Language Understanding

03 - Introduction to Natural Language Processing

Hossein Zeinali

Agenda

- Text Normalization
- Tokenization
- Language Models
    - DNN Based Language Models
- Machine Translation
- POS Tagging
- Named Entity Recognition
- Sequence Tagging and Labeling

Text Normalization

- Every NLP task needs to do text normalization:
  - Segmenting/tokenizing words in running text
  - Normalizing characters especially in Persian
    - E.g. different types of "ی" and "ک"
  - Normalizing word formats
    - How should we process capitalized words?
    - How about inflected forms like *cats* versus *cat*?
      - What about morphologically complex languages like Arabic/Persian?
  - Segmenting sentences in running text

- Text-normalization in Text-to-Speech:
  - Transforming text into a single canonical form: "$200" => "two hundred dollars"

How many words?

- Sara's **cat** in the hat is different from other **cats**!
  - Lemma: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - Word-form: the full inflected surface form
    - cat and cats = different word-forms
- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- E.g. "They picnicked by the pool, then lay back on the grass and looked at the stars."
    - 16 tokens and 14 types

Word Tokenization

- Convert a running text to a sequence of tokens
  - Tokens: words, punctuations

- Penn Treebank tokenization example:
  - **Input**: "The San Francisco-based restaurant," they said, "doesn’t charge $10".
  - **Output**: " | The | San | Francisco-based | restaurant | , | “ | they | said | , | “ | does | n’t | charge | $ | 10 | “ | .

Issues in Tokenization

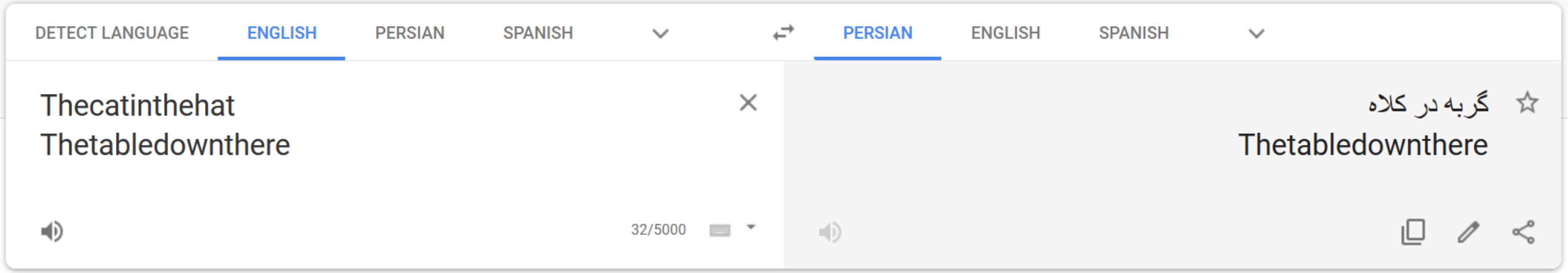
- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??
- What about Persian?

Maximum Matching Word Segmentation

- Given a wordlist and a string.
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer
  3. Move the pointer over the word in string
  4. Go to 2
- Examples:
  - Thecatinthehat
    - the cat in the hat
  - Thetabledownthere
    - the table down there
    - theta bled own there

Maximum Matching Word Segmentation

- Given a wordlist and a string.
  1. Start a pointer at the beginning of the string
  2. Find the longest word in dictionary that matches the string starting at pointer



    - the cat in the hat
  - Thetabledownthere
    - the table down there
    - theta bled own there

Probabilistic Language Models

Why Language Modeling?

- Goal: assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds tonight) > P(**large** winds tonight)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling?

- Goal: compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, w_4, w_{5,} \dots, w_n)$$

- Related task: probability of an upcoming word or next-word prediction:

$$P(w_5 | w_1, w_2, w_3, w_4)$$

- A model that computes either $P(W)$ or $P(w_n | w_1, w_2, \dots, w_{n-1})$ is called a **language model.**

How to Compute the Probability?

