



CS 540 Introduction to Artificial Intelligence

Natural Language Processing

University of Wisconsin-Madison
Fall 2026 Sections 1 & 2

Announcements

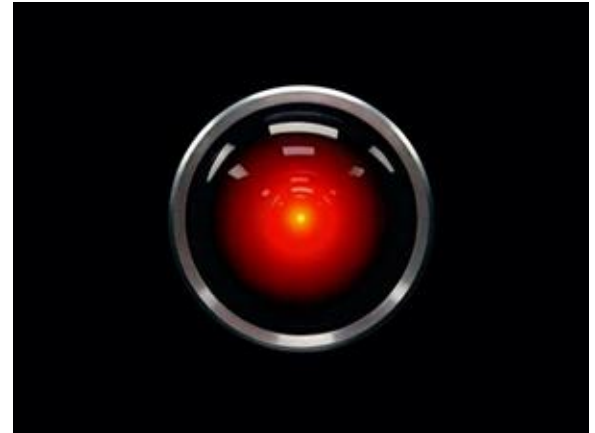
- **HW 1 online:**
 - Due on **Friday, September 18 at 11:59PM**
- **Midterm:**
 - October 23, 5:45PM-7:15PM in Humanities 2340
 - Alternate time the day before; **reach out now to arrange this**
- Discussion Section: **Tuesdays 4-5:30 in MH 3610**
- Class roadmap:

NLP
Machine Learning: Introduction
Machine Learning: Unsupervised Learning

What is **NLP**?

Combining computing with human language. Want to:

- Answer questions
- Summarize or extract information
- Translate between languages
- Generate dialogue/language
- Write stories automatically



Why is it **hard**?

Many reasons:

- Ambiguity: “*Mary saw the duck with the telescope in the park*”. Several meanings.
- Understanding of the world
 - “Bob and Joe are fathers”.
 - “Bob and Joe are brothers”.



Approaches to NLP

A brief history

- Symbolic NLP: 50's to 90's
- Statistical/Probabilistic: 90's to present
 - Neural nets: 2010's to present
 - Large Language Model (LLM): GPT etc.

Lots of progress!

Lots more to work to do



ELIZA program

Outline

- Introduction to language models
 - n-grams, training, evaluation, generation
- Bag of Words (BOW) and TF-IDF
- Word representations
 - One-hot, word embeddings, transformer-based

Language Models

- Basic idea: use probabilistic models to **assign a probability to a sentence W**

$$P(W) = P(w_1, w_2, \dots, w_n) \text{ or } P(w_{\text{next}} | w_1, w_2 \dots)$$

- w_i may be letters or words
- Goes back to Shannon
 - Information theory: letters

Zero-order approximation	XFOML RXKHRJFFJUJ ALPWXFJWXYJ FFJEYVJCQSGHYD QPAAMKBZAACIBZLKJQD
First-order approximation	OCRO HLO RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL
Second-order approximation	ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE
Third-order approximation	IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE
First-order word approximation	REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE

Training The Model

Recall the chain rule of probability:

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2|w_1) \dots P(w_n|w_{n-1} \dots w_1)$$

- How do we estimate these probabilities? That is, how do we train our model (in ML language)?
 - From data (text corpus)
 - Can't estimate reliably for long *histories*.

Training: Make Assumptions

- Markov assumption with shorter history:

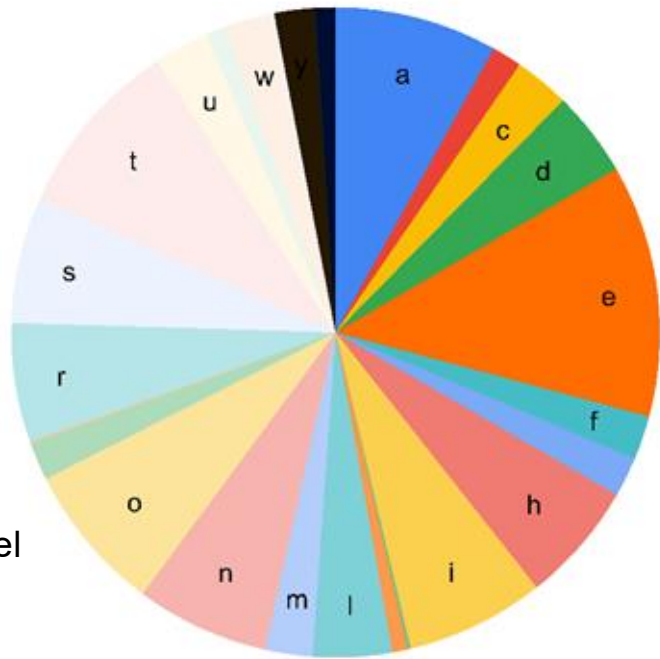
$$P(w_i | w_{i-1} w_{i-2} \dots w_1) = P(w_i | w_{i-1} w_{i-2} \dots w_{i-k})$$

