AI INSTITUTE #AIISC
UNIVERSITY OF SOUTH CAROLINA

UNIVERSITY OF SOUTH CAROLINA

CSCE 580: Introduction to AI

Lecture 17-20: Text, Language Models

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

15, 22, 24, 29 OCT, 2024

Carolinian Creed: "I will practice personal and academic integrity."

Credits: **Copyrights of all material reused acknowledged**

Organization of Lectures 17-20

- Introduction Segment
  - Recap of Lecture 16
  - Announcement: Graduate paper
- Main Segment
  - Text Processing (Lecture 17)
  - Language Models (LMs) (Lectures 17, 18)
  - Learning for LMs with NN (Lectures 18, 19)
  - Large LMs (19, Project 2)
  - Using LMs - data preparation, evaluation, finetuning (20, Project 2)
- Concluding Segment
  - Course Project Discussion
  - About Next Lecture – Lecture 21
  - Ask me anything

Introduction Section

Graduate Paper Presentation

- Papers between 2022-2024 (last 4 years)
- At top AI venues: AAAI, Neurips, IJCAI, ICML, ICLR, **or discuss with instructor**

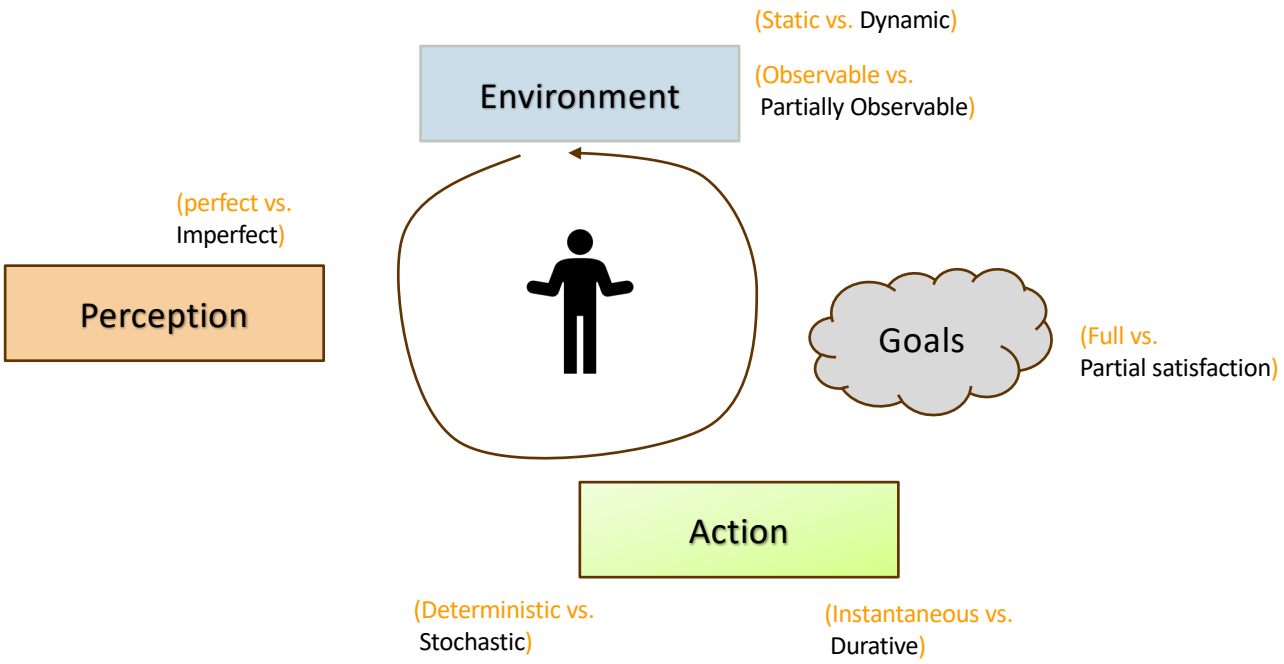
- Guideline on presentation – Nov 22, 2024 [Undergrads to attend]
  - Summary of the paper
  - Critique (+ves/ -ves)
  - Relevance to your and anyone else's project in the class

- Guidelines on a writeup
  - Verbalization of the presentation with three parts: summary, critique and relevance to class projects
  - A running example (from the paper or your own)

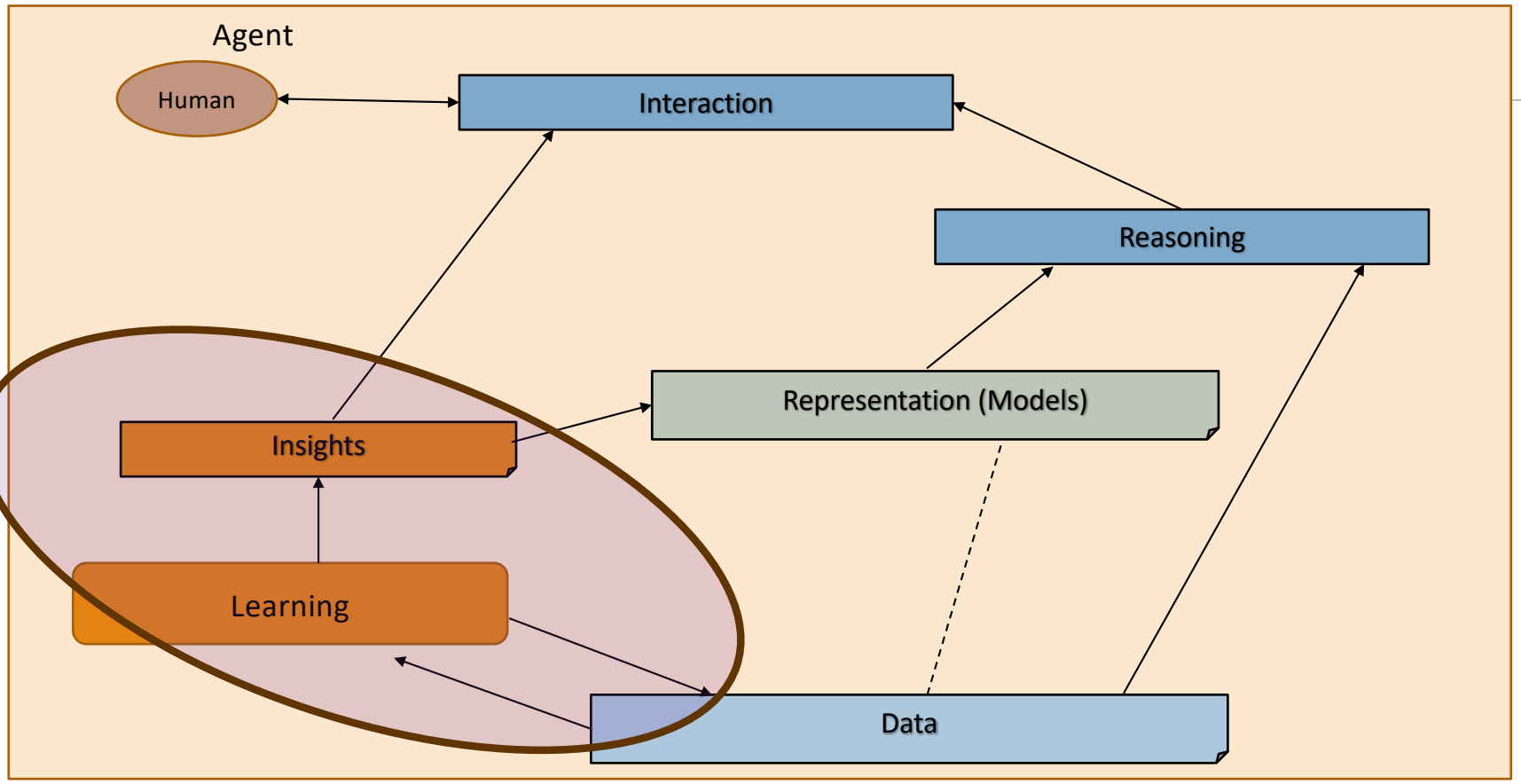
Recap of Lecture 16

- Topic discussed
  - Neural Networks
  - Deep Learning
  - Adversarial attacks
  - Trust Issues

Intelligent Agent Model



Relationship Between Main AI Topics



Where We Are in the Course

CSCE 580/ 581 – In This Course

- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 4-5: Search, Heuristics - Decision Making
- Week 6: Constraints, Optimization – Decision Making
- Week 7: Classical Machine Learning – Decision Making, Explanation
- Week 8: Machine Learning - Classification
- Week 9: Machine Learning - Classification – Trust Issues and Mitigation Methods
- Topic 10: Learning neural network, deep learning, Adversarial attacks
- Week 11: Large Language Models – Representation, Issues
- Topic 12: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 13: Planning, Reinforcement Learning – Sequential decision making
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

Main Section

Credit: Retrieved from internet

Text Processing

Examples

- Processing
  - Resumes -> e.g., selecting candidates
  - Answer papers (of examinations) -> e.g., grading
  - Medical reports -> e.g., helping patients
- Predicting
  - Business intelligence
  - Financial forecasts
- Generating
  - News summary
  - Summarizing literature for research

Common Textual Data Processing Steps for ML

- Input: strings / documents/ corpus
- Processing steps (task dependent / optional - *)
  - Parsing
  - Word pre-processing
    - Tokenization – getting tokens for processing
    - Normalization* - making into canonical form
    - Case folding* – handling cases
    - Lemmatization* – handling variants (shallow)
    - Stemming* – handling variants (deep)
  - Semantic parsing – representations for reasoning with meaning *
  - Embedding – creating vector representation*

CSCE 771 goes into details

Common NLP Tasks

- Extracting entities [Entity Extraction]
- Finding sentiment [Sentiment Analysis]
- Generating a summary [Text Summarization]
- Translating to a different language [Machine translation]
- Natural Language Interface to Databases [NLI]
- Natural Language Generation [NLG]

CSCE 771 goes into details

Word Representation: Paper Discussion

Contextual Word Representations: Putting Words into Computers",

by Noah Smith, CACM June 2020

https://cacm.acm.org/research/contextual-word-representations/

Problem

- How to represent words ?
- How to measure similarity, e.g., between words, and texts?
- How to determine different contexts (senses) in which words are used?
- How to handle noise, typos?

S1 - This is an apple
S2 - These are apples

S3 - This is an apples
S4 - There are apply

Option 1 - Characters

- How to represent words?
  - Characters / Unicode / …
- How to measure similarity between words, and texts?
  - Edit distance***: actions to convert one string to another***
  - Hamming distance: ***difference considering substitution***
- How to determine different contexts (senses) in which words are used?
  - Neighborhood of words: Bi-, tri-, N-gram representations

Distance between: Kitten, Sitting

Edit Distance

Algorithm	Operations Allowed			
	Insertions	Deletions	Substitutions	Transposition
Levenshtein Distance	✓	✓	✓	
Longest Common Subsequence (LCS)	✓	✓		
Hamming Distance			✓	
Damerau–Levenshtein Distance	✓	✓	✓	✓
Jaro distance				✓

Levenshtein distance:
1.kitten → **s**itten (substitute "s" for "k")
2.sitt**e**n → sitt**i**n (substitute "i" for "e")
3.sittin → sittin**g** (insert "g" at the end)

LCS distance (insertions and deletions only):
1.kitten → itten (delete "k" at 0)
2.itten → **s**itten (insert "s" at 0)
3.sitt**e**n → sittn (delete "e" at 4)
4.sittn → sitt**i**n (insert "i" at 4)
5.sittin → sittin**g** (insert "g" at 6)

Source: https://en.wikipedia.org/wiki/Edit_distance

Option 2 - Vectors

- How to represent words? Vectors
  - But, what scheme in vectors
    - One-hot encoding
    - Arbitrary, principled, …
- How to measure similarity between words, and texts?
  - Cosine similarity
- How to determine different contexts in which words are used?
  - Neighborhood of words: Bi-, tri-, N-gram representations
  - Contextual word vectors

Cosine Similarity

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\cdot\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Property: two proportional vectors have a cosine similarity of 1, two orthogonal vectors have a similarity of 0, and two opposite vectors have a similarity of -1.