- The Chain Rule:
  $P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \dots P(x_n|x_1, \dots, x_{n-1})$

- In language modeling:

$$P(w_1 w_2 \dots w_n) = P(w_1) \prod_{i=2}^{n} P(w_i|w_1 \dots w_{i-1})$$

- How to estimate these probabilities?
  - Could we just count and divide?
    - No! Too many possible sentences!
    - We'll never see enough data for estimating these

How to Compute the Probability?

- Markov Assumption: simplifying the equation based on the order

$$P(w_1 w_2 \dots w_n) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-k} \dots w_{i-1})$$

- In other words, we approximate each component in the product:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

- Unigram:

$$P(w_1 w_2 \dots w_n) = \prod_{i=1}^{n} P(w_i)$$

- Bigram:

$$P(w_1 w_2 \dots w_n) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-1})$$

- N-gram models: trigrams, 4-grams, 5-grams
  - Now we can use counting for estimation

Evaluation of N-gram Models

- **Extrinsic** evaluation:
  - Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B
- **Intrinsic** evaluation: perplexity
  - Is a good approximation if the test data looks **just like** the training data

Perplexity

- The best language model is one that best predicts an unseen test set
  - Gives the highest P(sentence)
- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_n)^{-\frac{1}{n}} = \sqrt[n]{\frac{1}{P(w_1 w_2 \dots w_n)}}$$

- Using chain rule:

$$PP(W) = \sqrt[n]{\prod_i \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

- **Minimizing perplexity is the same as maximizing probability**

Problems: Zero in Probabilities

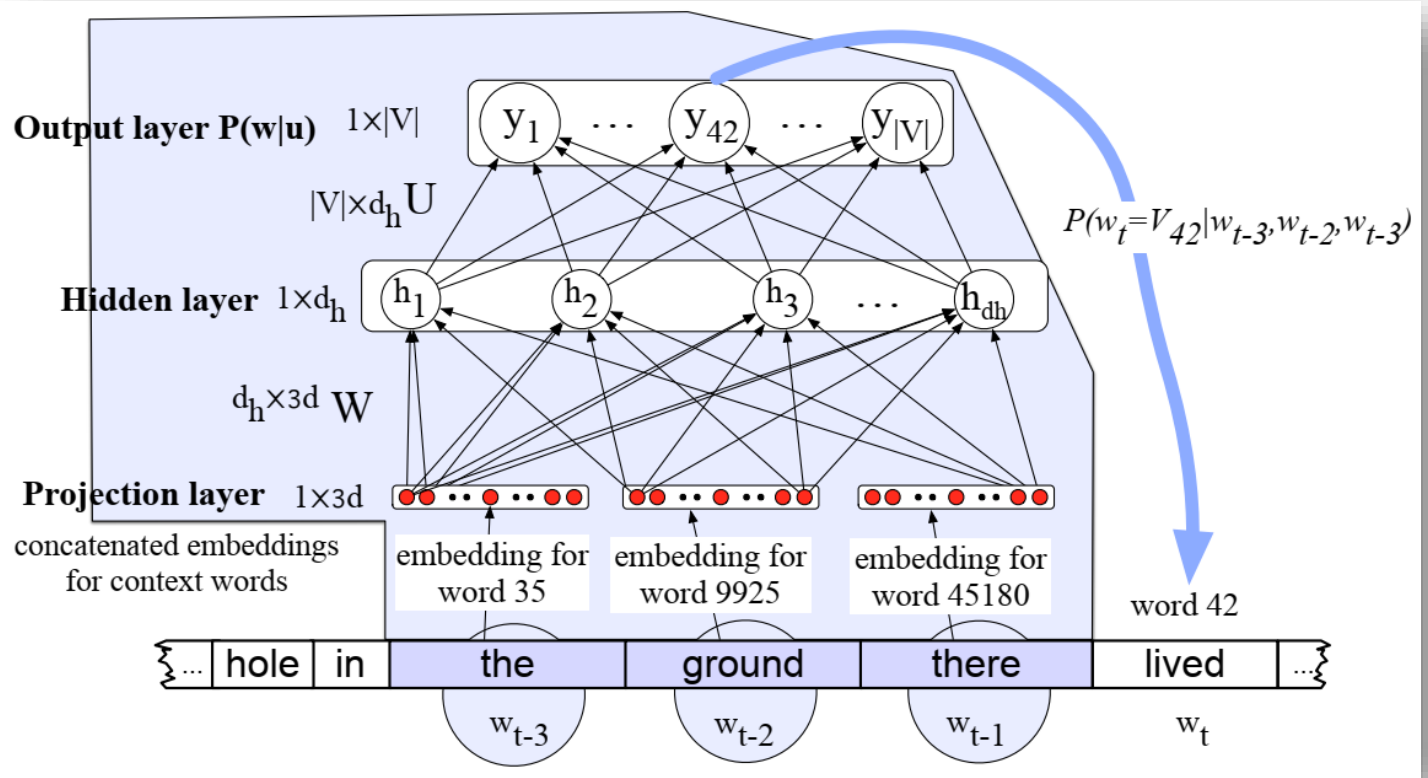
- Shakespeare as corpus:
  - N=884,647 tokens, V=29,066
  - Shakespeare produced 300,000 bigram types out of V2= 844 million possible bigrams.
    - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
  - Bigrams with zero probability:
    - Mean that we will assign 0 probability to the test set!
    - Hence we cannot compute perplexity (can't divide by 0)!
- Solutions:
  - **Add-one estimation (Laplace smoothing):** Pretend we saw each word one more time than we did. Just add one to all the counts!
  - **Backoff**: use trigram if you have good evidence, otherwise bigram, otherwise unigram
  - **Interpolation** between unigram, bigram, trigram (weighted sum)
  - And more advanced methods like Kneser-Ney Smoothing
- How to deal with out-of-vocabulary (OOV) words?