- Present doesn't depend on whole past
 - Just recent past, i.e., *context*.
 - What's ***k=0?***

k=0: **Unigram** Model

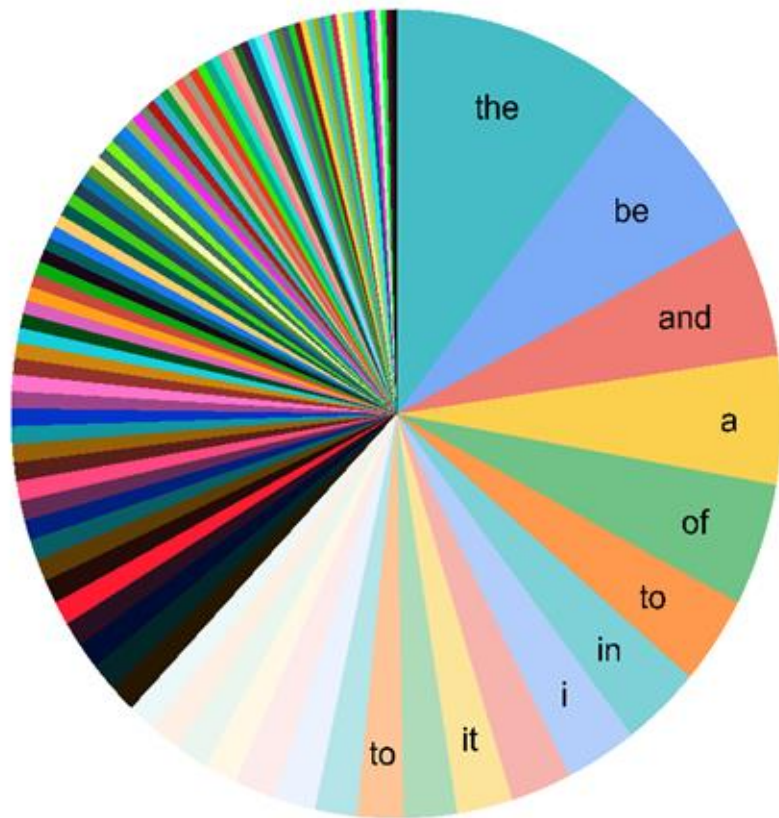
- Full independence assumption:
 - (Present doesn't depend on the past)

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2) \dots P(w_n)$$



The English letter frequency wheel

Unigram word model



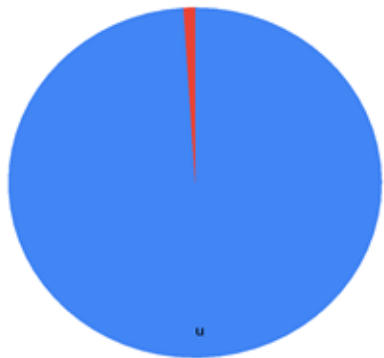
Example (from Dan Jurafsky's notes)

fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass thrift, did, eighty, said, hard, 'm, july,
that, or, limited, the

k=1: Bigram Model (a.k.a Markov chain)

- Markov Assumption:
 - (Present depends on immediate past)