Usually used for [0,1]

Sci-kit method python: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

Source: https://en.wikipedia.org/wiki/Cosine_similarity

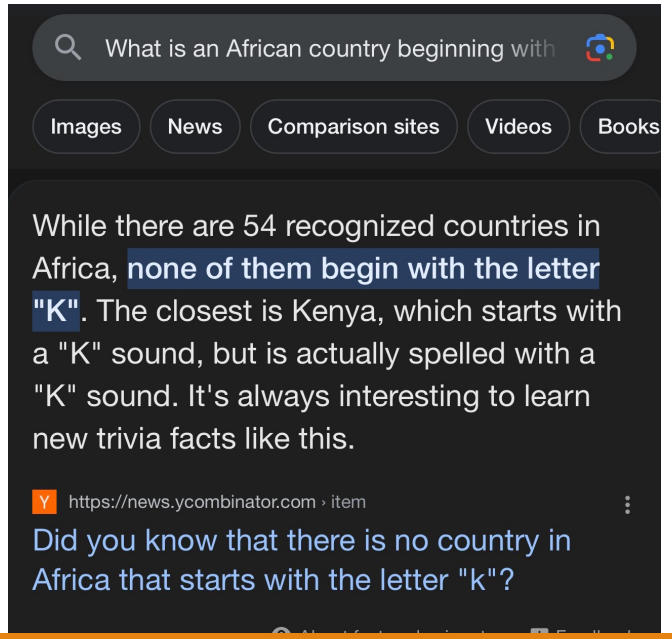
Language Models (LMs)

LLM Fiascos

- See many on GitHub: https://github.com/biplav-s/course-nl-f24/blob/main/reading-list/Readme-LLMs.md

- “what is an african country that starts with k”

  Google and its LLM [8/9 Nov 2023]



Language Model

Problem:
Given a sentence fragment, predict what word(s) come next

Applications:
- Spelling correction
- speech recognition
- machine translation,
- …

Language Model:
estimate probability of substrings of a sentence

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{P(w_1, w_2, \ldots, w_{i-1}, w_i)}{P(w_1, w_2, \ldots, w_{i-1})}$$

Bigram approximation

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) \approx \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$$

From Jurafsky & Martin

Language Model

Markovify library
https://github.com/jsvine/markovify

Language Model:
estimate probability of substrings of a sentence

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{P(w_1, w_2, \ldots, w_{i-1}, w_i)}{P(w_1, w_2, \ldots, w_{i-1})}$$

See code samples with Markovify library on Github

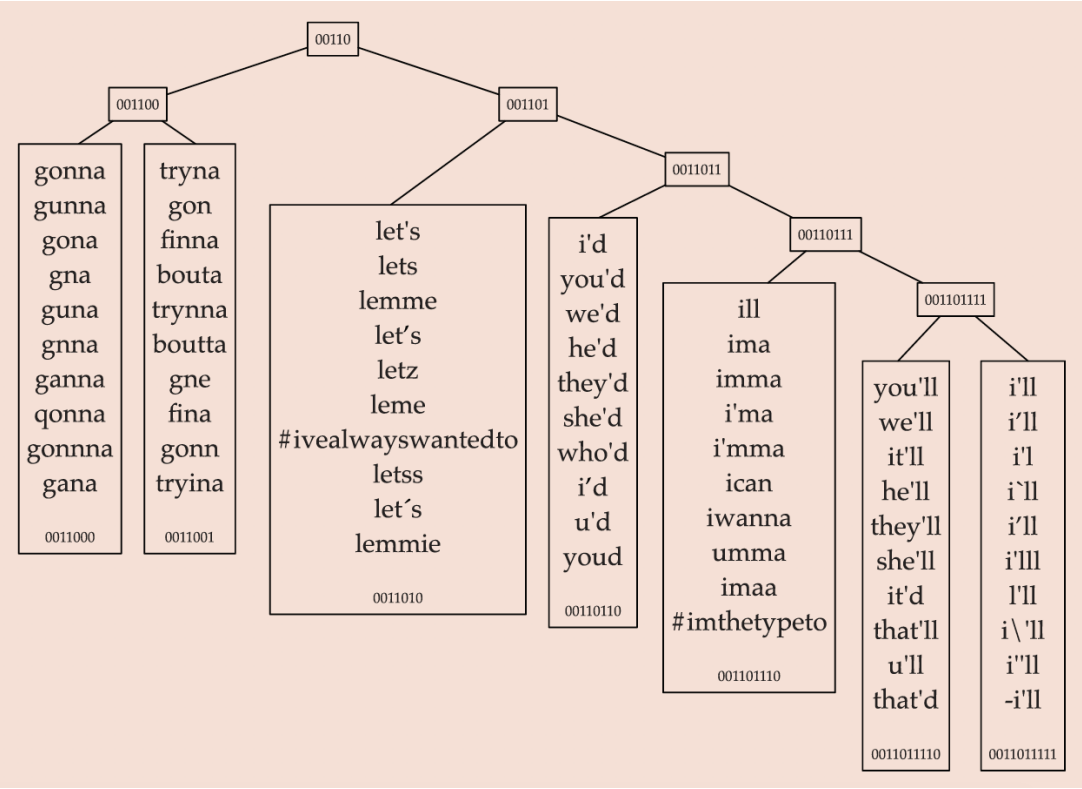
- *Prepare data – two datasets shown*
- *Try generator:*
  - *https://github.com/biplav-s/course-nl/blob/master/l7-language/code/TryMarkovifyLangModel.ipynb*

Contextual Word Embeddings

- Words as discrete
- Words with distributional assumptions:
  - Context: given a word, its nearby words or sequences of words
  - Words used in similar ways are likely to have related meanings; i.e., words used in the same (similar) context have related meanings
    - No claim about meaning except relative similarity v/s dis-similarity of words

Contextual Representation by Clustering

- Cluster words by context
- Compare with words in a manually-created taxonomy, e.g., Wordnet



The 10 most frequent words in clusters in the section of the hierarchy with prefix bit string 00110.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N.A. Improved part-ofspeech tagging for online conversational text with word clusters. In Proceedings of 2013 NAACL.

Credit:
Contextual Word Representations: Putting Words into Computers", by Noah Smith, CACM June 2020

Contextual Representation by Dimensionality Reduction

- Creating word vectors in which each dimension corresponds to the frequency the word type occurred in some context.
- Strategy 1: select contexts
  - Examples
    - Custom methods
    - TF-IDF
  - Approach
    - Use words
      - Words in the neighborhood
      - Words of specific types
    - Build vectors
    - Use vector operations to derive meaning

context words	v(astronomers)	v(bodies)	v(objects)
't			1
,		2	1
.	1		1
1			1
And			1
Belt			1
But	1		
Given			1
Kuiper			1
So	1		
and		1	
are		2	1
between			1
beyond		1	
can			1
contains		1	
from	1		
hypothetical			1
ice		1	
including		1	
is	1		
larger		1	
now	1		
of	1		
only			1
out		1	
potential		1	
the	1		1
these		2	1
they	1		
think	2		
those			1
thought		2	
what	1		

$$\text{cosine\_similarity}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}$$

	astronomers	bodies	objects
astronomers	$\frac{14}{\sqrt{14} \cdot \sqrt{14}} = 1$	$\frac{0}{\sqrt{24} \cdot \sqrt{14}} = 0$	$\frac{1+1}{\sqrt{14} \cdot \sqrt{16}} \approx 0.134$
bodies		$\frac{24}{\sqrt{24} \cdot \sqrt{24}} = 1$	$\frac{2+2+2}{\sqrt{24} \cdot \sqrt{16}} \approx 0.306$
objects			$\frac{16}{\sqrt{16} \cdot \sqrt{16}} = 1$

Bodies and **objects** are **most similar** (0.306) than

- **Bodies** and **astronomers** (0)
- **Objects** and **astronomers** (0.134)

Credit:
Contextual Word Representations: Putting Words into Computers", by Noah Smith, CACM June 2020

TF-IDF based Word Representation -1

- Given N documents

- **Term frequency (TF):** for term (word) t in document d = tf(t, d)

Variants to reduce bias due to document length

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Sources:
(a) sci-kit documentation
(b) Wikipedia: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

TF-IDF based Word Representation -2

- Given N documents
- Term frequency (TF): for
  term (word) t in document d
  = tf(t, d)
- **Inverse document frequency IDF**(t)

  = log [N / **DF**(t)] + 1

DF(t) = **document frequency**, the number of documents in the document set that contain the term t.

- **TF-IDF**(t, d) = TF(t, d) * IDF(t),

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right) + 1$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N-n_t}{n_t}$

Sources:
(a) sci-kit documentation
(b) Wikipedia: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

TF-IDF Example Calculation

See sample code on GitHub:
https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l5-wordrepresent/Word%20Representations%20-%20Vectors.ipynb

Contextual Representation by Dimensionality Reduction - 1

- Strategy 2: learn contexts from documents. Vector size is given as input
- Train a neural network to learn vector representation
  - value placed in each dimension of each word type's vector is a parameter that will be optimized
  - Selection of parameter values is done using iterative algorithms / gradient descent
  - **Hope** is that ***different senses*** in which a word is used will be captured through the learning procedure as long as the dataset is large enough to represent all senses. Paper quotes: 30 meanings of **get**
- **Optionally**: Sometime task specific inputs are given during pre-processing, processing or post-processing

Disadvantage: individual dimensions are no longer interpretable

Contextual Representation by Dimensionality Reduction -2

- Strategy 2: learn contexts from documents. Vector size is given as input

Sometime task specific inputs are given during pre-processing, processing or post-processing

- Pre-processing
  - Vector initialization by pre-training. Called **finetuning**
- Processing
  - **Knowledge-infusion** (emerging area)
- Post-processing
  - Adjust output vectors so that word types that are related in reference taxonomy (like WordNet) are closer to each other in vector space. Called **retrofitting**.