Neural Language Models (LM)

- **Language modeling**: predicting upcoming words from prior word context.
- Advantages over N-gram LM:
  - Don't need smoothing
  - Can handle much longer histories
  - Can generalize over contexts of similar words
  - Underlie many of the models in NLP
- Feedforward neural LM:
  - A standard feedforward network that takes as input at time t a representation of some number of previous words (w_{t-1}, w_{t-2}, etc.) and outputs a probability distribution over possible next words.

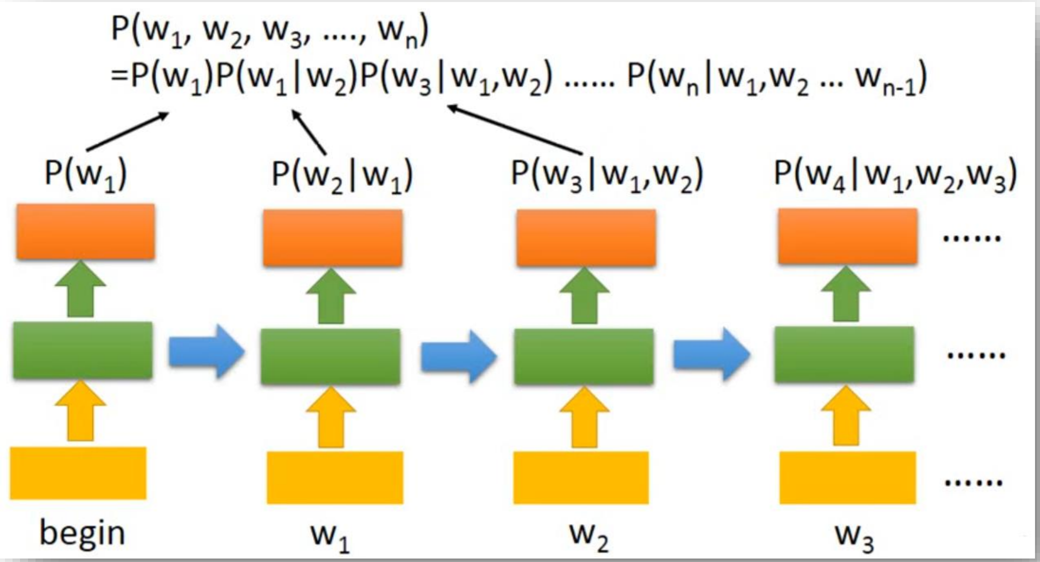
$$P(w_t | w_1^{t-1}) \approx P(w_t | w_{t-N+1}^{t-1})$$

