$ at [Bletchley Park]$_{\texttt{LOC}}$.

- Standard tag set: `PER`, `ORG`, `LOC`
- Sometimes: `TIME`, `MONEY`, ...
- For other applications, names of products, books, molecules, etc.

Named Entity Recognition (NER)

- Framed as sequence labelling using BIO tag scheme
- Using BIO makes NER a token level task
- B for beginning of entity, I for inside entity, O for other

  During/O World/O War/O II/O ,/O Turing/B-PER worked/O for/O the/O Government/B-ORG Code/I-ORG and/I-ORG Cypher/I-ORG School/I-ORG at/O Bletchley/B-LOC Park/I-LOC

- Why BIO? adjacent entities: *[United States] [Department of State]*

United	States	Department	of	State
/B-ORG	/I-ORG	/B-ORG	/I-ORG	/I-ORG

- Sometimes BILOU: L for last, U for unit

Discriminative modeling

Let $(Y_t, X_t)_{t \in \{1,\ldots,n\}}$ be our observed sequence of labels and tokens

Our goal is to learn the distribution

$$P(Y_1, \ldots, Y_T \mid X_1, \ldots, X_T)$$

Discriminative modeling

- Naive solution: formulate as n independant classification problems
- Given features $\mathbf{x}_t$ representing word t and its context:
- Learn a linear classifier with logistic regression to predict label y_t.
- Examples of features:
  - previous word x_{t-1} and/or next word x_{t+1}
  - pair of word (x_{t-1}, x_t) or (x_t, x_{t+1})
- Limitation: does not take into account previous/next predictions
- Solution: condition the prediction of y_t on y_{t-1}

Maximum entropy Markov models

- Our goal is to learn the distribution

$$P(Y_1, ..., Y_T \mid X_1, ..., X_T)$$

- Using the chain rule, we can factorize as

$$P(Y_{1:T} \mid X_{1:T}) = \prod_{t=1}^{T} P(Y_t \mid Y_{1:t-1}, X_{1:T})$$

- Then, use Markov independence assumption:

$$P(Y_t \mid Y_{1:t-1}, X_{1:T}) = P(Y_t \mid Y_{t-1}, X_{1:T})$$

- Use a (log) linear model to parametrize this distribution.

Maximum entropy Markov models

- We introduce feature vector $\Phi(\mathbf{x}, t, y_t, y_{t-1})$ and get

$$p(y_t \mid y_{t-1}, \mathbf{x}) = \frac{\exp(\mathbf{w}^\top \Phi(\mathbf{x}, t, y_{t-1}, y_t))}{\sum_{k \in \mathcal{Y}} \exp(\mathbf{w}^\top \Phi(\mathbf{x}, t, y_{t-1}, k))}$$

- This is a regular log linear model:
  each class probability is a "learnt" (w) linear combination of feature extracted from x, y_{t-1}, y_t

- The parameters $\mathbf{w}$ are learned using (stochastic) gradient descent
- The training data corresponds to $(\mathbf{x}_{1:T}, t, y_{t-1}, y_t)$

- Then, we have a model of

$$p(y_1, \ldots, y_T \mid x_1, \ldots, x_T).$$

Maximum Entropy Markov Models (MEMMs): inference

- Given trained model and input $\mathbf{x}$,

  how to compute most probable sequence of labels?

- Naive solution: greedy decoding
- For $t = 1, \ldots T$:

$$y_t = \operatorname*{argmax}_k p(k \mid y_{t-1}, \mathbf{x}_{1:T})$$

- **Limitation:** what if

$$p(i \mid y_t) = p(j \mid y_t) + \varepsilon$$

but

$$\max_k p(k \mid i) \ll \max_k p(k \mid j)$$

Maximum entropy Markov models: inference

- Compute most probable sequence using Viterbi!
- We define:

$$s(t,k) = \max_{\substack{\mathbf{y}_{1:t} \\ y_t = k}} p(\mathbf{y}_{1:t} \mid \mathbf{x})$$

