Representation of Words in NLP: A Historical Perspective

From Symbols to Contextualized Embeddings

Presented by:
Alex Astolfi
Filippo Momentè
Massimo Stefan

23-10-2023

Index

Introduction to NLP

What is NLP?

Natural Language Understanding

Natural Language Generation

Speech or Voice recognition

Machine translation

Spelling correction and grammar checking

Is a subfield of Artificial Intelligence (AI) that focuses on the interaction between machines and human languages.

AI: The historical evolution

Symbolical

Statistical

Neural

1950s - early 1960s

The Dawn of NLP

The Dawn of NLP

1949: Weaver's Memorandum on Machine Translation (MT)

Warren Weaver's memorandum on Machine Translation marks the inception of NLP research, envisaging the potential of computers in translating languages.

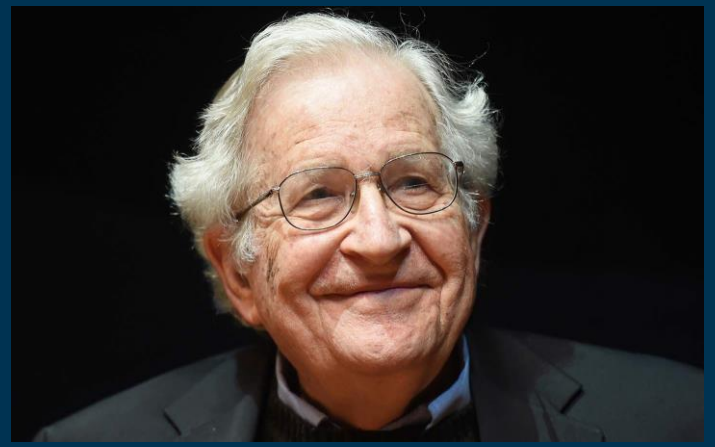
1954: Georgetown-IBM Experiment

The landmark Georgetown-IBM experiment showcases an English-Russian translation algorithm, translating 60 sentences using dictionary-lookup algorithms.

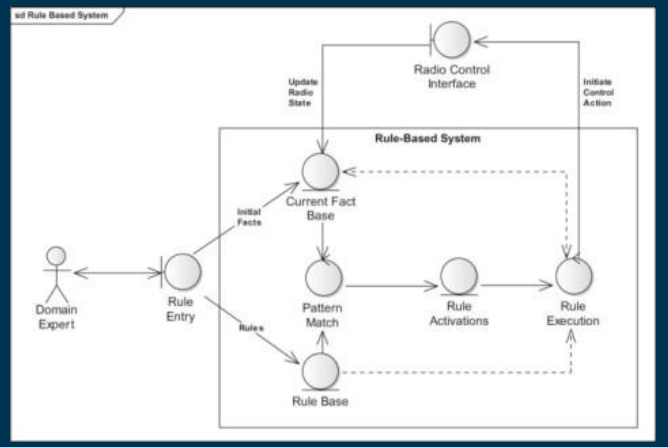
Late 1960s - early 1980s

Transition to Structured Representation

Late 60s: The Emergence of Rule-Based Systems



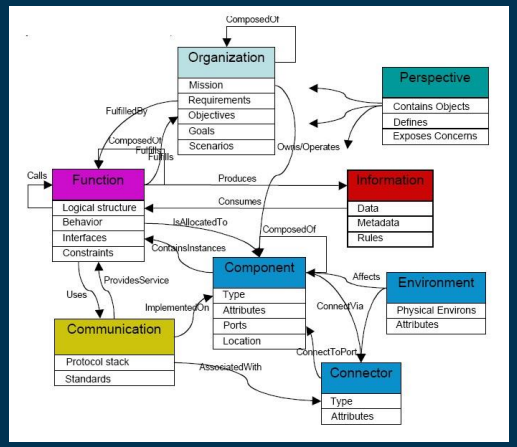
Noam Chomsky's introduction of generative grammars



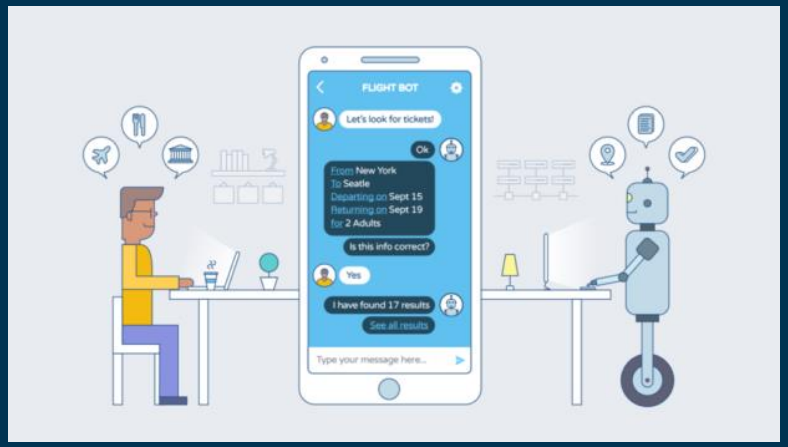
Adoption of rule-based systems

ELIZA, simulated a Rogerian psychotherapist using rule-based techniques (1966)