Neural Language Models (LM)

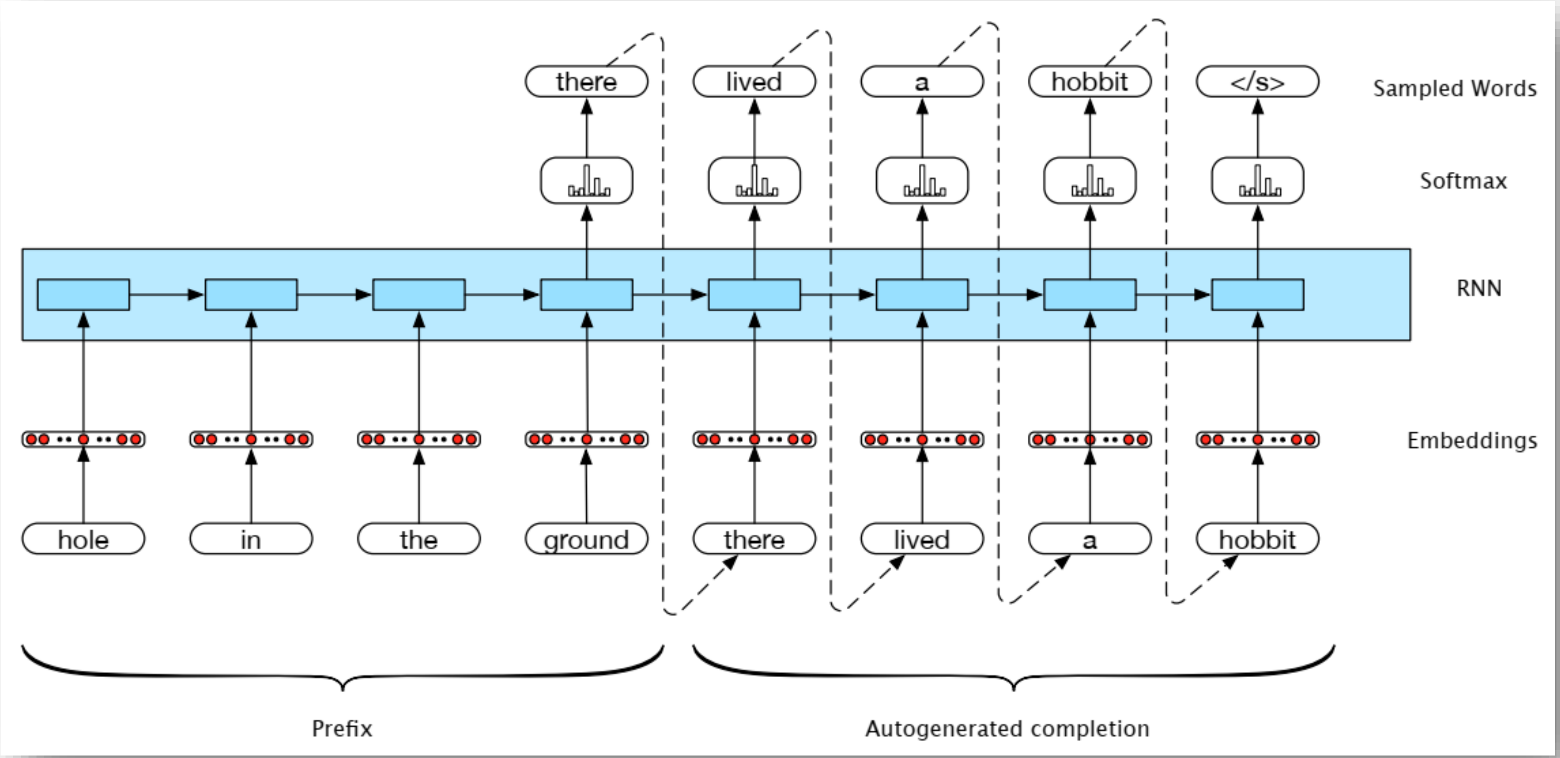


RNN for Language Modeling

- Context is important in language modeling:
  - But n-gram language models use a fixed context window.
  - Feedforward networks also use a fixed context window.
- Solution:
  using RNNs



Using an RNN to generate Text



Machine Translation

Conditional Language Models

- There are many applications where we want to predict words conditioned on some input:
  - Speech recognition: condition on speech signal
  - Machine translation: condition on text in another language
  - Text completion: condition on the first few words of a sentence
  - Optical character recognition: condition on an image of text
  - Image captioning: condition on an image
  - Grammar checking: condition on surrounding words

Machine Translation Problem

- MT is a sequence to sequence transform
- The translation problem can be described as modeling the probability distribution $P(E|F)$, where F is a string in the source language and E is a string in the target language.
- Using Bayes' Rule, this can be rewritten

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)} \cong P(F|E)P(E)$$

- $P(F|E)$ is called the "translation model" (TM). $P(E)$ is called the "language model" (LM).
    - The LM should assign probability to sentences which are "good English".

Other NLP Applications

POS Tagging

- We are interesting to syntactic word classes that called Part of Speech (POS)
- What is a word class?
