

CSC485/2501 A2

TA: Ruiyu Wang

Assignment 2

- Now available!
- Due 17.00 Monday, Nov 3.

Assignment 2

- Word Sense Disambiguation (q0, q1, q2) <- this tutorial!
- LLM causal tracing (q3)
- After A2, you will be familiar with...
 - NLTK
 - WordNet
 - The Lesk algorithm
 - Word embeddings methods such as word2vec and BERT
 - How LLMs work internally when prompting

Word Sense Disambiguation

Given a word in a sentence, find the correct sense of it

- A word can have a lot of different meanings!

Naturally: breaking down sentences into words and use word processing methods to do that

- NLTK and WordNet!

WordNet and NLTK

WordNet provides some useful functions for retrieving word senses and other information.

```
>>> from nltk.corpus import wordnet as wn
```

Synset basic: use `synsets()` to look up a word, its hypernyms and hyponyms

```
>>> wn.synsets('dog')
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'),
 Synset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]
```

Hypernyms have broader meanings than the current word; hyponyms are more specific

```
>>> wn.synsets('dog')[0].hypernyms()
[Synset('canine.n.02'), Synset('domestic_animal.n.01')]
>>> wn.synsets('dog')[0].hyponyms()
[Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), Synset('dalmatian.n.02'),
 Synset('great_pyrenees.n.01'), Synset('griffon.n.02'), Synset('hunting_dog.n.01'),
 Synset('lapdog.n.01'), Synset('leonberg.n.01'), ... (rest omitted)]
```

WordNet and NLTK

WordNet provides a simple **tokenizer** that splits word sequences into discrete tokens (which can be used by various nlp methods for computation)

```
>>> from nltk.tokenize import word_tokenize
>>> word_tokenize('this is an example sentence!')
['this', 'is', 'an', 'example', 'sentence', '!']
```

*Note that modern language models have different tokenization algorithms, instead of simply splitting sentences into words

- \$3.88: ['\$3.88'] or ['\$', '3', '.', '88']?
- sometimes: ['sometimes'] or ['some', 'times']?
- We will cover this part later in this tutorial!