Credit:
Contextual Word Representations: Putting Words into Computers", by Noah Smith, CACM June 2020

Where Are We

- Learning representation
  - Approach 1: count-based
    - Creating word vectors in which each dimension corresponds to the frequency the word type occurred in some context.
    - Example: TF-IDF
  - Approach 2: learning-based
    - learn contexts from documents. Vector size is given as input
    - Examples: Word2Vec, Glove, RNN/LSTM (arc), Transformers

Reading

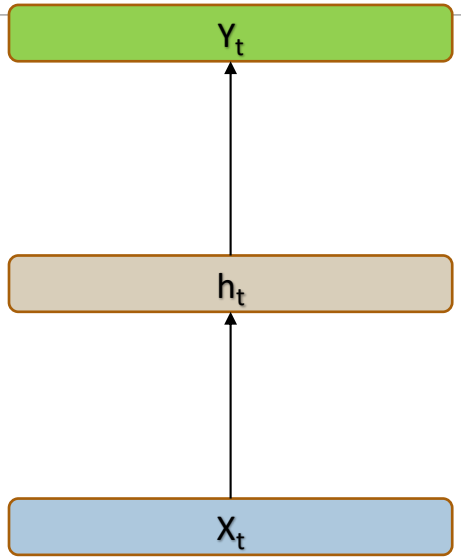
- Contextual Word Representations: Putting Words into Computers", by Noah Smith, CACM June 2020
- Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep Learning--based Text Classification: A Comprehensive Review. ACM Comput. Surv. 54, 3, Article 62 (April 2022), 40 pages. https://doi.org/10.1145/3439726
- Hang Li, Language Models: Past, Present, and Future, Communications of the ACM, July 2022, Vol. 65 No. 7, Pages 56-63 10.1145/3490443

Learning for LMs with NN

Recall: (Feed forward) NN

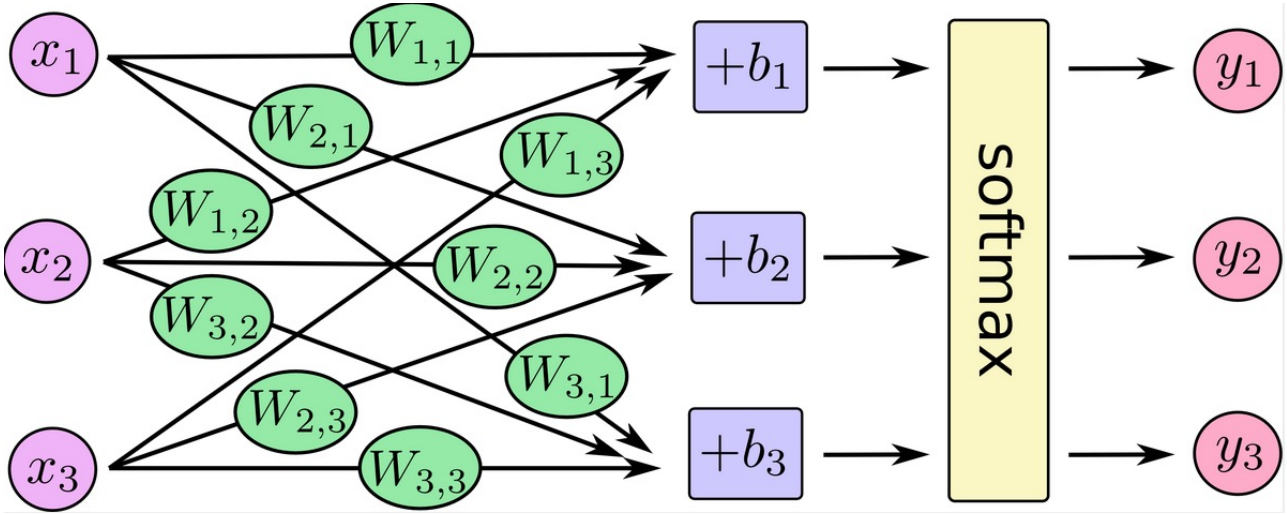
Propagation

$$f(X_j) = X_j W + b$$



Intuitive Description: https://jalammar.github.io/visual-interactive-guide-basics-neural-networks/, https://jalammar.github.io/feedforward-neural-networks-visual-interactive

Using (Feed forward) NN



Softmax

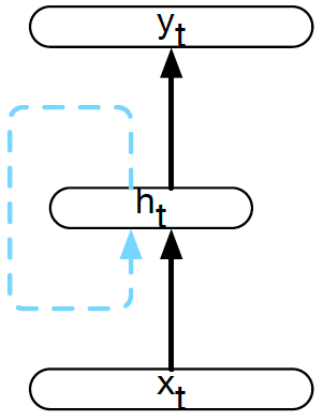
$$f(x)= \frac{1}{1+e^{-x}}$$

Source; see also : https://jalammar.github.io/visual-interactive-guide-basics-neural-networks/, https://jalammar.github.io/feedforward-neural-networks-visual-interactive

RNN - Recurrent Neural Networks

- **Recurrence**: A *recurrence* relation is an equation that defines a sequence based on a rule that gives the next term as a *function* of the previous term(s).
[https://mathinsight.org/definition/recurrence_relation]

- Simple Recurrent NN or Elman Network

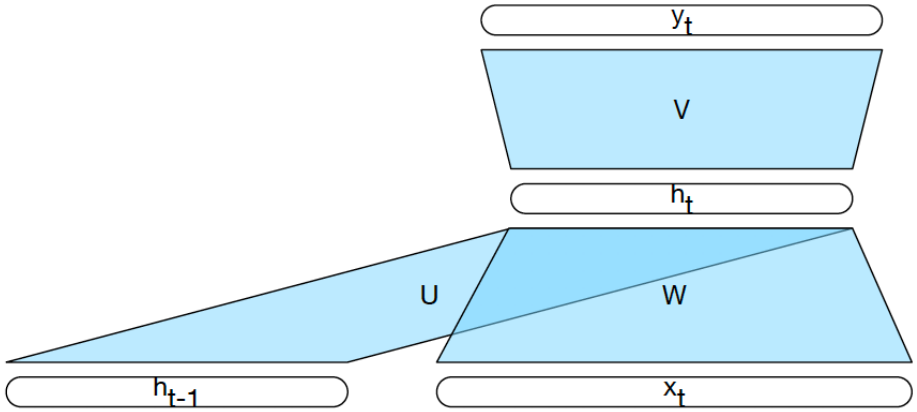


Source: Jurafsky and Martin

RNN

Recurrence unrolled

U, W, V are Weights to be learned



$$h_t = g(Uh_{t-1} + Wx_t)$$
$$y_t = f(Vh_t)$$

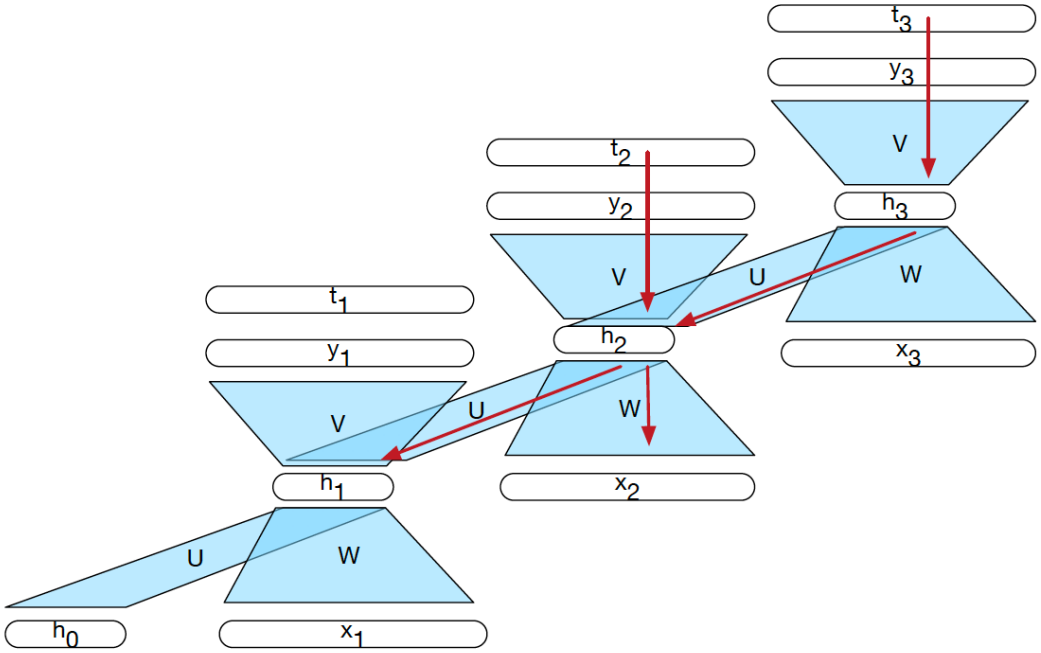
$$y_t = softmax(Vh_t)$$

Source: Jurafsky and Martin

RNN Backpropagation of Errors

Recurrence unrolled

U, W, V are Weights to be learned



Source: Jurafsky and Martin

RNN-based Language Model

- Based on characters or words
- At each step (i.e., character or word)
  - the network retrieves a word embedding for the current word as input
  - combines it with the hidden layer from the previous step to
    - compute a new hidden layer
    - generate an output layer which is passed through a softmax layer to generate a probability distribution over the entire vocabulary.