1970s: Conceptual Ontologies and Early Chatterbots



Birth of conceptual ontologies



Creation of early chatterbots

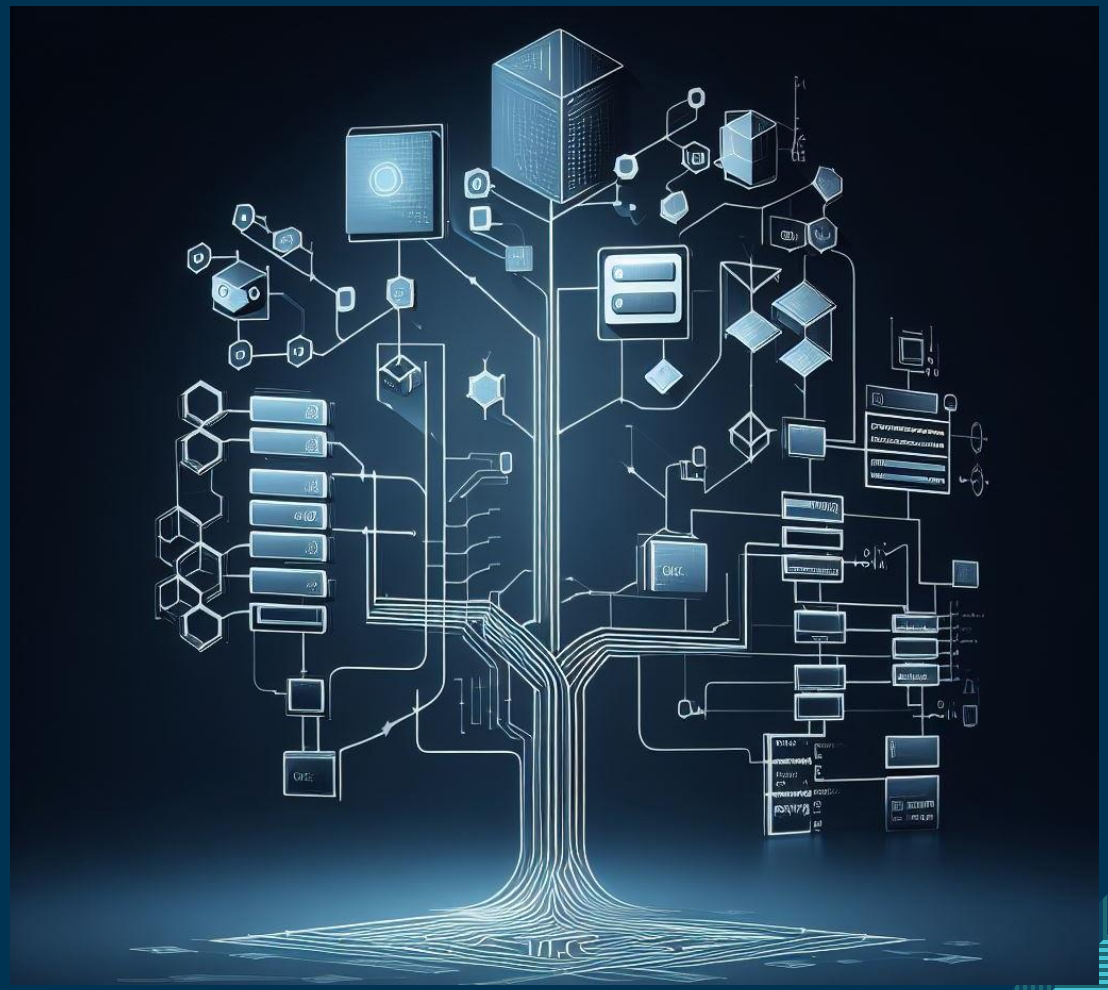
```
ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or
a question mark.
PARRY: People get on my nerves sometimes.
ELIZA: I am not sure I understand you fully.
PARRY: You should pay more attention.
ELIZA: Suppose you should pay more attention.
                                                ...
ELIZA: I am not sure I understand you fully,
PARRY: It's clear to me if not to you.
ELIZA: Do you think its likely that not to I?
PARRY: I went to the races.
ELIZA: Please go on.
PARRY: I gamble on horses.
ELIZA: What does that suggest to you?
PARRY: I went to the races.
                                                ...
PARRY: That's just the way it is.
ELIZA: Suppose that's just the way it were not.
PARRY: So?
ELIZA: I am not sure I understand you fully.
```