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_n|w_{n-1})$$



$p(.|q)$: the “after q” wheel



$p(.|j)$: the “after j” wheel

texaco, rose, one, in, this, issue,
is, pursuing, growth, in, a, boiler,
house, said, mr., gurria, mexico, 's,
motion, control, proposal, without,
permission, from, five, hundred,
fifty, five, yen outside, new, car,
parking, lot, of, the, agreement,
reached this, would, be, a, record,
november

k=n-1: **n**-gram Model

Can do trigrams, 4-grams, and so on

- More expressive as n goes up
- Harder to estimate

Training: just count? I.e, for bigram:

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

Example: Training in a bigram model

Corpus (3 sentences):

<start> the cat sat <end>

<start> the dog sat <end>

<start> the cat ran <end>

Bigram probabilities, $P(w_2 \mid w_1) = \text{count}(w_1 w_2) / \text{count}(w_1)$:

$P(\text{the} \mid \text{<start>}) = 3/3 = 1$

$P(\text{cat} \mid \text{the}) = 2/3$

$P(\text{dog} \mid \text{the}) = 1/3$

$P(\text{sat} \mid \text{cat}) = 1/2$

$P(\text{ran} \mid \text{cat}) = 1/2$

$P(\text{sat} \mid \text{dog}) = 1/1 = 1$

$P(\text{<end>} \mid \text{sat}) = 2/2 = 1$

$P(\text{<end>} \mid \text{ran}) = 1/1 = 1$

$P(w_2 \mid w_1) = 0$ for every other word pair (e.g. $P(\text{ran} \mid \text{dog}) = 0$)

Example: Training in a bigram model (cont'd)

Probability of a seen sentence:

$P(<\text{start}> \text{ the cat sat } <\text{end}>)$

$= P(\text{the} | <\text{start}>) * P(\text{cat} | \text{the}) * P(\text{sat} | \text{cat}) * P(<\text{end}> | \text{sat})$

$= 1 * 2/3 * 1/2 * 1 = 1/3$

Probability of an unseen sentence:

$P(<\text{start}> \text{ the dog ran } <\text{end}>)$

$= 1 * 1/3 * P(\text{ran} | \text{dog}) * 1 = 1 * 1/3 * 0 * 1 = 0$

"the dog ran" is grammatical, but $\text{count}(\text{dog ran}) = 0$.

A single zero annihilates the whole product.

n-gram Training

Issues:

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

- **1.** Multiply tiny numbers?
 - **Solution:** use logs; add instead of multiply
- **2.** n-grams with zero probability?
 - **Solution:** smoothing

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i) + 1}{\text{count}(w_{i-1}) + V}$$

Simple data generation from letter bigram

Writing = sampling

- Say we start with q
- Sample from $P(\cdot | q)$: spin the “after q” wheel  , we get u
- Sample from $P(\cdot | u)$: spin the “after u” wheel, say we get e
- Sample from $P(\cdot | e)$: spin the “after e” wheel, say we get r
- ...

Sampling Shakespeare unigram LM

- To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
- Every enter now severally so, let
- Hill he late speaks; or! a more to leg less first you enter
- Will rash been and by I the me Loves gentle me not slavish page, the and hour; ill let
- Are where exeunt and sighs have rise excellency took of .. sleep knave we. near; vile like

Sampling Shakespeare bigram LM

- What means, sir. I confess she? then all sorts, he is trim, captain.
- Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
- What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?
- Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt

Sampling Shakespeare trigram LM

- Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
- This shall forbid it should be branded, if renown made it empty.
- What ist that cried?
- Indeed the duke; and had a very good friend.

Evaluating Language Models

How do we know we've done a good job?

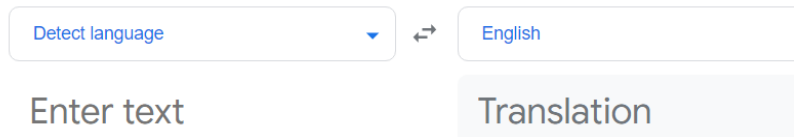
- Observation
- Train/test on separate data & measure metrics
- **Metrics:**
 - 1. Extrinsic evaluation
 - 2. Perplexity



Extrinsic Evaluation

How do we know we've done a good job?