$$P(w_n|w_1^{n-1}) = y_n$$
$$= softmax(Vh_n)$$

Prob. of a word

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$
$$= \prod_{k=1}^{n} y_k$$

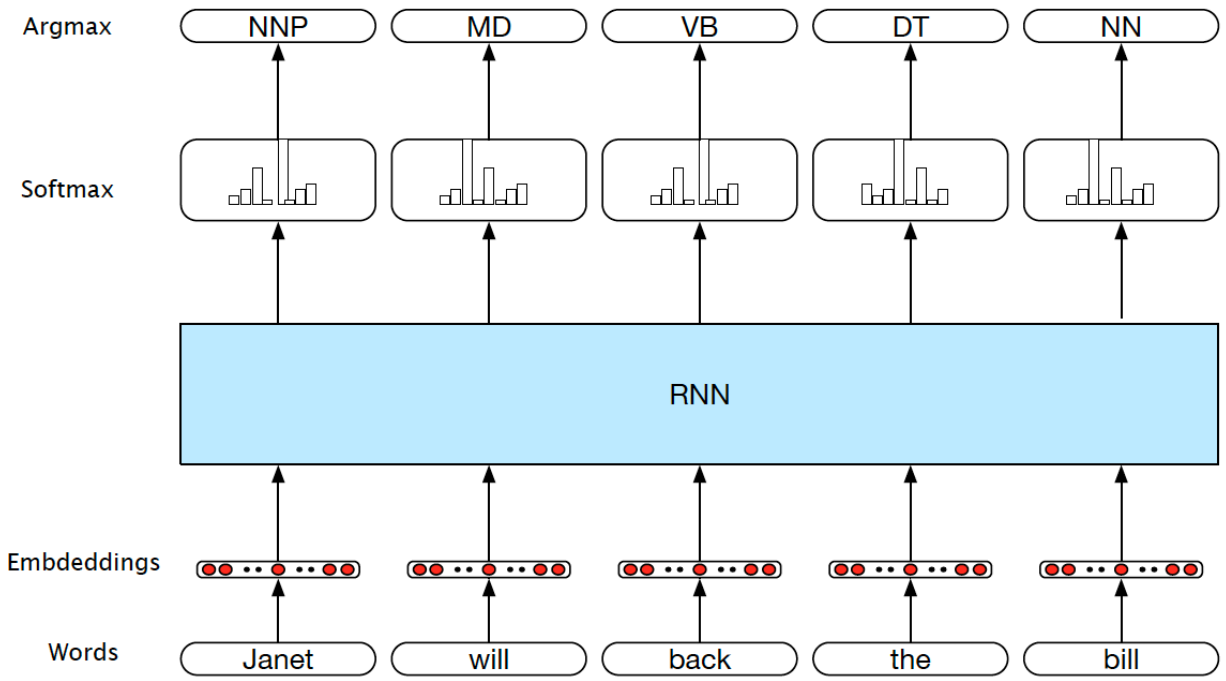
Prob. of a sequence

Source: Jurafsky and Martin

RNN Discussion

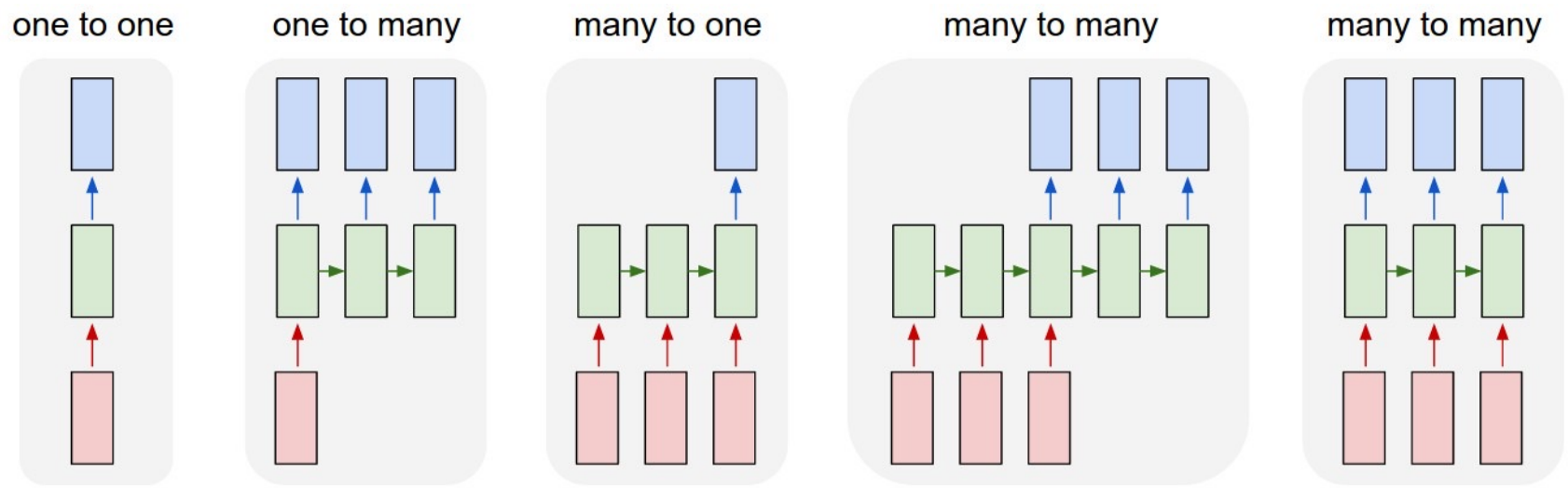
- Language model
  - Not dependent on N-gram boundaries
  - Whole sequence is the context
- Program generation
  - Complexity is Turing-complete
  - In practical terms: On the Practical Computational Power of Finite Precision RNNs for Language Recognition, Gail Weiss, Yoav Goldberg, Eran Yahav, ACL 2018, https://www.aclweb.org/anthology/P18-2117/

RNN Usage Example: Sentence Labeling



Source: Jurafsky and Martin

RNN - Many Applications



Language model

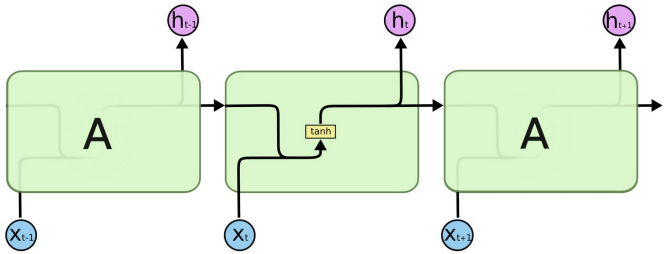
Caption generation

Sentiment detection

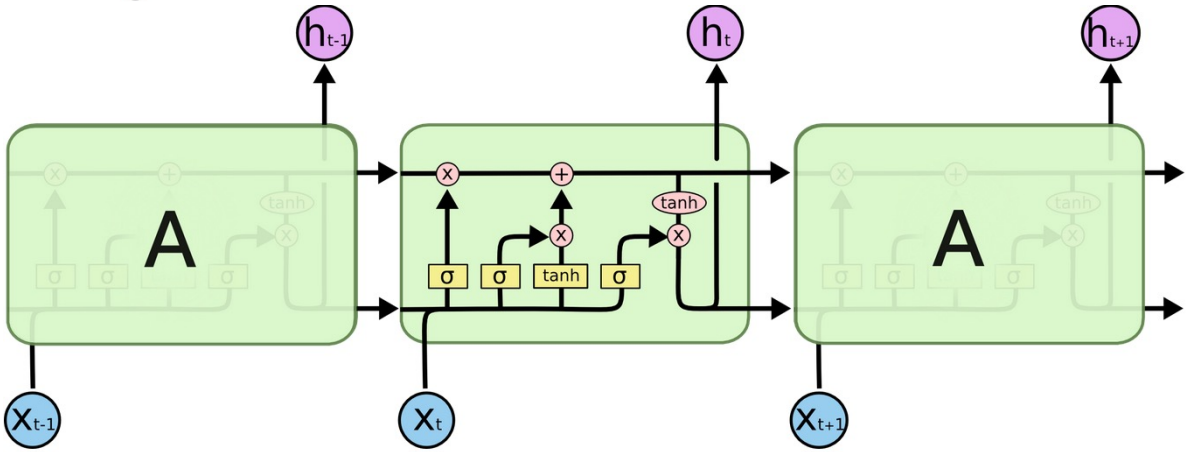
Machine translations

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN and LSTM - Long Short Term Memory



RNN

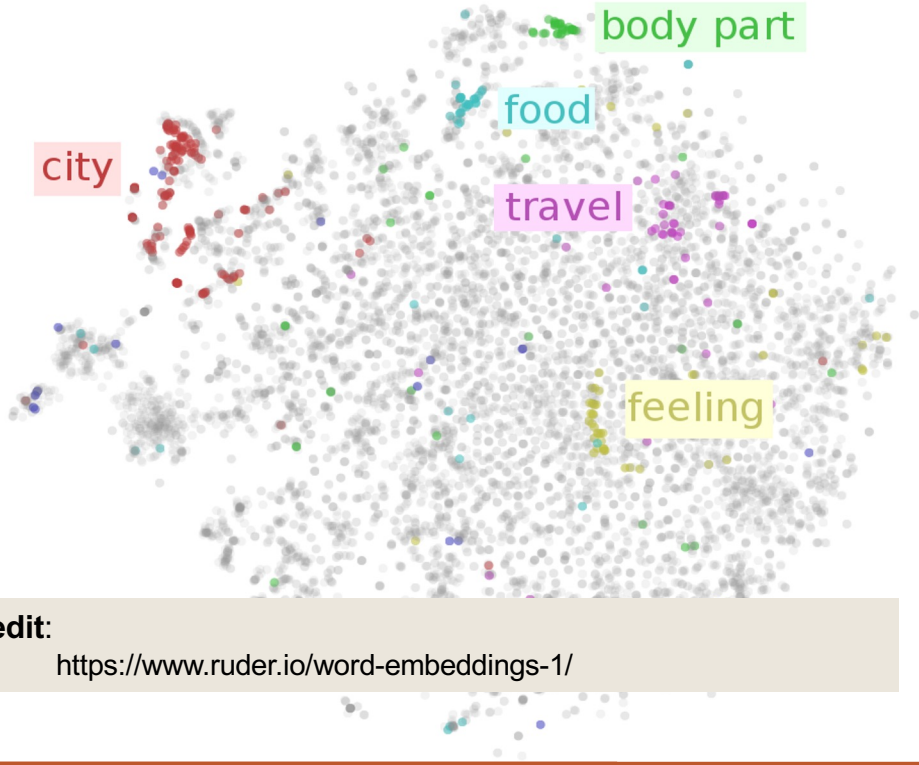


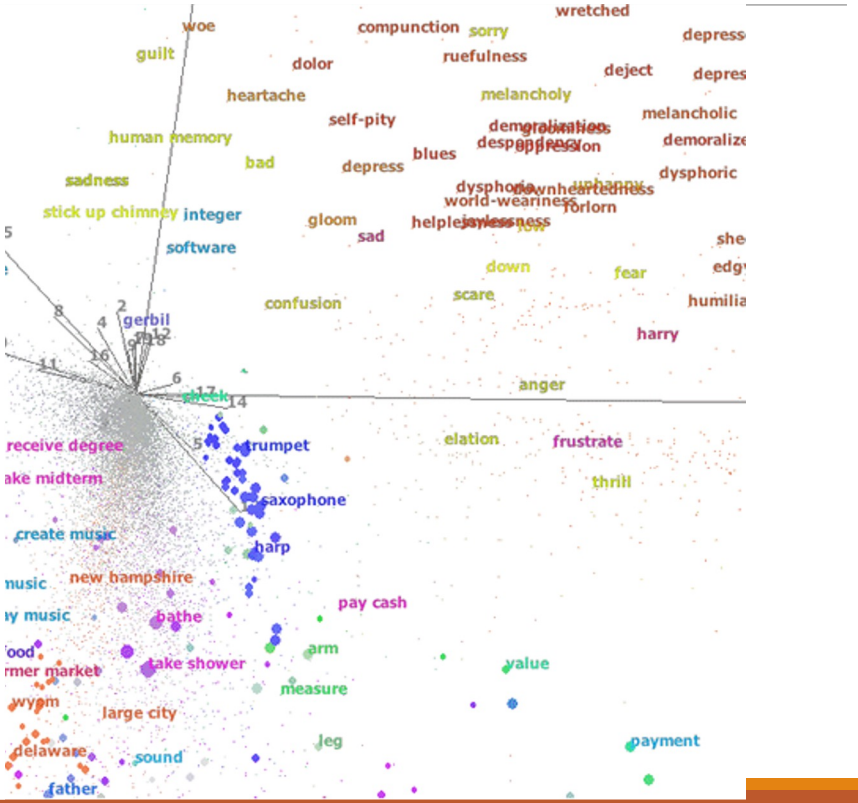
LSTM

To learn long term relationships; has 4 NNs

Source and details: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Evaluating Language Models





Credit:

- https://www.ruder.io/word-embeddings-1/

Evaluating LMs

Evaluation – Language Model

- **Intrinsic evaluation**: measure the quality of a model independent of any application
- **Extrinsic evaluation**: situate model in an application and evaluate the whole application for improvement. Also called in-vivo evaluation

Perplexity

$$\text{Average NLL} = -\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, w_2, ..., w_{i-1})$$

where N is the total number of words in the test dataset, and $P(w_i|w_1, w_2, ..., w_{i-1})$ is the probability of word w_i given the previous words $w_1, w_2, ..., w_{i-1}$.

$$\text{Perplexity} = e^{\text{Average NLL}}$$

Value range: best: 1, **worst**: positive infinite;
practical upper bound: number of words in vocabulary

1. P(John) = 0.1
2. P(bought | John) = 0.4
3. P(apples | bought) = 0.3
4. P(from | apples) = 0.5
5. P(the | from) = 0.6
6. P(market | the) = 0.7

Now, let's compute the probability of the generated sequence:

$P(\text{"John bought apples from the market"}) = P(John) \times P(bought|John) \times P(apples|bought) \times P(from|apples) \times P(the|from) \times P(market|the)$

$P(\text{"John bought apples from the market"}) = 0.1 \times 0.4 \times 0.3 \times 0.5 \times 0.6 \times 0.7$

Hence, P("John bought apples from the market") = 0.00252

Average NLL = – log(0.00252) / 6. [N = 6 as the model generated six words]

Hence, Average NLL = 0.99725

Perplexity = Exp(Average NLL) = 2.71

Credit:

- https://blog.uptrain.ai/decoding-perplexity-and-its-significance-in-llms/
- https://medium.com/@priyankads/perplexity-of-language-models-41160427ed72

Perplexity Comments

1. A model with a vocabulary of 10,000 words and a perplexity of 2.71 is much better than a model with a vocabulary of 100 words and the same perplexity score of 2.71.
2. Lower perplexity results in higher consistency. As we know, LLMs are non-deterministic, i.e., the same inputs can result in two different outputs; a lower perplexity means that the model is more likely to produce the same output over multiple runs.
3. Perplexity is the inverse of the geometric mean of the probability of each word. Hence, the inverse of average probability (2.3077 in the previous case) can be considered a good proxy for quick calculations.
4. This calculation happens in the tokens space (compared to the words space), but the core principle remains the same.