PARRY, simulate a person with paranoid schizophrenia (1972)

Early 1980s: Symbolic Approaches in NLP

Dominance of symbolic approaches:

- Rule-based systems
- Formal grammars
- Parsing algorithms
- Predicate logic



Word representation: Symbols

💡 Idea: Words are NOT represented through numbers but through **strings of characters**

☑ Pros:

- Embeddings are highly interpretable
- Explicit knowledge representation
- Allow rule manipulation and customization
- Easily incorporate domain-specific knowledge

❌ Cons:

- Embedding are complex to create
- Require high maintenance
- Lack scalability to RW applications
- Capture limited semantic information
- Cannot effectively resolve ambiguities

🤔 When to use: can be applied on all tasks where explicit and interpretable rules are needed (parsing, basic MT, syntax checking, …) or simple well-defined non-scalable domains

Late 1980s - 2000s

Transitioning from Symbolic to Data-Driven Paradigms in NLP

Late 80s: Statistical language models

Idea: Instead of writing rules, models handle language through **probability distributions** learnt from text data

Computational power

\+

Chomskyan theories

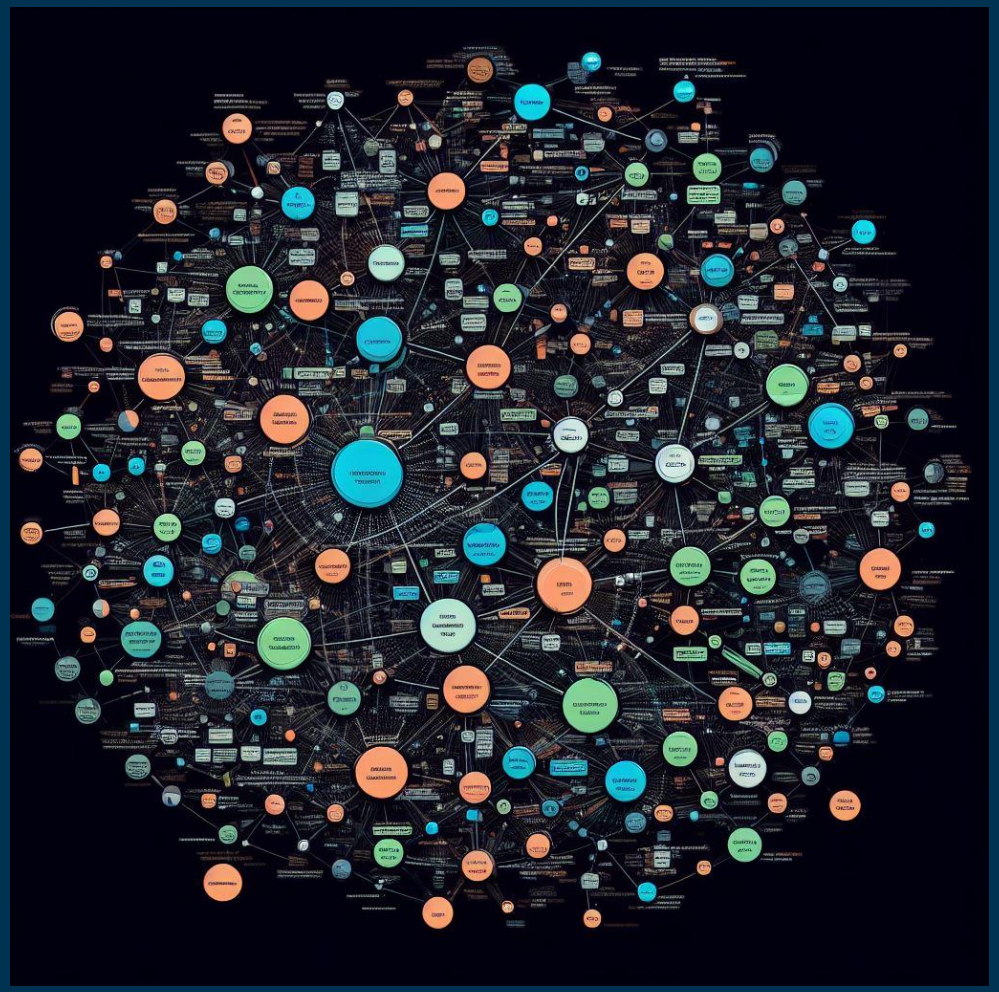
=

Statistical models

1990s: Corpora Linguistics and Statical Approach

- Rise of corpora linguistics and use of datasets
- Success of statistical approaches in machine translation, but still limited amount of data
- Introduction of supervised machine learning techniques.