- Then, we have

$$\begin{aligned}
s(t,k) &= \max_{\substack{\mathbf{y}_{1:t} \\ y_t = k}} p(k \mid y_{t-1}, \mathbf{x}) p(\mathbf{y}_{1:t-1} \mid \mathbf{x}) \\
&= \max_j \max_{\substack{\mathbf{y}_{1:t-1} \\ y_{t-1} = j}} p(k \mid y_{t-1}, \mathbf{x}) p(\mathbf{y}_{1:t-1} \mid \mathbf{x}) \\
&= \max_j \max_{\substack{\mathbf{y}_{1:t-1} \\ y_{t-1} = j}} p(k \mid j, \mathbf{x}) p(\mathbf{y}_{1:t-1} \mid \mathbf{x}) \\
&= \max_j p(k \mid j, \mathbf{x}) \max_{\substack{\mathbf{y}_{1:t-1} \\ y_{t-1} = j}} p(\mathbf{y}_{1:t-1} \mid \mathbf{x}) \\
&= \max_j p(k \mid j, \mathbf{x}) s(t-1, j)
\end{aligned}$$

Maximum entropy Markov models: inference

- Given a sequence of tokens $\mathbf{x}_{1:T}$ and parameters of the model.
- Initialization: $\forall k,\ s(1,k) = p(k \mid \texttt{Start}, \mathbf{x}_{1:T})$
- For $t = 2, ..., T$:

$$s(t,k) = \max_j p(k \mid j, \mathbf{x}_{1:T}) \times s(t-1, j)$$

- return $\max_k s(T,k)$

- Complexity of algorithm: $O(dT|\mathcal{Y}|^2)$, where d is number of features.

Features for sequence tagging

- We need to define $\Phi(\mathbf{x}, t, y_{t-1}, y_t)$
- Transition features (similar to HMM):

$$\delta(y_{t-1} = \texttt{Noun},\ y_t = \texttt{Verb})$$

- Emission features (similar to HMM):

$$\delta(x_t = \textit{eating},\ y_t = \texttt{Verb})$$

- But also (impossible in HMM):

$$\delta(x_{t-1} = \textit{the},\ x_t = \textit{flies},\ y_t = \texttt{Noun})$$

- Or:

$$\delta(x_t = \textit{flies},\ y_{t-1} = \texttt{Det},\ y_t = \texttt{Noun})$$

Features for POS tagging or NER

Word features

- Prefix and suffix: *un-*, *-ing*, *-ed*, ...
- Word shape: `Capitalization`, `ContainsDigit`, `DD/DD/DDDD`, ...
- Gazetteers: list of named entities

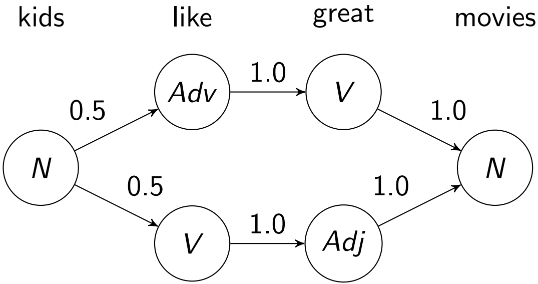
Context features

- Word before and after
- Tag before and after

Nowadays

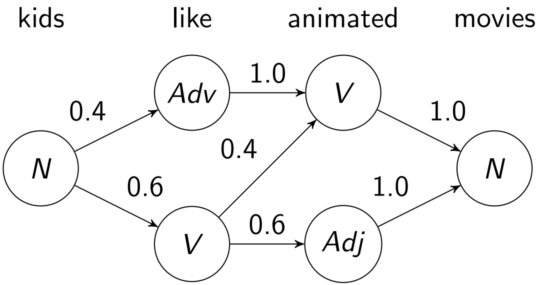
- Most word features replaced by pre-trained word vectors
- Most context features can be replaced by neural network