Credit:

- https://blog.uptrain.ai/decoding-perplexity-and-its-significance-in-llms/

Perplexity

- Suppose
  - P('X') = 0.25
  - P('Y') = 0.5
  - P('Z') = 0.25
- Perplexity
  - ('XXX') = - exp (log (0.25 x 0.25 x 0.25) * (1/3)) = 3.94
  - Perplexity ('XYX') = - exp(log(0.25 x 0.5 x 0.25) * (1/3)) =
- Lower the number, the better is the model

Evaluation – Extrinsic

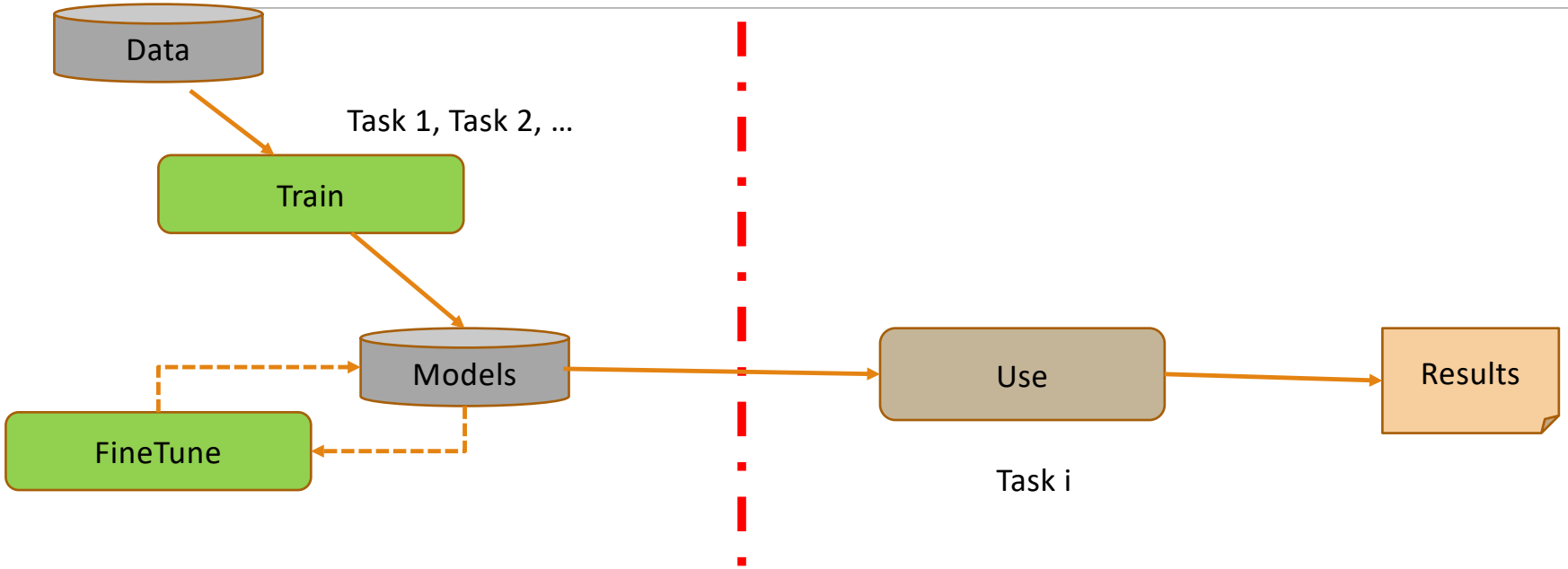
- **Extrinsic evaluation**: situate model in an application and evaluate the whole application for improvement. Also called in-vivo evaluation

- **Examples**:
  - BLEU and Rogue scores of generated text

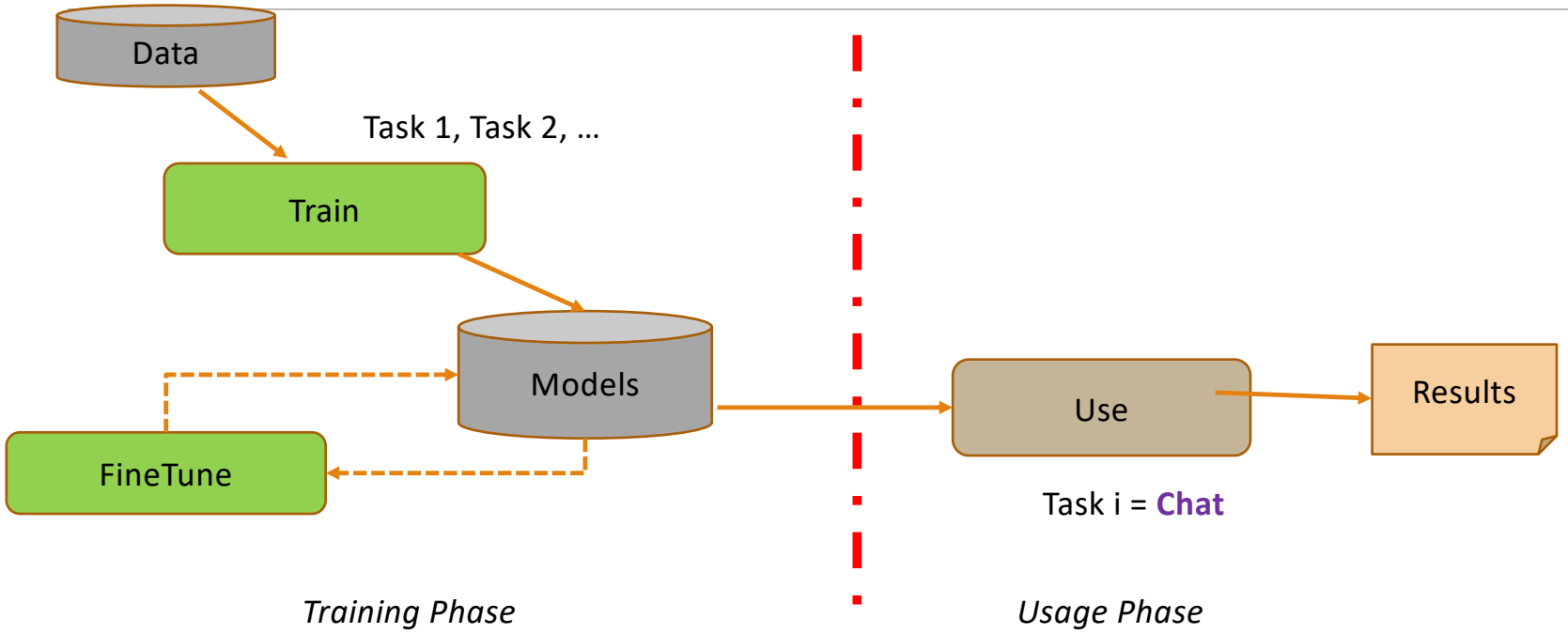
Credit:
- https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb

Large LMs (LLMs)

Large Language Models (LLMs) Basics



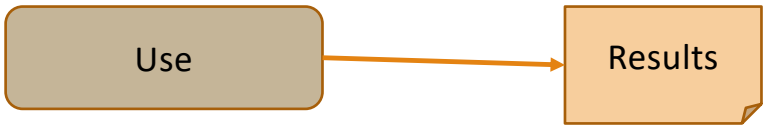
ChatGPT: Large Language Models (LLMs) based Chatbot



Another "Turning Point" Moment In Technology

Raised interest about Chatbots among public

- Excitement about new use-cases
- Concerns about social impact – cheating, jobs, misinformation
- Renewed calls for regulations



Task i = **Generally speaking: content generation – text, image, video, audio, …**

Usage Phase

BERT - **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Learns with two tasks

- Predicting missing words in sentences
  - mask out 15% of the words in the input, predict the masked words.
- Given two sentences A and B, is B the actual next sentence that comes after A, or just a random sentence from the corpus?

(12-layer to 24-layer Transformer) on (Wikipedia + BookCorpus)

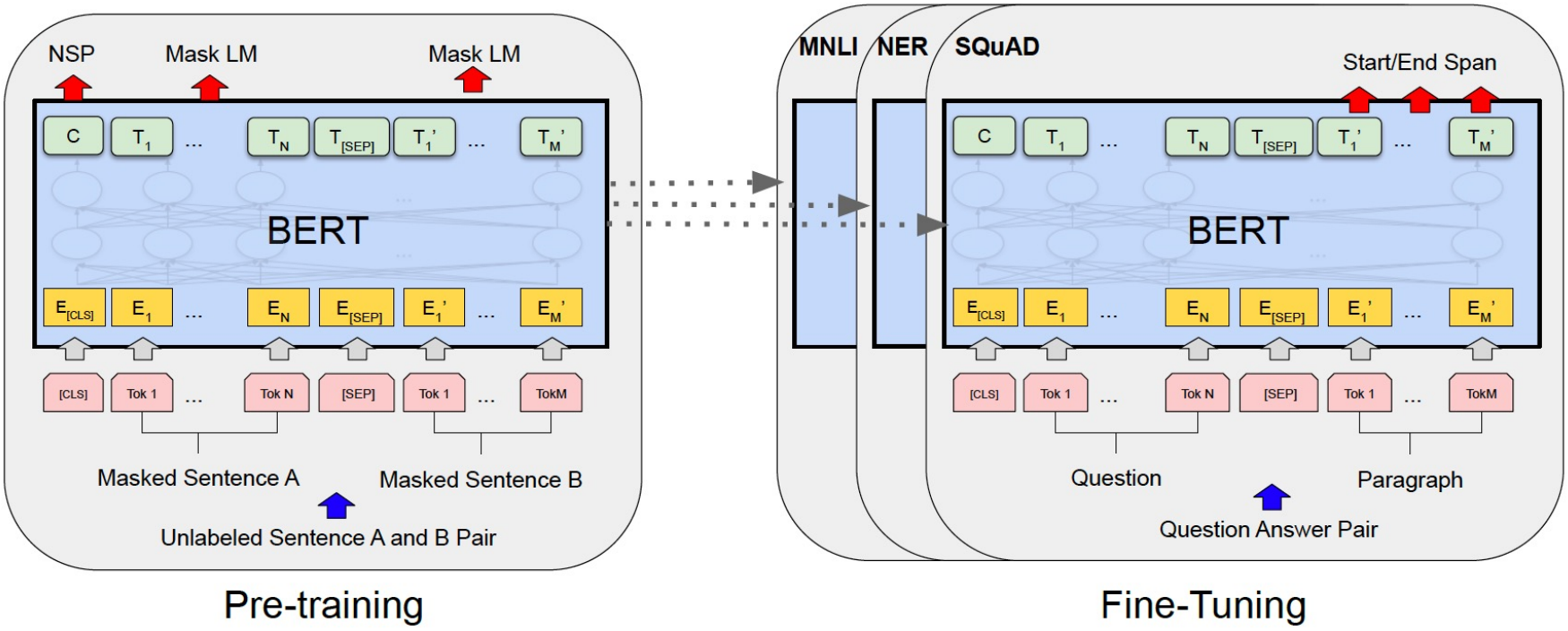
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon

Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence

Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence

Credit and details: https://github.com/google-research/bert

BERT: Before and During Usage



Credit and details: **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018

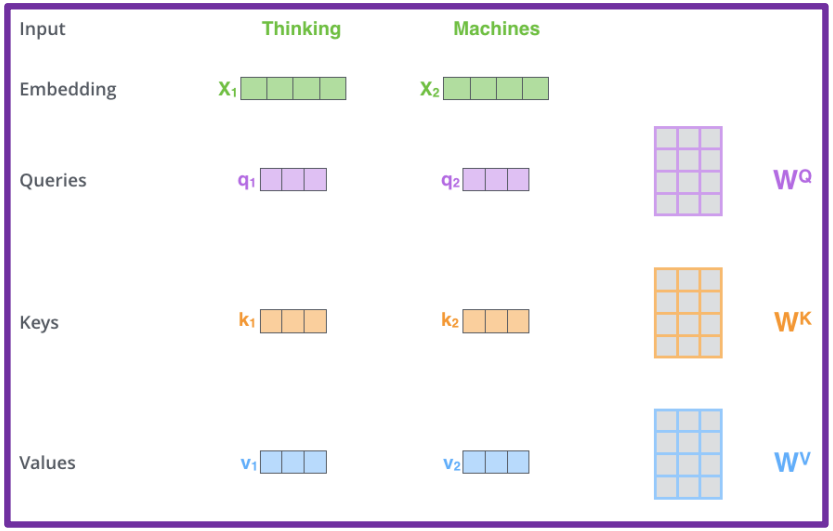
Transformer

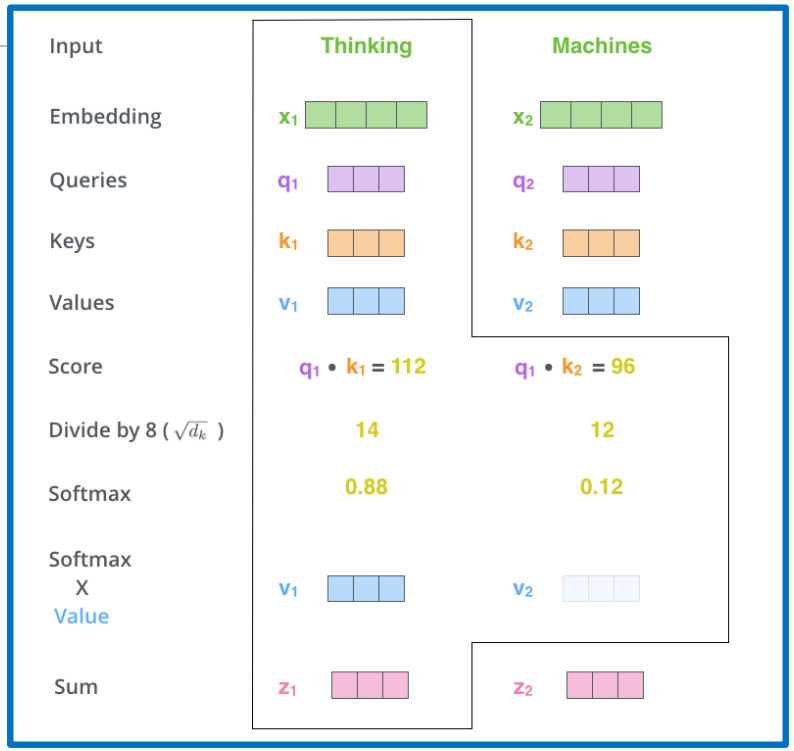
- RNN/ LSTM with
  - Attention
    - attention layer can access all previous states and weighs them according to some learned measure of relevancy to the current token, providing sharper information about far-away relevant tokens
    - **Query** vector, **Key** vector, and **Value** vectors introduced during encoding and decoding phase
  - Parallelization of learning
  - See Dr. Amitava Das's slide for Attention/ BERT video
    - https://prezi.com/view/amx5hBo8UhMOn1rPyJ02/

Source and details: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model), http://jalammar.github.io/illustrated-transformer/

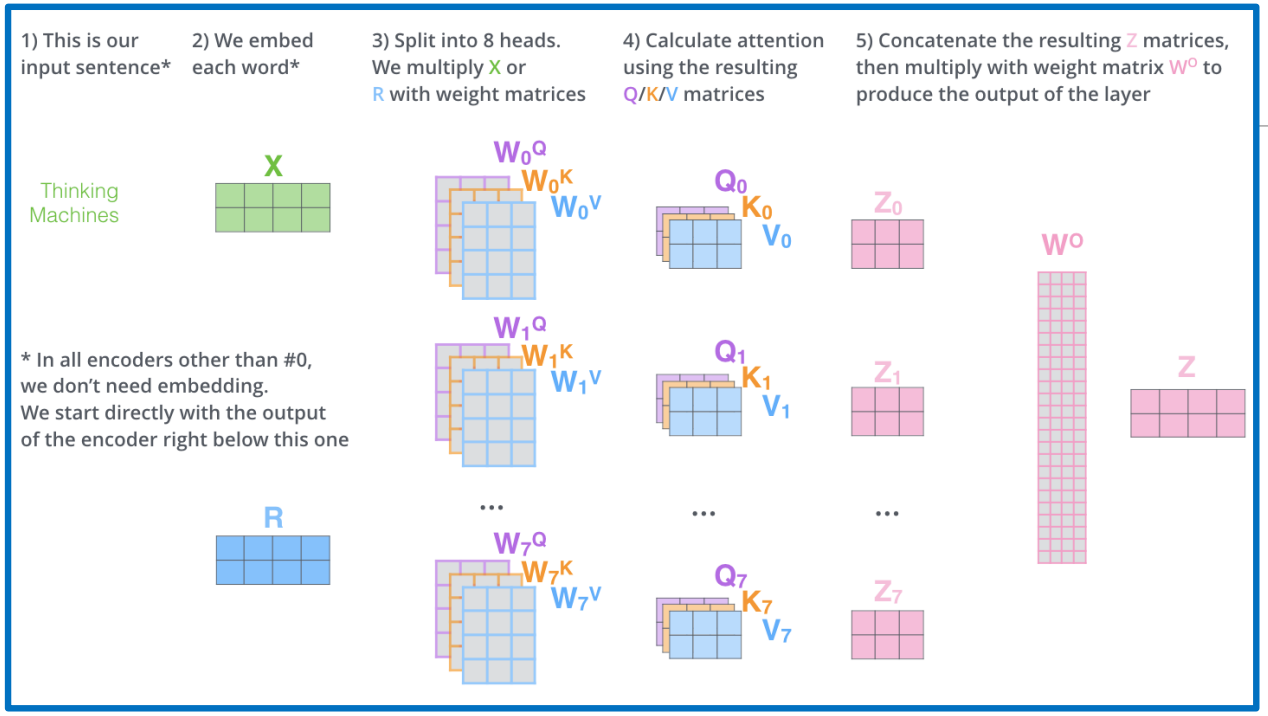
Transformer

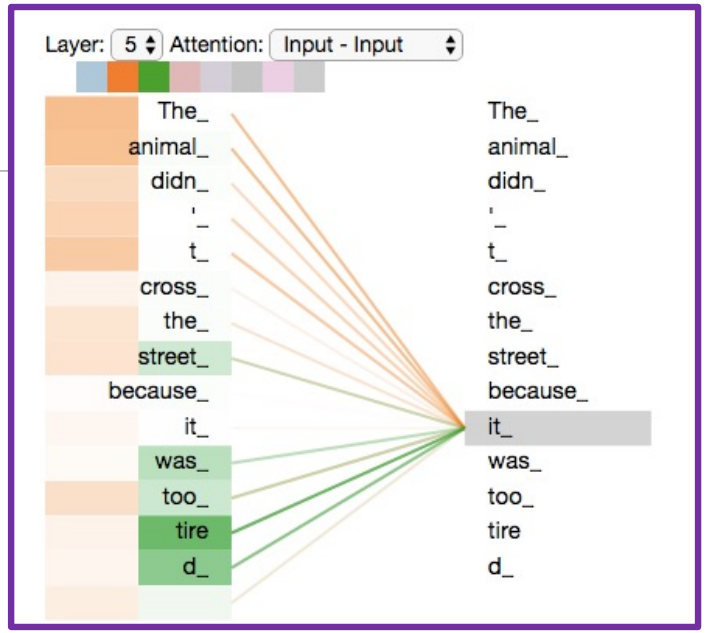
Credit: http://jalammar.github.io/illustrated-transformer/