2000s: Treebanks and Early Machine Learning



- Introduction of Penn Treebank and other annotated large corpora
- New statistical parsing techniques
- Research in semi-supervised/unsupervised learning

Word representation: Bag of words

- Idea: create a **sorted vocabulary of all unique words** present in the corpus and represent each word with a fixed length vector obtained based on counting frequency

Bag of words: One-hot encoding

Idea: each word is represented by a vector of vocabulary length with all 0s, except for its position

Example document: [“today I rained”, “today I ate pizza”]

Vocabulary: [ate, I, it, pizza, rained, today]

Encoding of “ate”: [1,0,0,0,0,0]
Encoding of “pizza”: [0,0,0,1,0,0]

Note: can be extended to sentences (based on words present or not)
Encoding of “today I ate pizza”: [1,1,0,1,0,1]

Bag of words: Count vector

Idea: a matrix D*V represent the corpus, where:

- D is the number of documents
- V is the vocabulary length

Example document 1: [“today I ate pizza”]

Example document 2: [“I ate pizza and you ate pasta”]

Example document 3: [“today it rained”]

Encoding of “I ate pizza and you ate pasta”: [1,2,1,0,1,1,0,0,1]

Vocabulary: [and, ate, I, it, pasta, pizza, rained, today, you]

	and	*ate*	*I*	*it*	*pasta*	*pizza*	*rained*	*today*	*you*
D1	0	1	1	0	0	1	0	1	0
D2	1	2	1	0	1	1	0	0	1
D3	0	0	0	1	0	0	1	1	0

Bag of words: TF-IDF vectors

Idea: produce representations which take into account the importance of a word across the document

Each entry of a matrix considers:

1. **Term frequency (TF)**: how rare or common a term is in the words of the collection of documents
2. **Inverse Document Frequency (IDF)**: how rare or common a term is in a collection of documents

These 2 numbers are then multiplied together.

Example document 1: [“today I ate pizza”]

Example document 2: [“I ate pizza and you ate pasta”]

Example document 3: [“today it rained”]

Vocabulary: [and, ate, I, it, pasta, pizza, rained, today, you]

Example: (ate, D1)

TF = ¼

IDF = $\log(\frac{3}{2})$

Entry (1,2) = ¼ * $\log(\frac{3}{2}) \approx 0.044$

Word representation: Bag of words

Idea: create a **sorted vocabulary of all unique words** present in the corpus and represent each word with a fixed length vector obtained based on counting frequency

☑ Pros:

- Encodings are simple and easy to implement
- Sparse vectors (efficient storage)
- Scalable
- Do NOT require training (based on counting)

❌ Cons:

- High dimensionality leading to overfitting
- Fail to capture order and grammatical structure
- Don’t yield any semantic information
- Difficult to introduce OOV words

When to use: all downstream tasks which do not require structure or semantic understanding (classification, information retrieval, keyword extraction, …)

Word representation: Co-occurrences

Idea: a V*V matrix where each entry corresponds to a couple of words and it is the **number of times in which the two words occur together**, within a fixed distance called **context window**

Example document 1: [“today I ate pizza”]

Example document 2: [“I ate pizza and you ate pasta”]

Example document 3: [“today it rained”]

Vocabulary: [and, ate, I, it, pasta, pizza, rained, today, you]

Context window: 2

Co-occurrence (I, ate): 2

Co-occurrence (end, ate): 0

	and	*ate*	*I*	*it*	*pasta*	*pizza*	*rained*	*today*	*you*
and	1	0	0	0	0	1	0	0	1
ate	0	3	2	0	1	2	0	0	1
I	0	2	2	0	0	0	0	1	0
it	0	0	0	1	0	0	1	1	0
pasta	0	1	0	0	1	0	0	0	0
pizza	1	2	0	0	0	2	0	0	0
rained	0	0	0	1	0	0	1	0	0
today	0	0	1	1	0	0	0	2	0
you	1	1	0	0	0	0	0	0	1

Note: can be reduced to a matrix V*N (removing function words) and its representation can be reduced by applying mathematical methods

Word representation: Co-occurrences

💡 Idea: a V*V matrix where each entry corresponds to a couple of words and it is the **number of times in which the two words occur together**, withing a fixed distance called **context window**

☑ Pros:

- No training required
- Low dimensionality
- Can be used as features for ML algorithms
- Capture semantic similarity
- Introduce context understanding