  - Words that somehow ‘behave’ alike:
    - Appear in similar contexts
    - Perform similar functions in sentences
    - Undergo similar transformations
- Why do we want to identify them?
  - Pronunciation (mard/mord in Persian)
  - Stemming
  - Semantics
  - More accurate Language Models
  - Simple syntactic information

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Bijankhan Tagset

ADJ	Adjective, General	MQUA	Modifier of Quantifier
ADJ_CMPR	Adjective, Comparative	MS	Mathematic Symbol
ADJ_INO	Past Participle	N_PL	Noun, Plural
ADJ_ORD	Adjective, Ordinal	N_SING	Noun, Singular
ADJ_SIM	Adjective, Simple	NN	Number
ADJ_SUP	Adjective, Superlative	NP	Noun Phrase
ADV	Adverb, General	OH	Oh Interjection
ADV_EXM	Adverb, Exemplar	OHH	Oh noun
ADV_I	Adverb, Question	P	Preposition
ADV_NEGG	Adverb, Negation	PP	Prepositional Phrase
ADV_NI	Adverb, Not Question	PRO	Pronoun
ADV_TIME	Adverb, Time	PS	Psedo-Sentence
AR	Arabic Word	QUA	Quantifier
CON	Conjunction	SPEC	Specifier
DEFAULT	Default	V_AUX	Verb, Auxiliary
DELM	Delimiter	V_IMP	Verb, Imperative
DET	Determiner	V_PA	Verb, Past Tense
IF	Conditional	V_PRE	Verb, Predicative
INT	Interjection	V_PRS	Verb, Present Tense
MORP	Morpheme	V_SUB	Verb, Subjunctive

Approaches to POS Tagging

- Rule-based Approach
  - Uses handcrafted sets of rules to tag input sentences
- Statistical approaches
  - Use training corpus to compute probability of a tag in a context
- Hybrid systems
  - Is based on rules while rules are automatically induced from hand-tagged data