Transformer





Credit: http://jalammar.github.io/illustrated-transformer/

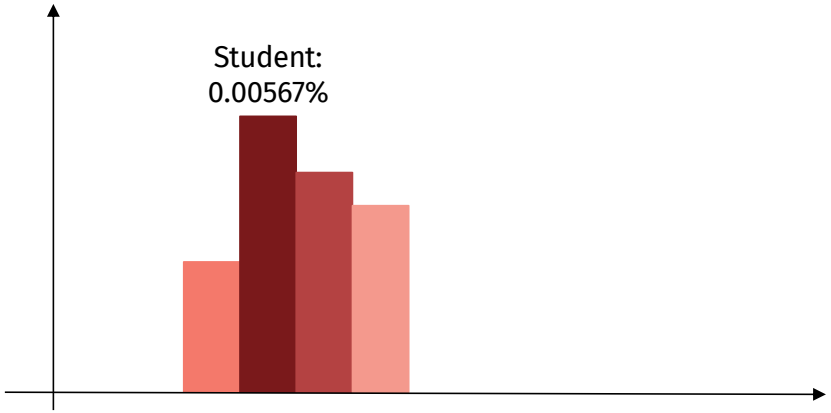
(Large) Language Modeling

Input Kaushik Roy is a PhD Student

Sequence	Kaushik	Roy	is	a	PhD	Student
	1	2	3	4	5	6

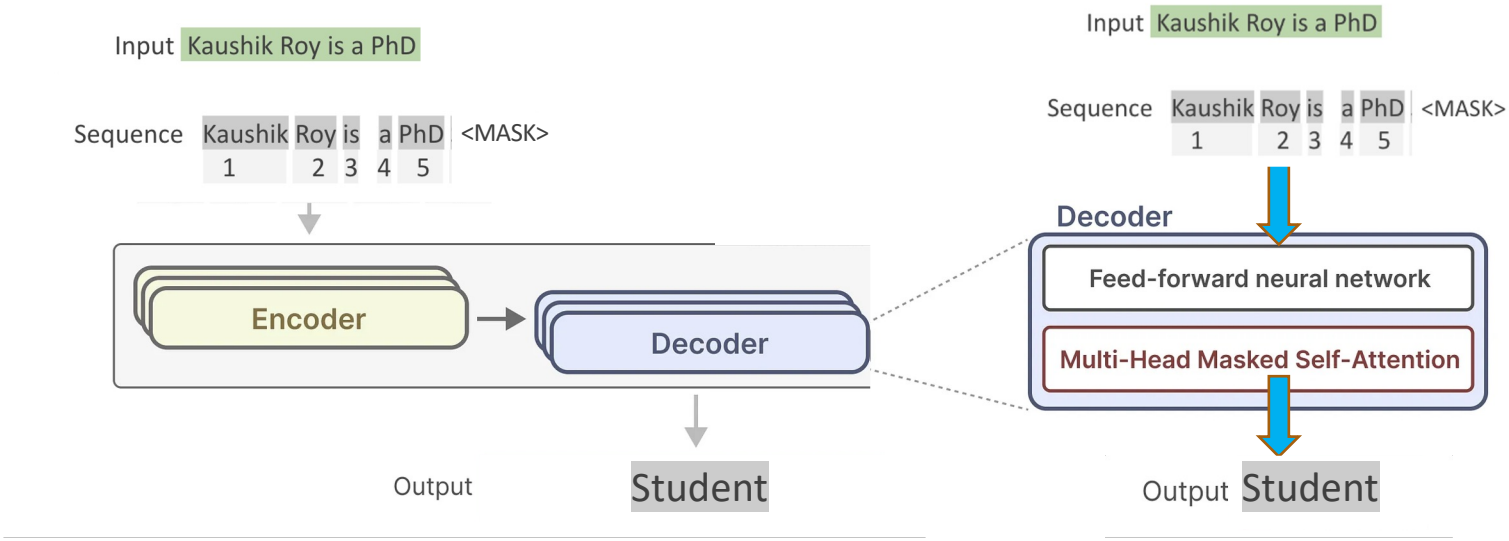
Student

Did you mean: Student
Did you mean: Student Square
Did you mean: Student Success



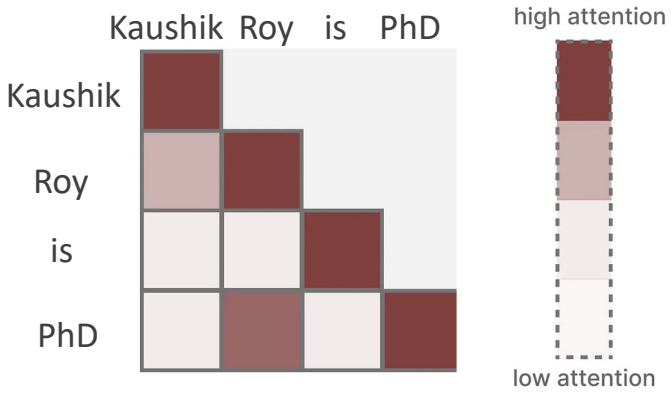
Credit: Kaushik Roy, CSCE 771 Guest Lectures

BERT and family



Credit: Kaushik Roy, CSCE 771 Guest Lectures

BERT and family - Multi-headed Self-Attention



Each attention head can be implemented in parallel

worker 1

$\mathbf{Q}_1$ $\mathbf{K}_1$ $\mathbf{V}_1$ → $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

worker 2

$\mathbf{Q}_2$ $\mathbf{K}_2$ $\mathbf{V}_2$ → $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

worker 3

$\mathbf{Q}_3$ $\mathbf{K}_3$ $\mathbf{V}_3$ → $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

Credit: Kaushik Roy, CSCE 771 Guest Lectures

Self Attention Snippet and Live Coding - BERT from Scratch

```python
class SelfAttention(nn.Module):
    def __init__(self, embed_size, heads):
        super(SelfAttention, self).__init__()
        self.embed_size = embed_size
        self.heads = heads
        self.head_dim = embed_size // heads

        assert (
            self.head_dim * heads == embed_size
        ), "Embedding size needs to be divisible by heads"

        self.values = nn.Linear(self.head_dim, embed_size, bias=False)
        self.keys = nn.Linear(self.head_dim, embed_size, bias=False)
        self.queries = nn.Linear(self.head_dim, embed_size, bias=False)
        self.fc_out = nn.Linear(embed_size, embed_size)
```

Github: https://github.com/kauroy1994/CSCE-771-NLP-Class-11/tree/main (BERT-based CV Processing)

Credit: Kaushik Roy, CSCE 771 Guest Lectures

Live Coding - BERT using Libraries

- Use the transformers library to extract information from the CV

```python
import pdfplumber
from transformers import AutoTokenizer, AutoModelForTokenClassification
from transformers import pipeline

# Step 1: Load the CV PDF and extract text
def extract_text_from_pdf(pdf_path):
    with pdfplumber.open(pdf_path) as pdf:
        pages = [page.extract_text() for page in pdf.pages]
    return ''.join(pages)

# Extract text from the CV
pdf_path = "CV.pdf"
cv_text = extract_text_from_pdf(pdf_path)

# Step 2: Load pre-trained BERT model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("dslim/bert-base-NER")
model = AutoModelForTokenClassification.from_pretrained("dslim/bert-base-NER")

# Step 3: Use pipeline for Named Entity Recognition
nlp = pipeline("ner", model=model, tokenizer=tokenizer, grouped_entities=True)

# Step 4: Extract entities from the CV text
ner_results = nlp(cv_text)

# Display the recognized entities
for entity in ner_results:
    print(f"Entity: {entity['word']}, Label: {entity['entity_group']}")

# Step 5: Post-process the entities for CV data extraction (optional)
# For example, grouping entities like degree, institution, and dates
def extract_education_details(ner_results):
    education = []
    current_education = {}
    for entity in ner_results:
        if entity['entity_group'] == 'ORG':
            current_education['institution'] = entity['word']
        elif entity['entity_group'] == 'MISC':  # Assuming degrees are labeled as MISC
            current_education['degree'] = entity['word']
        elif entity['entity_group'] == 'DATE':
            current_education['year'] = entity['word']

        # Save the current education entry
        if 'institution' in current_education and 'degree' in current_education and 'year' in current_education:
            education.append(current_education)
            current_education = {}
    return education

education_details = extract_education_details(ner_results)

# Display the structured education data
print("Extracted Education Details:", education_details)
```

Github: https://github.com/kauroy1994/CSCE-771-NLP-Class-11/tree/main (BERT-based CV Processing)

Credit: Kaushik Roy, CSCE 771 Guest Lectures

Major LM Types

- Large
  - Large training dataset
  - Large number of parameters
- General purpose
  - Commonality of human languages
  - Resource restriction
- Pre-trained and fine-tuned

Credits: Google Cloud Skills Boost

LLMs have three different architectures - (a) encoder-only, (b) decoder-only, and (c) encoder-decoder, each with their own benefits.

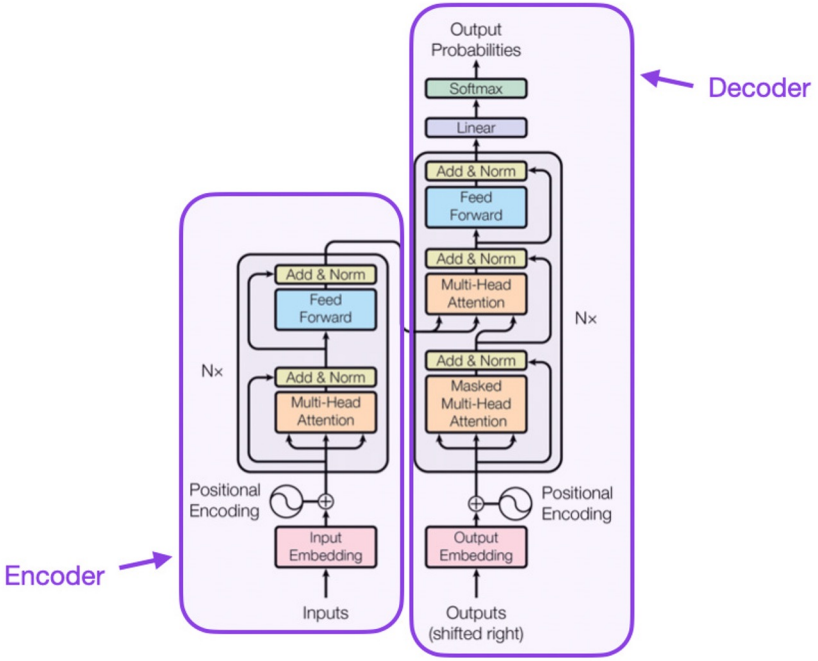
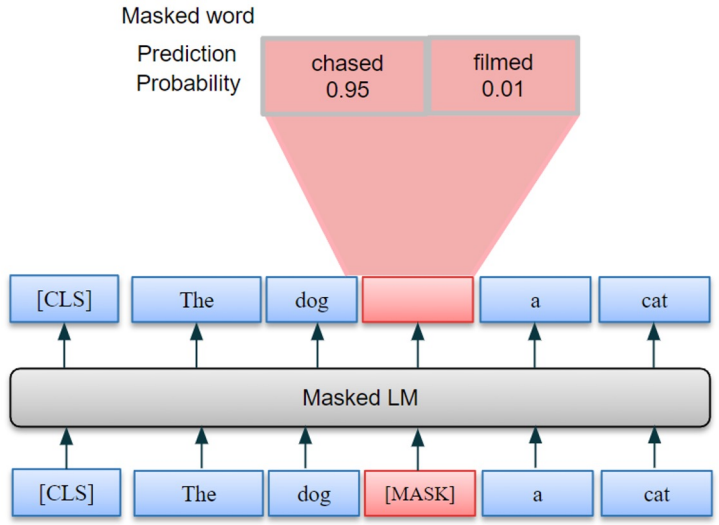


Figure. The Transformer - model architecture.

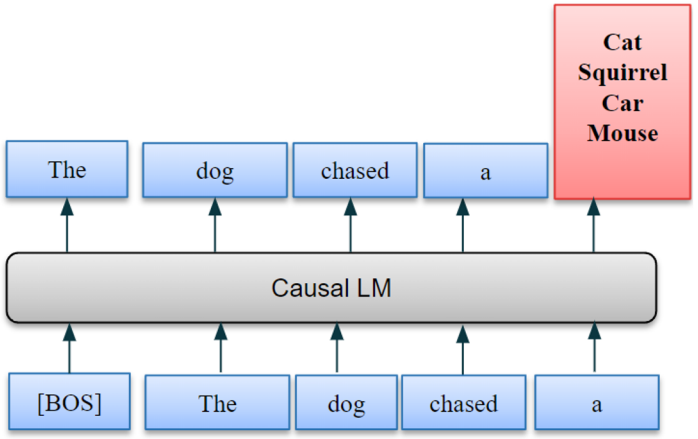
Encoder-only

- Encoder-only architectures are trained to understand the bidirectional context by predicting words randomly masked in a sentence.
- **Example:** BERT
- **Effective for:** sentiment analysis, classification, entailment.



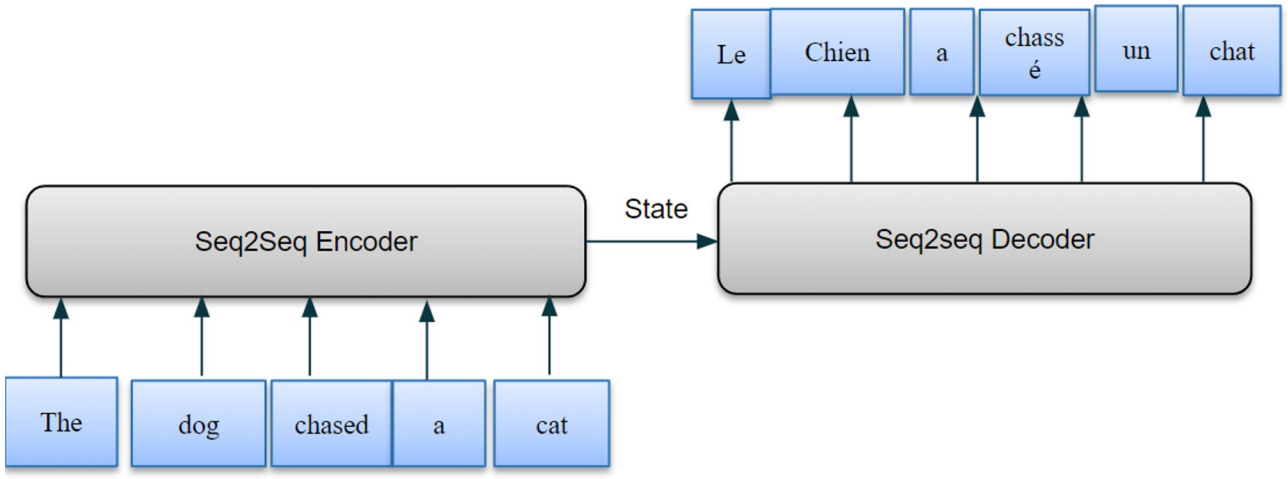
Decoder-only

- Decoder-only architectures are designed for tasks where text generation is sequential and dependent on the preceding context.
- They predict each subsequent word based on the preceding words, modeling the probability of a word sequence in a forward direction.
- **Example:** GPT-4, Llama series, Claude, Vicuna.
- **Effective for:** Content generation.