❌ Cons:

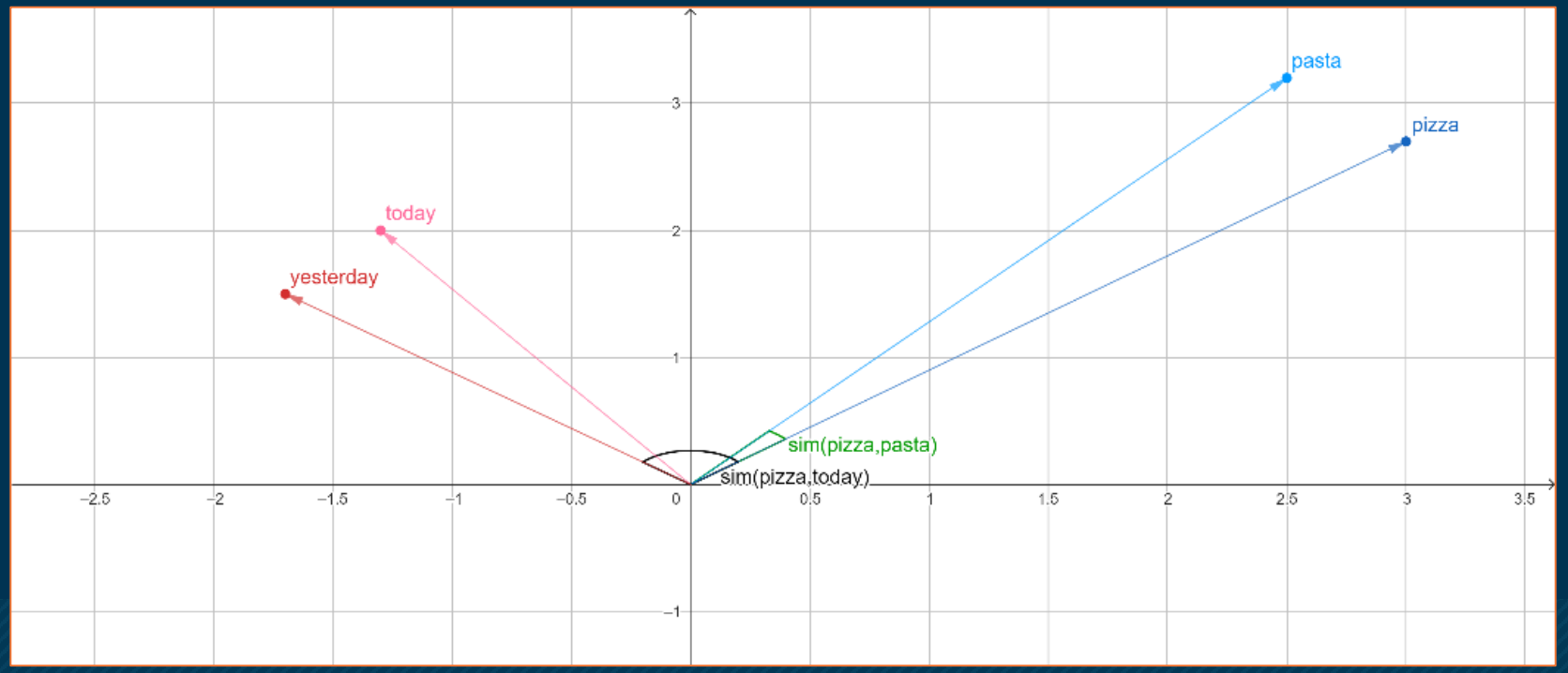
- Co-occurrences matrixes are large
- Require memory to be stored
- Decomposition can be computationally expensive
- High influence of context window size
- Word order not encoded

🧐 When to use: tasks where capturing semantic information and similarity between words is important (analogy, MT, …) or clustering tasks, word sense disambiguation, NER, …

Excursus: vector spaces and cosine similarity

- Idea 1: represent each **word with a point** in an N-dimensional space, the **vector** corresponding to the word **is an arrow from the origin** of the axes to the point

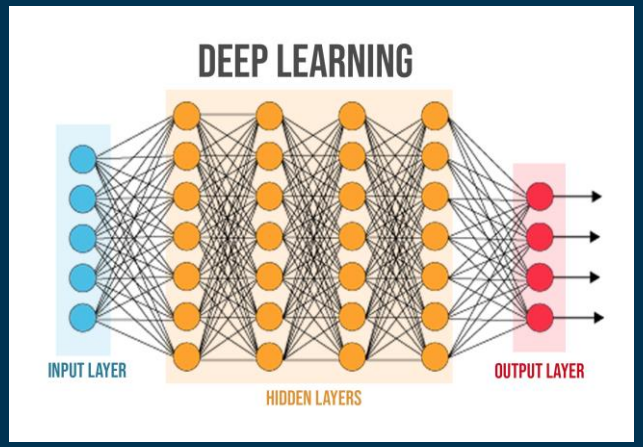
- Idea 2: the **cosine similarity** is a measure of the angle between them and represent their **similarity**