Label bias problem



- We have $p(V \mid Adv, \text{great}) = 1$, because we've always observed verbs after adverbs in training data
- Both paths have equal probabilities, states with single outgoing transition ignore observation!
- Because local normalization
- Because MEMM only condition on observation

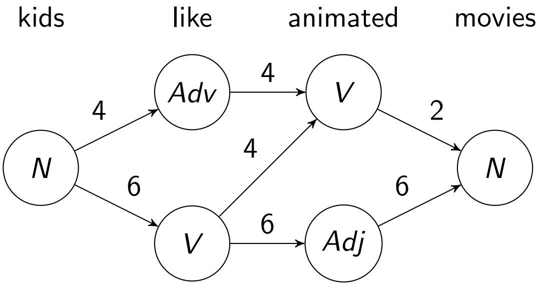
Label bias problem



- Locally, we have: N, V, Adj, N
- But most probably: N, Adv, V, N
- Locally normalized models: favor state with low transition entropy!

Label bias problem

- Solution? Global normalization!



- Here most probable sequence: N, V, Adj, N.
- If path contains one very unlikely transition: can be strongly penalized

Conditional Random Fields

Conditional Random Fields

- Discriminative model of **y** given **x**:

$$p(y_1, \dots, y_T \mid x_1, \dots, x_T) = \frac{\exp(\mathbf{w}^\top \Phi(\mathbf{y}, \mathbf{x}))}{\sum_{\mathbf{z} \in \mathcal{Y}^T} \exp(\mathbf{w}^\top \Phi(\mathbf{z}, \mathbf{x}))}$$

- One large log linear model: normalize over all sequences!
- $\Phi(\mathbf{y}, \mathbf{x})$: feature vector for the pair of the whole sequences **x** and **y**
- Normalization factor has exponential number of sequences:
  $\rightarrow$ *Howtocomputethisefficiently*?

Conditional Random Fields

- Main idea: restrict the feature vector to decompose as

$$\Phi(\mathbf{y}, \mathbf{x}) = \sum_{t=1}^{T} \phi(\mathbf{x}, t, y_{t-1}, y_t)$$

- Example of feature vector $\phi(\mathbf{x}, t, y_{t-1}, y_t)$:

$$\phi_r(y_{t-1}, y_t) + \phi_e(x_t, y_t)$$

- This means that transition score: independent of position/input
- Association score between label and input: only local
- In this case main difference with HMM:

discriminative + global normalization

- Most common extension: ϕ_e can depend on whole $\mathbf{x}$

Conditional Random Fields

- Using previous decomposition, efficient inference:

$$\begin{aligned}
\operatorname{argmax} p(\mathbf{y} \mid \mathbf{x}) &= \operatorname{argmax} \frac{\exp(\mathbf{w}^\top \Phi(\mathbf{y}, \mathbf{x}))}{\sum_{\mathbf{z}} \exp(\mathbf{w}^\top \Phi(\mathbf{z}, \mathbf{x}))} \\
&= \operatorname{argmax} \exp(\mathbf{w}^\top \Phi(\mathbf{y}, \mathbf{x})) \\
&= \operatorname{argmax} \mathbf{w}^\top \Phi(\mathbf{y}, \mathbf{x}) \\
&= \operatorname{argmax} \sum_t \mathbf{w}^\top \phi(\mathbf{x}, t, y_{t-1}, y_t)
\end{aligned}$$

- Again, we can use Viterbi to compute the argmax!