Encoder-Decoder

- Encoder-Decoder architectures are designed to transform an input sequence into a related output sequence.
- **Example:** T5, CodeT5, FlanT5
- **Effective for:** summarization, language translation.



Large Language Models (LLMs)

- **Large Language Models**.
  - **What are LLMs?**: Large Language Models are deep learning models trained on massive amounts of text data to understand and generate human-like language.
  - **Importance**: LLMs power many modern AI applications such as chatbots, machine translation, text summarization, and code generation.
- **Example Models**:
  - **GPT-3**, with 175 billion parameters, is a well-known example of an LLM used in tools like OpenAI's ChatGPT.
  - **Llama, Mixtral, Gemma,** by Meta, Mistral, and Google, respectively.
- **Key Advantages**:
  - Understanding context across long sequences of text.
  - Generating coherent and contextually accurate language outputs.

Autoregressive Models

- **Autoregressive Architecture**:
  - **How GPT works**: GPT predicts the next word in a sequence based on the words that came before it. This makes it **unidirectional** or **autoregressive**.
  - **Transformer Backbone**: GPT uses transformer layers to model long-range dependencies between words, which is a significant improvement over older models like RNNs and LSTMs.
- **Key Features**:
  - **Self-Attention Mechanism**: GPT uses self-attention to focus on important words in a sequence, allowing it to learn context.
  - **Scaling GPT**: GPT-2 had 1.5 billion parameters, and GPT-3 scaled up to 175 billion parameters, greatly improving performance on a wide range of tasks.
- **GPT Variants**:
  - **GPT-2**: 1.5 billion parameters, trained on 8 million web pages.
  - **GPT-3**: 175 billion parameters, trained on hundreds of billions of tokens.

Using BERT in Practice – Huggingface Libraries

- Transformers – https://github.com/huggingface/transformers
- APIs to download and use pre-trained models, fine-tune them on own datasets and tasks
  - Code Sample

```
# Loading BERT
model_class, tokenizer_class, pretrained_weights =  (ppb.DistilBertModel, ppb.DistilBertTokenizer, 'distilbert-base-uncased')

# Load pretrained model/tokenizer
tokenizer = tokenizer_class.from_pretrained(pretrained_weights)
model = model_class.from_pretrained(pretrained_weights)
```

- Provides pretrained models in 100+ languages.
- Use with popular deep learning libraries, PyTorch and TensorFlow,
  - Possible to train / fine-tune models with one, and load it for inference with another

Using BERT in Practice – Huggingface Libraries

- DistilBERT
  - Details: https://medium.com/huggingface/distilbert-8cf3380435b5
  - Teacher-student learning, also called model distillation
    - Teacher: bert-base-uncased
    - Student: dstilBERT - BERT without *the token-type embeddings and the pooler* , and half the layers
  - **"DistilBERT,** has **about half** the total number of parameters of BERT base and retains 95% of BERT's performances on the language understanding benchmark GLUE"

- Sample code of usage for sentiment classification: https://github.com/biplav-s/course-nl/blob/master/l12-langmodel/UsingLanguageModel.ipynb
- Also see: https://huggingface.co/blog/sentiment-analysis-python

HF / DistilBERT and LLM Project

- **Resources**: https://huggingface.co/docs/transformers/en/model_doc/distilbert
- Create notebooks for
  - Data cleaning
  - LLM finetuning
  - Conducting (task) evaluation on
    - LLM
    - Fine-tuned LLM
    - GPT (ChatGPT

Example Pre-Trained Models

1. ALBERT (from Google Research and the Toyota Technological Institute at Chicago) released with the paper ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, by Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut.
2. BART (from Facebook) released with the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov and Luke Zettlemoyer.
3. BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
4. BERT For Sequence Generation (from Google) released with the paper Leveraging Pre-trained Checkpoints for Sequence Generation Tasks by Sascha Rothe, Shashi Narayan, Aliaksei Severyn.
5. CamemBERT (from Inria/Facebook/Sorbonne) released with the paper CamemBERT: a Tasty French Language Model by Louis Martin*, Benjamin Muller*, Pedro Javier Ortiz Suárez*, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah and Benoît Sagot.
6. CTRL (from Salesforce) released with the paper CTRL: A Conditional Transformer Language Model for Controllable Generation by Nitish Shirish Keskar*, Bryan McCann*, Lav R. Varshney, Caiming Xiong and Richard Socher.
7. DeBERTa (from Microsoft Research) released with the paper DeBERTa: Decoding-enhanced BERT with Disentangled Attention by Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen.
8. DialoGPT (from Microsoft Research) released with the paper DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation by Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, Bill Dolan.
9. DistilBERT (from HuggingFace), released together with the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter by Victor Sanh, Lysandre Debut and Thomas Wolf. The same method has been applied to compress GPT2 into DistilGPT2, RoBERTa into DistilRoBERTa, Multilingual BERT into DistilmBERT and a German version of DistilBERT.
10. DPR (from Facebook) released with the paper Dense Passage Retrieval for Open-Domain Question Answering by Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih.
11. ELECTRA (from Google Research/Stanford University) released with the paper ELECTRA: Pre-training text encoders as discriminators rather than generators by Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning.
12. FlauBERT (from CNRS) released with the paper FlauBERT: Unsupervised Language Model Pre-training for French by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, Didier Schwab.
13. Funnel Transformer (from CMU/Google Brain) released with the paper Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing by Zihang Dai, Guokun Lai, Yiming Yang, Quoc V. Le.
14. GPT (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training by Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.
15. GPT-2 (from OpenAI) released with the paper Language Models are Unsupervised Multitask Learners by Alec Radford*, Jeffrey Wu*, Rewon Child, David Luan, Dario Amodei** and Ilya Sutskever**.
16. LayoutLM (from Microsoft Research Asia) released with the paper LayoutLM: Pre-training of Text and Layout for Document Image Understanding by Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou.
17. Longformer (from AllenAI) released with the paper Longformer: The Long-Document Transformer by Iz Beltagy, Matthew E. Peters, Arman Cohan.
18. LXMERT (from UNC Chapel Hill) released with the paper LXMERT: Learning Cross-Modality Encoder Representations from Transformers for Open-Domain Question Answering by Hao Tan and Mohit Bansal.
19. MarianMT Machine translation models trained using OPUS data by Jörg Tiedemann. The Marian Framework is being developed by the Microsoft Translator Team.
20. MBart (from Facebook) released with the paper Multilingual Denoising Pre-training for Neural Machine Translation by Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, Luke Zettlemoyer.
21. MMBT (from Facebook), released together with the paper a Supervised Multimodal Bitransformers for Classifying Images and Text by Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, Davide Testuggine.
22. Pegasus (from Google) released with the paper PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization> by Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter J. Liu.
23. Reformer (from Google Research) released with the paper Reformer: The Efficient Transformer by Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya.
24. RoBERTa (from Facebook), released together with the paper a Robustly Optimized BERT Pretraining Approach by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. ultilingual BERT into DistilmBERT and a German version of DistilBERT.
25. SqueezeBert released with the paper SqueezeBERT: What can computer vision teach NLP about efficient neural networks? by Forrest N. Iandola, Albert E. Shaw, Ravi Krishna, and Kurt W. Keutzer.
26. T5 (from Google AI) released with the paper Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer by Colin Raffel and Noam Shazeer and Adam Roberts and Katherine Lee and Sharan Narang and Michael Matena and Yanqi Zhou and Wei Li and Peter J. Liu.
27. Transformer-XL (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context by Zihang Dai*, Zhilin Yang*, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov.
28. XLM (from Facebook) released together with the paper Cross-lingual Language Model Pretraining by Guillaume Lample and Alexis Conneau.
29. XLM-RoBERTa (from Facebook AI), released together with the paper Unsupervised Cross-lingual Representation Learning at Scale by Alexis Conneau*, Kartikay Khandelwal*, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov.
30. XLNet (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pretraining for Language Understanding by Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le.

https://github.com/huggingface/transformers

Preq-requisites for Understanding Advanced Language Models

- Advanced language models need pre-requisites to understand
  - BERT, Transformers, GPT-2, GPT-3, GPT-4, …

- Understand word representation
- Understand context representation
- Understand machine learning/ neural methods

Commentary: http://jalammar.github.io/illustrated-gpt2/

Course Project

Discussion: Projects

- New: two projects
  - Project 1: model assignment
  - Project 2: single problem/ llm based solving / fine-tuning/ presenting result

HF and DistilBERT

- **Resources**: https://huggingface.co/docs/transformers/en/model_doc/distilbert
- **Usage example**:
  - https://huggingface.co/blog/sentiment-analysis-python

Evaluating with ChatGPT

- University has no policy to provide paid version currently
- Ways to use free:
  - https://www.kdnuggets.com/2023/05/3-ways-access-gpt4-free.html
  - Poe, https://poe.com/ - a reasonable interface for many models

Lecture 17-19: Summary

- We talked about
  - Text Processing
  - Language Models (LMs)
  - Learning for LMs with NN
  - Large LMs
  - Project 2

Concluding Section

About Next Lecture – Lecture 23

Lecture 21: LLMs/ AI and Trust Issues

- ML/LLM - Trust issues

15	Oct 8 (Tu)	Student presentations - project
16	Oct 10 (Th)	ML – NN, Deep Learning
17	Oct 15 (Tu)	Processing Natural Languages/ Language Models
	Oct 17 (Th)	
18	Oct 22 (Tu)	Large Language Models (LLMs) / Foundation Models
19	Oct 24 (Th)	Using LLMs – how and when ?
20	Oct 29 (Tu)	Using LLMs – when not and why?
21	Oct 31 (Th)	Machine Learning – Trust Issues (Methods - Explainability)
	Nov 5 (Tu)	

Lecture 22-23: Decision Problems

- Making simple decisions
  - Maximum Expected Utility (MEU)
- Making complex decisions
  - Markov Decision Processes (MDPs)

17	Oct 15 (Tu)	Processing Natural Languages/ Language Models
	Oct 17 (Th)	
18	Oct 22 (Tu)	Large Language Models (LLMs) / Foundation Models
19	Oct 24 (Th)	Using LLMs – how and when ?
20	Oct 29 (Tu)	Using LLMs – when not and why?
21	Oct 31 (Th)	Machine Learning – Trust Issues (Methods - Explainability)
	Nov 5 (Tu)	
22	Nov 7 (Th)	Making Decisions - Simple
23	Nov 12 (Tu)	Making Decisions - Complex
24	Nov 14 (Th)	Sequential Decision Making: Planning, RL
25	Nov 19 (Tu)	Sequential Decision Making: Planning, RL