- **Pick a task** and use the model to do the task
- For two models, M_1 , M_2 , compare the accuracy for each task
 - **Ex:** Q/A system: how many questions right. Translation: how many words translated correctly
- Downside: slow, relies on an external task, some models may do well on one but not another



Detect language ▼ ↔ English

Enter text Translation

Intrinsic Evaluation: Perplexity

Perplexity is a **measure of uncertainty**

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

Compute average $PP(W)$ for all W from a dataset

Lower is better! Examples:

- WSJ corpus; 40 million words for training:
 - Unigram: 962, Bigram 170, Trigram 109

Further NLP Tasks

Language modeling is **not the only NLP task**:

- Part-of-speech tagging, parsing, etc.
- Question-answering, translation, summarization, classification (e.g., sentiment analysis), generation, etc.

Representing Documents as Vectors via Bag of Words (BOW)

Document 1

Madison is the capital city of Wisconsin. The city is known for its beautiful lakes and the university.

Document 2

The city sits between two lakes. Madison is beautiful in the summer and the lakes are popular.

Corpus

VOCABULARY

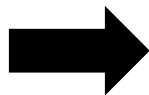
and, are, beautiful, between,
capital, city, for, in, is, its,
known, lakes, Madison, of,
popular, sits, summer, the,
two, university, Wisconsin

Bag of Words (BOW)

Document 1

Madison is the capital city of Wisconsin. The city is known for its beautiful lakes and the university.

$count(w_i)$



1	and
0	are
1	beautiful
0	between
1	capital
2	city
1	for
0	in
2	is
1	its
1	known
1	lakes
1	Madison
1	of
0	popular
0	sits
0	summer
3	the
0	two
1	university
1	Wisconsin

$$\frac{count(w_i)}{\sum_{i=1}^n count(w_i)}$$



normalized

0.056	and
0	are
0.056	beautiful
0	between
0.056	capital
0.111	city
0.056	for
0	in
0.111	is
0.056	its
0.056	known
0.056	lakes
0.056	Madison
0.056	of
0	popular
0	sits
0	summer
0.167	the
0	two
0.056	university
0.056	Wisconsin

Bag of Words (BOW)

Document 2

The city sits between two lakes. Madison is beautiful in the summer and the lakes are popular.

$count(w_i)$



1	and
1	are
1	beautiful
1	between
0	capital
1	city
0	for
1	in
1	is
0	Its
0	known
0	lakes
2	lakes
1	Madison
0	of
1	popular
1	sits
1	summer
3	the
1	two
0	university
0	Wisconsin

$$\frac{count(w_i)}{\sum_{i=1}^n count(w_i)}$$



normalized

0.059	and
0.059	are
0.059	beautiful
0.059	between
0.059	capital
0	city
0.059	for
0	in
0.059	is
0.059	Its
0	known
0	lakes
0.118	lakes
0.059	Madison
0	of
0.059	popular
0.059	sits
0.059	summer
0.176	the
0.059	two
0	university
0	Wisconsin

Is every word interesting?

Feature Vectors

- Sparse vectors: many "0"s appear.
- Shared words: "Madison", "beautiful," and "city" appear in both, showing semantic similarity.
- Frequency: The word "the" is the highest frequency

Vector 1

1	and
0	are
1	beautiful
0	between
1	capital
2	city
1	for
0	in
2	is
1	its
1	known
1	lakes
1	Madison
1	of
0	popular
0	sits
0	summer
3	the
0	two
1	university
1	Wisconsin

Vector 2

1	and
1	are
1	beautiful
1	between
0	capital
1	city
0	for
1	in
1	is
0	its
0	known
2	lakes
1	Madison
0	of
1	popular
1	sits
1	summer
3	the
1	two
0	university
0	Wisconsin

Term Frequency – Inversed Document Frequency (TF-IDF)

- Term Frequency (TF_{ij}) : How many times the term i appears in the document j (normalized over the total number of terms in the document j)
 - Bag of Words (BOW)
- Inverse Document Frequency (IDF_{ij}) : How rare a term is in a set of documents.

$$IDF_{ij} = \log\left(\frac{N}{df_i}\right)$$

← Total number of documents in the corpus

↑
number of documents containing the term w_i

Term Frequency – Inversed Document Frequency (TF-IDF)