Conditional Random Fields: inference

- We introduce

$$
\begin{aligned}
s(t,k) &= \max_{\substack{\mathbf{y}_{1:t} \\ y_t = k}} \mathbf{w}^\top \Phi(\mathbf{y}_{1:t}, x) \\
&= \max_{\substack{\mathbf{y}_{1:t} \\ y_t = k}} \mathbf{w}^\top \phi(\mathbf{x}, t, y_{t-1}, y_t) + \mathbf{w}^\top \Phi(\mathbf{y}_{1:t-1}, \mathbf{x}) \\
&= \max_j \max_{\substack{\mathbf{y}_{1:t-1} \\ y_{t-1} = j}} \mathbf{w}^\top \phi(\mathbf{x}, t, j, k) + \mathbf{w}^\top \Phi(\mathbf{y}_{1:t-1}, \mathbf{x}) \\
&= \max_j \mathbf{w}^\top \phi(\mathbf{x}, t, j, k) + \max_{\substack{\mathbf{y}_{1:t-1} \\ y_{t-1} = j}} \mathbf{w}^\top \Phi(\mathbf{y}_{1:t-1}, \mathbf{x}) \\
&= \max_j \mathbf{w}^\top \phi(\mathbf{x}, t, j, k) + s(t-1, j)
\end{aligned}
$$

- Using simple ϕ:

$$
s(t,k) = \max_j \mathbf{w}^\top \phi_r(j,k) + \mathbf{w}^\top \phi_e(x_t, k) + s(t-1, j)
$$

Conditional Random Fields: inference

- Given an input $\mathbf{x}$ and a trained model $\mathbf{w}$
- Initialization: $s(1,k) = \mathbf{w}^\top \phi(\mathbf{x}, 1, \texttt{Start}, k)$
- For $t = 2, \ldots T$:

$$s(t,k) = \max_j s(t-1, j) + \mathbf{w}^\top \phi(x, t, j, k)$$

- Return $\max_k s(T, k)$.

Conditional Random Fields: learning

- How to learn the parameters corresponding to the model?
- Use stochastic gradient descent
- How to compute the gradient w.r.t. $\mathbf{w}$ for one example?
- Loss for a given example $\mathbf{x}$:

$$\begin{aligned}\ell(\mathbf{x},\mathbf{y},\mathbf{w}) &= -\log(p(\mathbf{y} \mid \mathbf{x},\mathbf{w})) \\ &= -\mathbf{w}^\top \Phi(\mathbf{y},\mathbf{x}) + \log\left(\sum_{\mathbf{z}\in\mathcal{Y}^T} \exp(\mathbf{w}^\top \Phi(\mathbf{z},\mathbf{x}))\right)\end{aligned}$$

- First term is easy to compute. What about second term?

Conditional Random Fields: learning

- We want to compute the gradient of

$$A = \log\left(\sum_{\mathbf{z}\in\mathcal{Y}^T} \exp(\mathbf{w}^\top \Phi(\mathbf{z},\mathbf{x}))\right)$$

- First, gradient of log-sum-exp:

$$\frac{\partial}{\partial \alpha_i} \log\left(\sum_i \exp \alpha_i\right) = \frac{\alpha_i}{\sum_j \alpha_j}$$

- Thus, gradient of second term:

$$\frac{\partial A}{\partial \mathbf{w}} = \sum_{\mathbf{z}\in\mathcal{Y}^T} p(\mathbf{z} \mid \mathbf{x})\Phi(\mathbf{z},\mathbf{x})$$

$$
\begin{aligned}
\frac{\partial A}{\partial \mathbf{w}} &= \sum_{\mathbf{z} \in \mathcal{Y}^T} p(\mathbf{z} \mid \mathbf{x}) \Phi(\mathbf{z}, \mathbf{x}) \\
&= \sum_{\mathbf{z} \in \mathcal{Y}^T} p(\mathbf{z} \mid \mathbf{x}) \sum_{i=1}^{T} \phi(\mathbf{x}, t, z_{t-1}, z_t) \\
&= \sum_{t=1}^{T} \sum_{\mathbf{z} \in \mathcal{Y}^T} p(\mathbf{z} \mid \mathbf{x}) \phi(\mathbf{x}, t, z_{t-1}, z_t) \\
&= \sum_{t=1}^{T} \sum_{i,j} \sum_{\substack{\mathbf{z} \in \mathcal{Y}^T \\ z_{t-1}=i, z_t=j}} p(\mathbf{z} \mid \mathbf{x}) \phi(\mathbf{x}, t, z_{t-1}, z_t) \\
&= \sum_{t=1}^{T} \sum_{i,j} \phi(\mathbf{x}, t, i, j) \sum_{\substack{\mathbf{z} \in \mathcal{Y}^T \\ z_{t-1}=i, z_t=j}} p(\mathbf{z} \mid \mathbf{x}) \\
&= \sum_{t=1}^{T} \sum_{i,j} \phi(\mathbf{x}, t, i, j) q(t, i, j)
\end{aligned}
$$