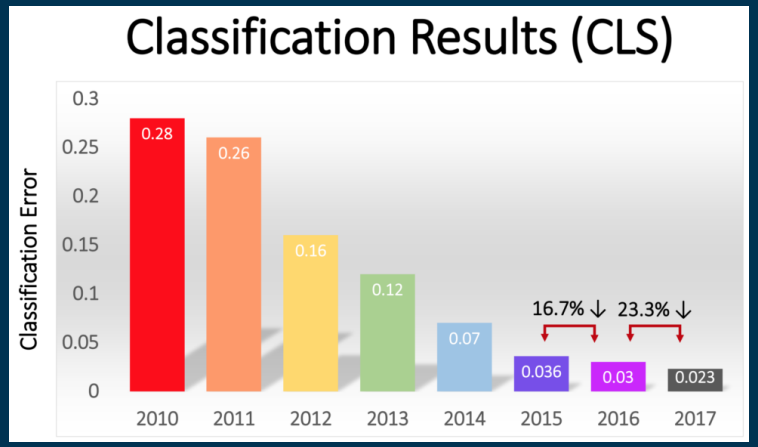
2010s

The Era of Neural Embeddings

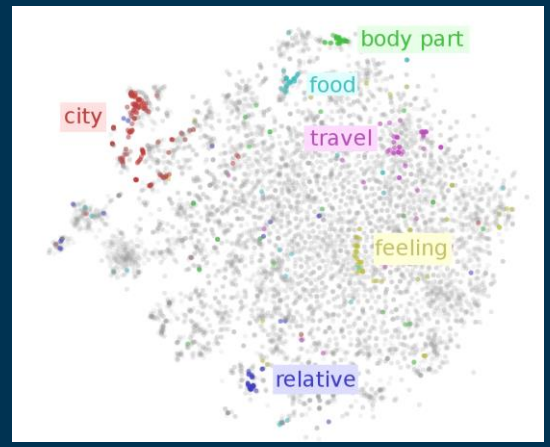
2010s: The Deep Learning Revolution in NLP