WordNet and NLTK

Stopwords are the common words that are often filtered out for text processing.

```
>>> from nltk.corpus import stopwords
```

```
>>> stopwords.words('english')
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',  
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
```

```
.....
```

```
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

*Why do we remove them?

**Do we want them to be removed in all the scenarios? ()

WSD: The Lesk Algorithm

Algorithm 1: The simplified Lesk algorithm.

input : a word to disambiguate and the sentence in which it appears

best_sense \leftarrow most_frequent_sense word

best_score \leftarrow 0

context \leftarrow the bag of words in sentence

for each sense of word **do**

 signature \leftarrow the bag of words in the definition and examples of sense

 score \leftarrow Overlap(signature, context)

if score > best_score **then**

 best_sense \leftarrow sense

 best_score \leftarrow score

end

end

return best_sense

Scoring

- Overlap(): count the intersections!
- What is the intersection?
 - The bag of words as context and signatures
 - Why bag of words? (frequency matters)
- Can we improve the algorithm?

Scoring

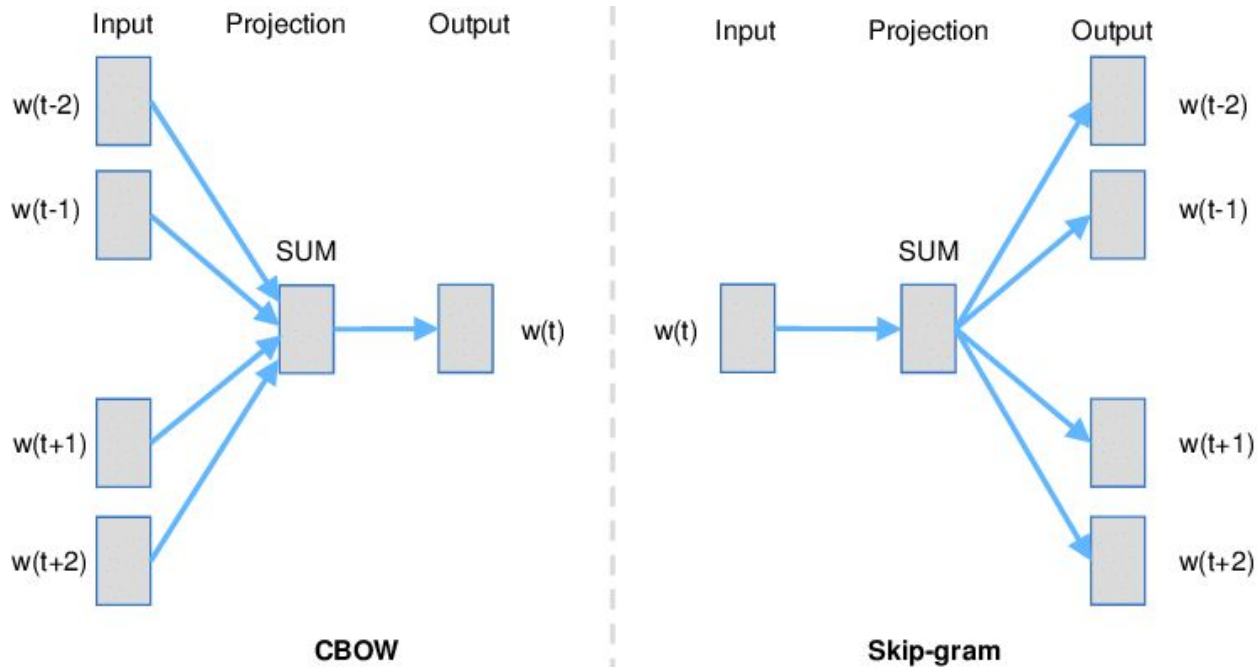
- Overlap(): count the intersections!
- What is the intersection?
 - The bag of words as context and signatures
 - Why bag of words? (frequency matters)
- Can we improve the scoring?
 - Yes! Using word vectors!
 - Represent the surrounding using numerical representations