Conditional Random Fields: learning

- How to compute $q(t, i, j)$ efficiently?
- We introduce $f(k, z_{k-1}, z_k) = \exp(\mathbf{w}^\top \phi(\mathbf{x}, k, z_{k-1}, z_k))$.

$$
\begin{aligned}
q(t,i,j) &= \frac{1}{Z} \sum_{\substack{\mathbf{z} \in \mathcal{Y}^T \\ z_{t-1}, z_t = i, j}} \prod_{k=1}^{T} f(k, z_{k-1}, z_k) \\
&= \frac{1}{Z} \left[\sum_{\substack{\mathbf{z}_{1:t-1} \\ z_{t-1} = i}} \prod_{k=1}^{t-1} f(k, z_{k-1}, z_k) \right] f(t,i,j) \left[\sum_{\substack{\mathbf{z}_{t:T} \\ z_t = j}} \prod_{k=t+1}^{T} f(k, z_{k-1}, z_k) \right]
\end{aligned}
$$

- We'll compute the left (α) and right (β) terms with dynamic programming

Conditional Random Fields: learning

- Forward-backward algorithm
- Forward computation:

$$\alpha(1,k) = \exp(\mathbf{w}^\top \phi(\mathbf{x}, \texttt{Start}, 1, k))$$
$$\alpha(t,k) = \sum_j \alpha(t-1,j) \exp(\mathbf{w}^\top \phi(\mathbf{x}, t, j, k))$$

- Normalization:

$$Z = \sum_k \alpha(T,k)$$

- Backward computation:

$$\beta(T,k) = 1$$
$$\beta(t,k) = \sum_j \beta(t+1,j) \exp(\mathbf{w}^\top \phi(\mathbf{x}, t+1, k, j))$$

Conditional Random fields: learning

- Finally, we have:

$$\sum_{\substack{\mathbf{z} \in \mathcal{Y}^{\mathbf{T}} \\ z_{t-1}=i, z_t=j}} p(\mathbf{z} \mid \mathbf{x}) = \frac{1}{Z} \alpha(t-1, i) \times f(t, i, j) \times \beta(t, j)$$

Conditional Random Fields: learning

- To compute the gradient:
- Perform forward-backward algorithm to compute the marginal probability over tag bigrams:

$$q(t, i, j) = \sum_{\substack{\mathbf{z} \in \mathcal{Y}^T \\ z_{t-1}=i, z_t=j}} p(\mathbf{z} \mid \mathbf{x})$$

- Compute gradient w.r.t. **w**:

$$-\sum_{t=1}^{T} \phi(\mathbf{x}, t, y_{t-1}, y_t) + \sum_{t=1}^{T} \sum_{i,j} q(t, i, j) \phi(\mathbf{x}, t, i, j)$$

$$\sum_{t=1}^{T} \sum_{i,j} \left[q(t, i, j) - \delta(y_{t-1} = i, y_t = j) \right] \phi(\mathbf{x}, t, i, j)$$