NNs are scaled up to multiple layers, the Deep Learning (DL) era begins



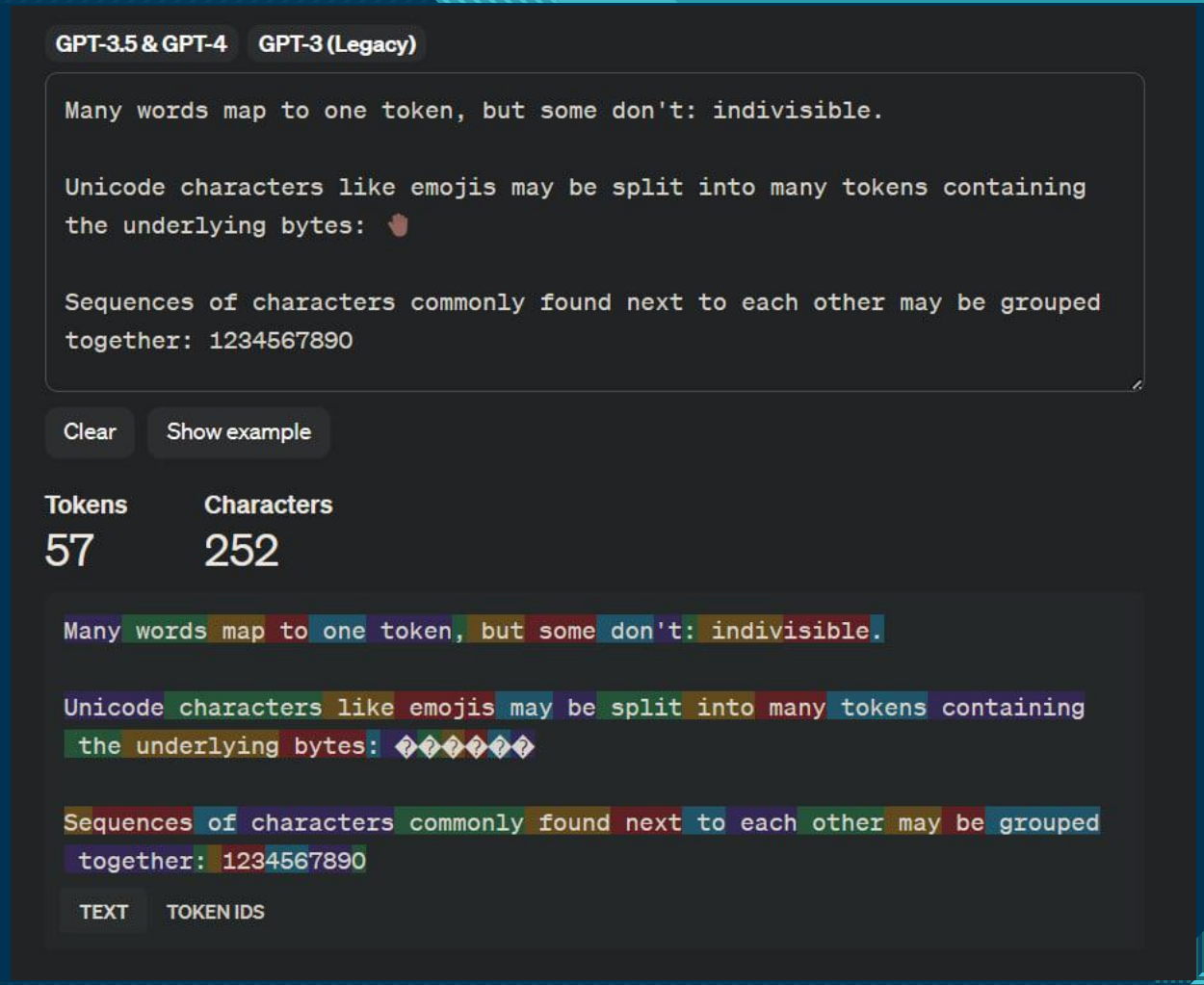
DL impact on the ImageNet challenge



Neural Embeddings can represent meaning, shift from counting to prediction

Tokenization

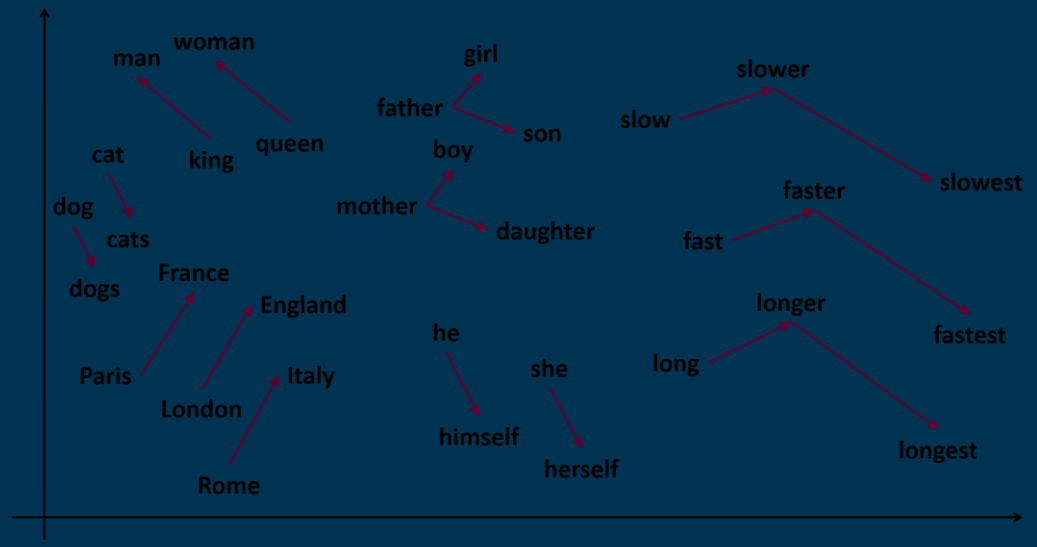
- Idea: break down a piece of text into smaller units, called **tokens**
- Improvements: from word level to sub-word level the tokenizer can handle OOV words and work for multiple languages



Word2Vec (2013): The First Strong Word Embedding Model

Idea:

1. Train a classifier (self-supervised) to predict current or context words
2. Discard the classifier
3. Use the learned weights as word embeddings



Algorithms proposed:

- **CBOW**: predict the current word given the context
- **SkipGram**: predict the context words given the current word

Fasttext (2017): Leverage Sub-Word Embeddings

💡 Idea:

1. Learn representations for character n-grams
2. Represent words as the sum of the n-gram vector

🔥 Improvement: account for internal structure (e.g. “dog”, “dogs”, “dogcatcher”, …)