 1. Count vectors via BOW representations
 $\{a, a, b, a\} \rightarrow [3, 1, 0]$, $\{a, b, c\} \rightarrow [1, 1, 1]$
 2. Pretrained embeddings (e.g. **word2vec**)
 - Compute the score using vector similarity metrics
 - Euclidean distance, Cosine similarity, etc.

Word2vec

Efficient Estimation of Word Representations in Vector Space. NeurIPS 13'.

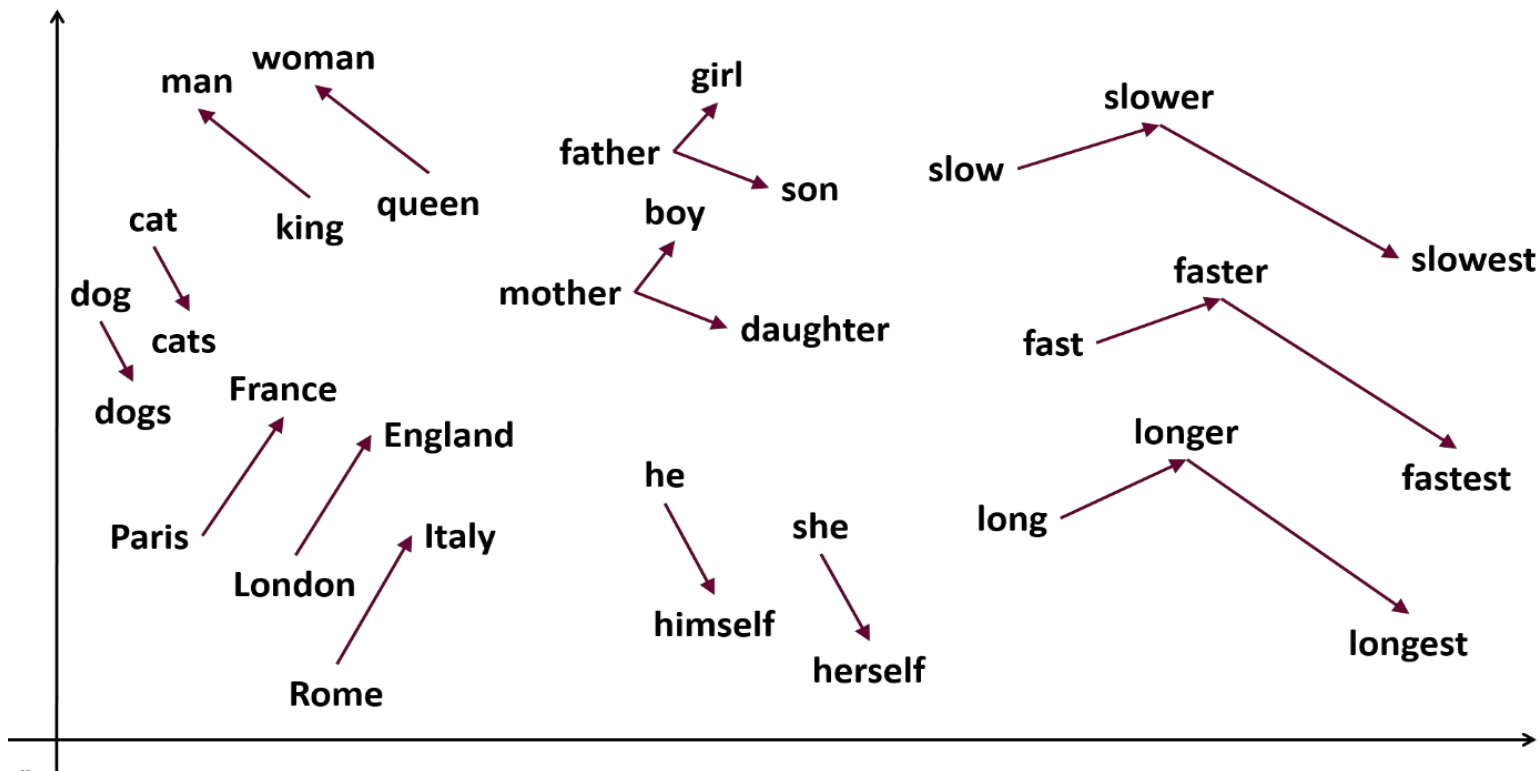
- Pretrained word vectors from large amount of text ('pretrained' means you don't have to train this again in this course).
- Words are mapped to vectors of real numbers.
 - `word2vec(word:str)->vector:np.array()`
- Instead of counting bag of word entries, we average the word vectors of each of them to create a new representation for the context/signatures

Word2vec



Mainstream word2vec embeddings are trained with Skip-Gram with Negative Sampling (SGNS)

Word2vec



Word2vec

Efficient Estimation of Word Representations in Vector Space. NeurIPS 13'.

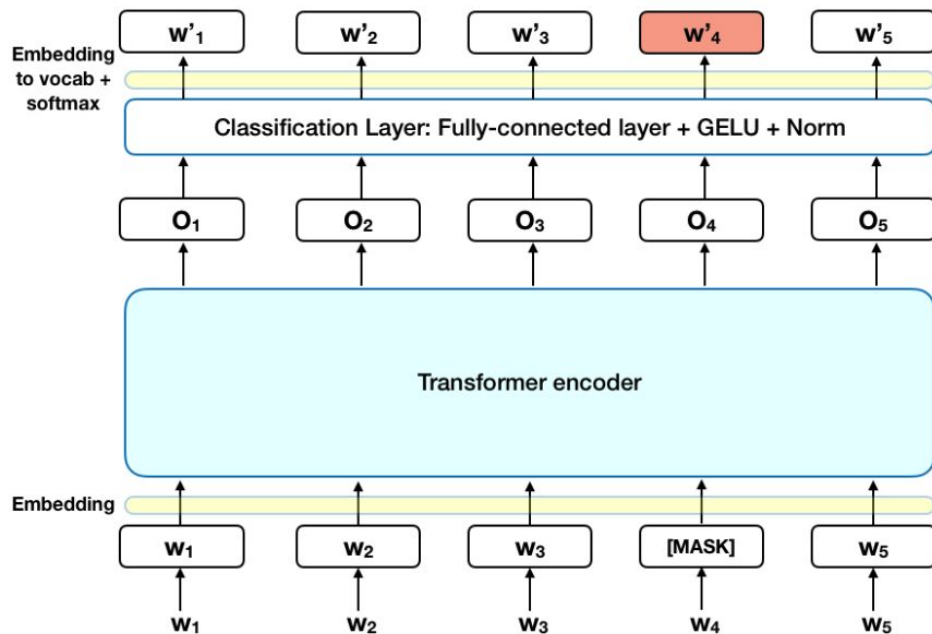
- Pretrained word vectors from large amount of text ('pretrained' means you don't have to train this again in this course).
- Words are mapped to vectors of real numbers.
 - `word2vec(word:str)->vector:np.array()`
- Instead of counting bag of word entries, we average the word vectors of each of them to create a new representation for the context/signatures
- Each word only has one **fixed** vector representation, although it could have multiple senses.

Fixed vs. Contextual Embedding

	Source	Nearest Neighbors