Conditional Random Fields: learning with autograd

- With torch or tensorflow, no explicit computation of gradients
- Reminder, loss for one example:

$$\ell(\mathbf{x}, \mathbf{y}, \mathbf{w}) = -\mathbf{w}^\top \Phi(\mathbf{y}, \mathbf{x}) + \log \left(\sum_{\mathbf{z} \in \mathcal{Y}^T} \exp(\mathbf{w}^\top \Phi(\mathbf{z}, \mathbf{x})) \right)$$

- Second term is equal to $\log Z$ (from forward-backward)
- Computed using the forward algorithm

- Do computation in log space!
  - Replace multiplication by addition
  - Replace addition by logsumexp. To sum n values a_i in log space:

$$m + \log \left(\sum_{i=1}^{n} \exp(a_i - m) \right)$$

where $m = \max_i a_i$

Sequence Modelling

Type of tasks

- Sequence Labelling
- Sequence Classification
- Sequence Generation (lecture 5)
- Topic Models (lecture 6)

A sequence classification task : Sentiment Analysis

Input: Brilliant and moving performances by Tom Courtenay and Peter Finch.[3]

Output: Positive

Annotation scheme
Sentiment labels: {Neutral, Negative, Positive}, {Negative, Positive}, {0,..,5}...

[3] IMDB dataset (Pos,Neg) (2)

Sequence Classification: a bag-of-words approach

Let $(W_1, ...W_T)_i$ be the input sequence, Y_i the label, let X be a pretrained word embedding matrix ($X \in R^{Vd}$).

- $X_{w_1}, ..., X_{w_T}$ the embedded sequence
- Compute a sentence vector representation with $s = \frac{1}{T}\sum_j X_{w_j}$ ($\in \mathrm{R}^d$)
- Train a logistic regression on $\{s_i, y_i\}_{1...n}$

"Bag-of-words" because we do not model the tokens positions in the sequence.

Evaluation

Evaluating classification tasks

- **Classification tasks** (token level like sequence labelling, or sequence level like sequence classification) are evaluated with *confusion matrix-based metrics*[4].
- Define class(es) of interest as *Positive*
- Count Correct Prediction (True Positive/Negative) and Incorrect Prediction (False Positive Prediction and False Negative)
- Aggregate those numbers with statistics: Accuracy, F1[5]

Example:

- POS tagging: Accuracy
- NER: F1 score with regard to the name entities
- Sentiment Classification: F1 score (usually)

4

5 https://en.wikipedia.org/wiki/$F1_score$

Evaluating classification tasks

Confusion Matrix

	Prediction		
	TP: True Positive	FN: False Negative	P′
Observations	FP: False Positive	TN: True Negative	N′
total	P	N	

Evaluation metrics for Classification

$$Accuracy = \frac{TP + TN}{P + N}$$

$$Precision = \frac{TP}{P'} \text{ and } Recall = \frac{TP}{P}$$

$$F1 = hmean(Precision, Recall) = (\frac{Precision^{-1} + Recall^{-1}}{2})^{-1}$$

- Accuracy is relevant if the classes are balanced and if no specific care should be given to the performance on a specific class
- Otherwise: F1-score (or more generally ROC curve based scores) should be used

The Modelling Challenge

All NLP tasks require a sequence model i.e an estimation of

$$P(t_1, .., t_n | x_1, .., x_n)$$

So far, we have studied one solution (e.g. MEMMs):

1. Modelling in terms of probability the problem
2. Making assumption on the distribution to simplify the joint distribution
3. Estimate it with data

In lecture 4, we will study how Deep Learning brings a more powerful solution for this problem

Lecture Summary

- Preprocessing (encoding, segmentation)
- Modelling sequence tagging tasks (such as POS or NER) with MEMMS and CRF model
- Modelling Sentiment Analysis with a *bag-of-words* model
- Evaluation

References I

Armand Joulin & Edouard Graves, ENSAE ML for NLP, 2019

[Gerdes] Gerdes, Kim; Kahane, S. Y. C. E. A. C. M. C. French spoken treebank. In *github.com/UniversalDependencies/UD_French − Spoken*.

[2] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.