Statistical POS Tagging

- Goal: choose the best sequence of tags T for a sequence of words W in a sentence:

$$T' = \underset{T}{\operatorname{argmax}}\, P(T|W)$$

- By Bayes Rule (giving us something easier to calculate)

$$P(T|W) = \frac{P(W|T)P(T)}{P(W)}$$

- Since we can ignore $P(W)$, we have

$$T' = \underset{T}{\operatorname{argmax}}\, P(W|T)\, P(T)$$

Statistical POS Tagging

- By using the Chain Rule:

$$P(T) = \prod P\left(t_i \middle| t_1^{i-1}\right)$$

  - Markov assumption:

$$P(T) \approx \prod P\left(t_i \middle| t_{i-N+1}^{i-1}\right)$$

- Using the Chain rule for $P(W|T)$:

$$P(W|T) = \prod P\left(w_i \middle| w_1^{i-1} t_1^i\right)$$

- Simplifying assumption: probability of a word depends only on its own tag $P(w_i|t_i)$

$$P(W|T) \approx \prod P(w_i|t_i)$$

Named Entity (NE) Recognition

- Why do NE Recognition?
  - Key part of Information Extraction system
  - Robust handling of proper names essential for many applications
  - Pre-processing for different classification levels
  - Information filtering
  - Information linking

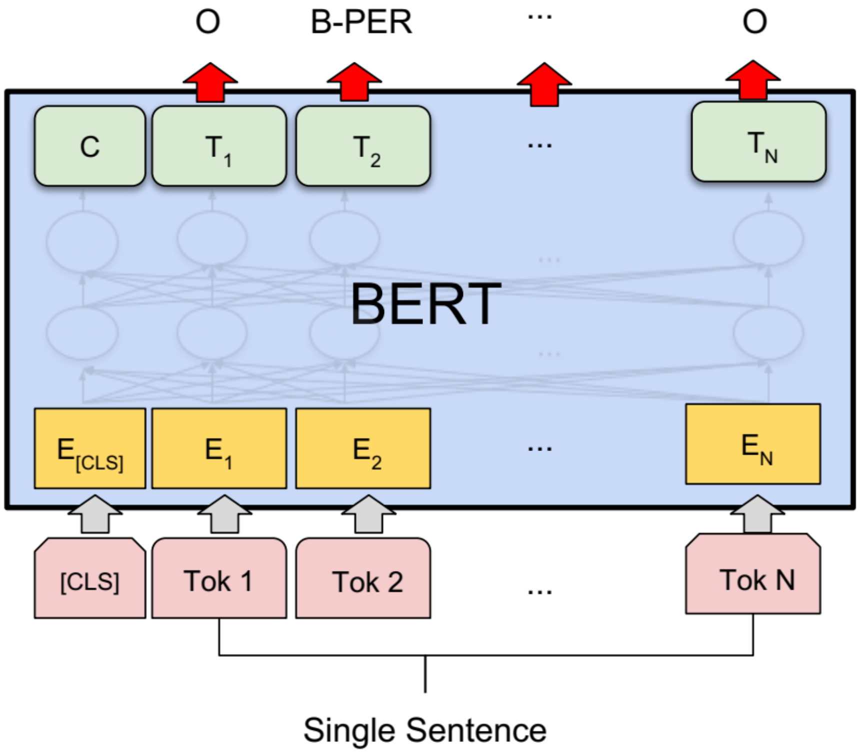
Named Entity Definition

- NE involves **identification** of proper names in texts, and **classification** into a set of predefined categories of interest.
- Three universally accepted categories: **person**, **location** and **organization**
- Other common tasks: recognition of date/time expressions, measures (percent, money, weight etc), email addresses etc.
- Other domain-specific entities: names of drugs, medical conditions, names of ships, bibliographic references etc.
- Category definitions are intuitively quite clear, but there are many grey areas.
- Many of these grey area are caused by **metonymy**.

Problems in NER

- Variation of NEs – e.g. John Smith, Mr Smith, John.
- Ambiguity of NE types
  - John Smith (company vs. person)
  - May (person vs. month)
  - Washington (person vs. location)
  - 1945 (date vs. time)
- Ambiguity with common words, e.g. “may”
- Issues of style, structure, domain, genre etc.
  - Punctuation, spelling, spacing, formatting, ...all have an impact

NER Using BERT



Thanks for your attention

References and IP Notice

- Daniel Jurafsky and James H. Martin, “Speech and Language Processing”, 3rd ed., 2019
- Some graphics were selected Slidesgo template