$TF\text{-}IDF_{ij}$: How important is the term w_i for the document j

$$TF\text{-}IDF_{ij} = TF_{ij} \times IDF_{ij} = TF_{ij} \times \log\left(\frac{N}{df_i}\right)$$

High $TF\text{-}IDF_{ij}$: The term is frequent in *this* document but rare in others

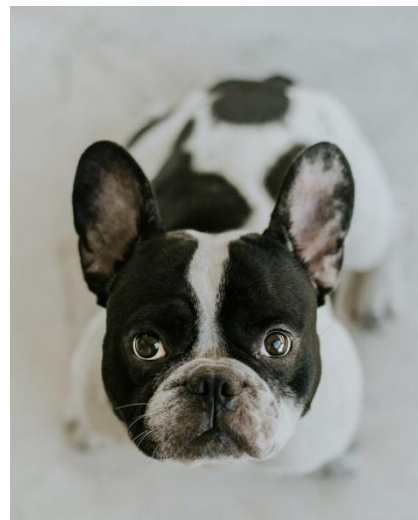
Low $TF\text{-}IDF_{ij}$: The word is either rare in this document or common across *all* documents

Representing Words

Traditional representation: **one-hot vectors**

$$\text{dog} = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$$

- Dimension: # of words in vocabulary
- Relationships between words?



Smarter Representations

Distributional semantics: account for relationships

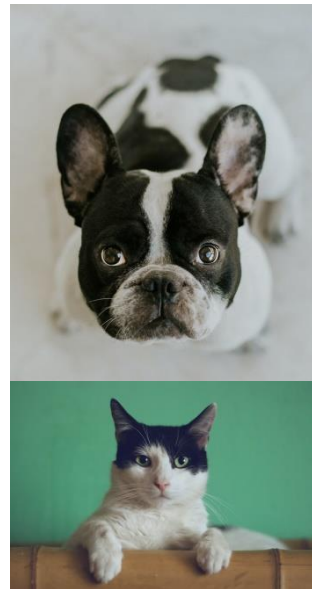
- Reps should be close/similar to other words that appear in a similar context

Dense vectors:

$$\text{dog} = [0.13 \quad 0.87 \quad -0.23 \quad 0.46 \quad 0.87 \quad -0.31]^T$$

$$\text{cat} = [0.07 \quad 1.03 \quad -0.43 \quad -0.21 \quad 1.11 \quad -0.34]^T$$

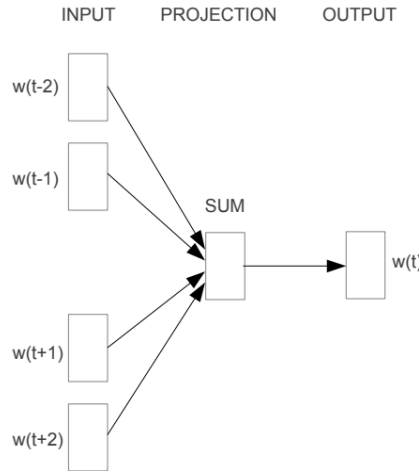
AKA word embeddings



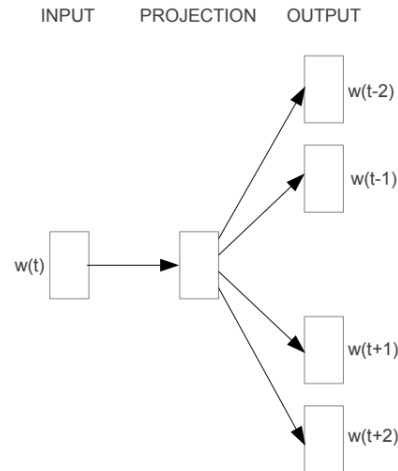
Training Word Embeddings

Many approaches (very popular 2010-present)