Fixed	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
Contextual	Chico Ruiz made a spectacular <u>play</u> on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent <u>play</u> .
	Olivia De Havilland signed to do a Broadway <u>play</u> for Garson {...}	{...} they were actors who had been handed fat roles in a successful <u>play</u> , and had talent enough to fill the roles competently , with nice understatement .

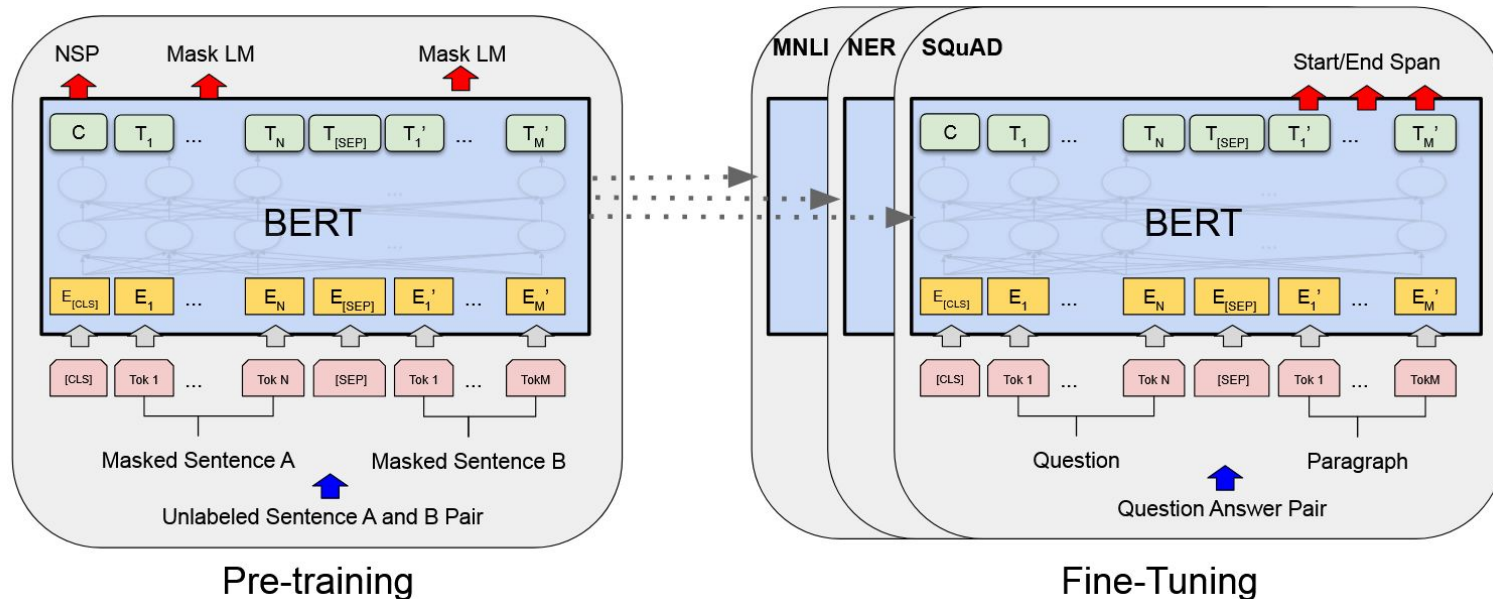


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 19'





Pre-trained on two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)



BERT Tips

1. Tokenize your sentences before send them to BERT
BERT does not take string inputs!
2. Be careful with your BERT tokens and how they corresponds to the initial sentence

BERT tokenizer does not simply split sentences into words

```
>>> out = tknz([[ 'Multiple', ',', 'pre-tokenized', 'sentences', '!']], ...)
```

```
>>> out.tokens(0)
```

```
[[ 'CLS'], 'Multiple', ',', 'pre', '-', 'token', '##ized', 'sentences', '!', '[SEP]']
```

3. In coding: do **not** loop tensors!

```
for (int i = 0; i < M; ++i)
```

```
    for (int j = 0; j < N; ++j)
```

```
        for (int k = 0; k < K; ++k)
```

```
            C[i][j] += A[i][k] * B[k][j];
```

Too slow!!!

Use torch's built-in functions instead (e.g. torch.mm). Remember to keep track on dimensions

GLHF!

If you have questions:

- Ask now!
- Post them on piazza

*The handout will be posted on the course website