- Word2vec: a famous approach

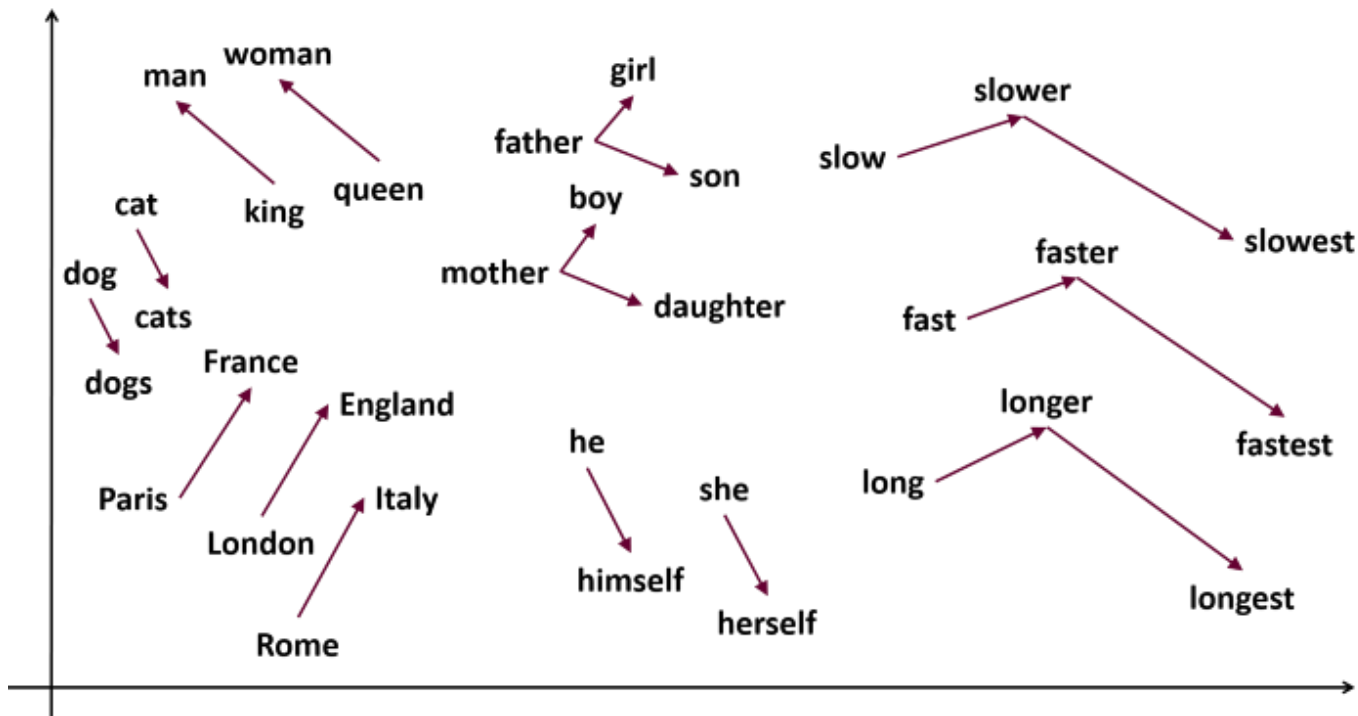


CBOW



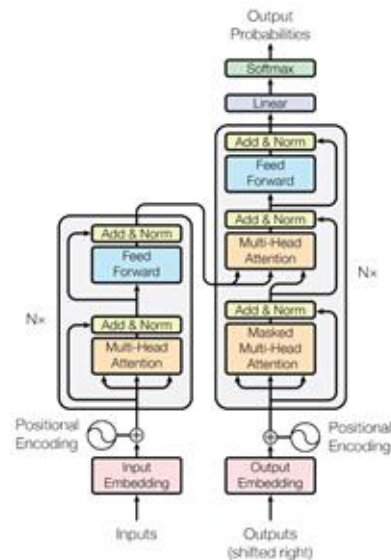
Skip-gram

Word Embeddings



Beyond “Shallow” Embeddings

- Transformers: special model architectures based on **attention**
 - Sophisticated types of neural networks
- Pretrained models
 - Based on transformers: BERT, GPT
 - Include context!
- **Fine-tune** for desired task



Suggested Reading

- Natural Language and Statistics, Notes by Zhu.
<https://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/NLP.pdf>
- Textbook: *Artificial Intelligence: A Modern Approach (4th edition)*.
Stuart Russell and Peter Norvig. Pearson, 2020.
 - Sections 23.1, 23.5, 24.1