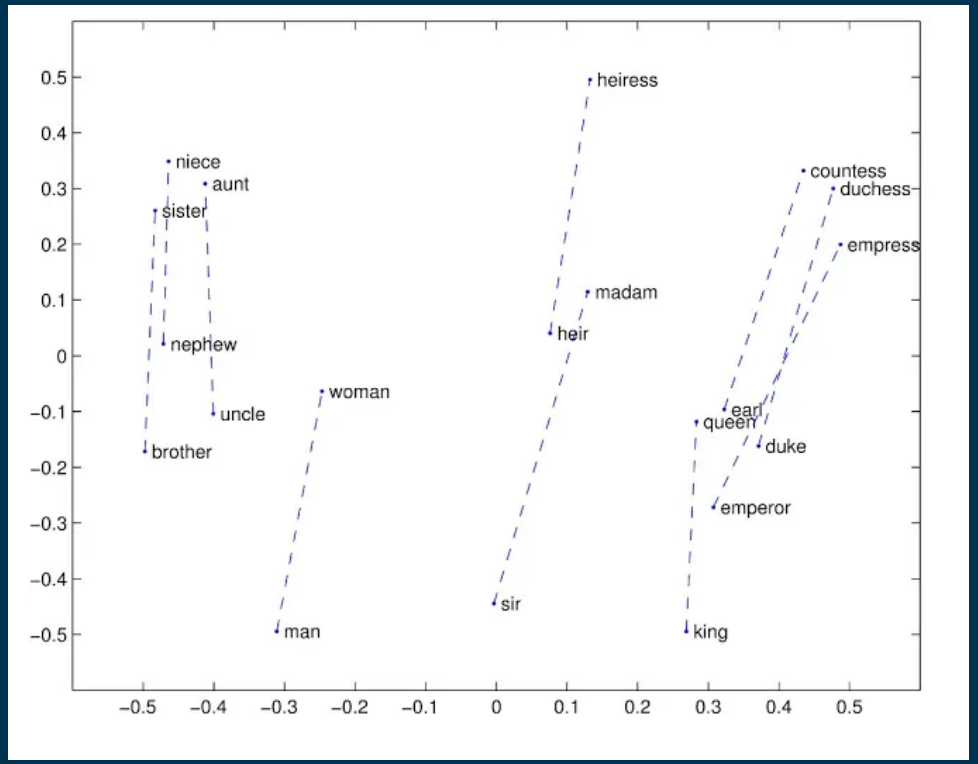
🔎 How: Each word is represented as a bag of character n-grams

📝 Example: word=“where” and n=3 => {“<wh”, “whe”, “her”, “ere”, “re>”, “<where>”}

GloVe (2014): Leverage Co-occurrence matrix

- Idea:

1. Generate word embeddings by aggregating word-word co-occurrence matrix from a corpus
2. The resulting embeddings show interesting linear substructures of the word in vector space



Word representation: Static embeddings

💡 Idea: are generated by training a model on a large corpus of text and **learning the distributional properties** of words

☑ Pros:

- Really high accuracy compared to previous models
- More interpretable than contextualized (see later)
- Are stable and fixed regardless of the context

❌ Cons:

- Do not capture the context
- Are less expressive than contextualized embeddings
- Ambiguous words (polysemy, ambiguity, homonymy) are not fully represented

When to use: task-specific data or baseline for downstream tasks (transfer learning) or tasks where semantic change and ambiguity are not present

CoVe (2017): Leverage Bi-directional LSTM

Idea:

1. Start from GloVe representations
2. Train an LSTM model for Machine Translation

Note: can be used for transfer learning concatenating its representation to the corresponding GloVe representations

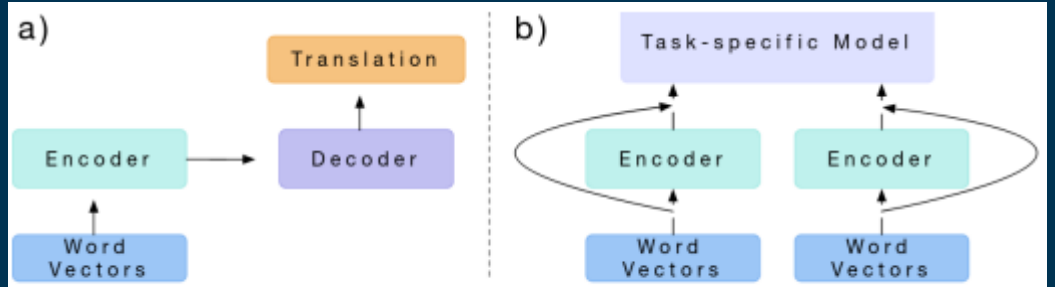
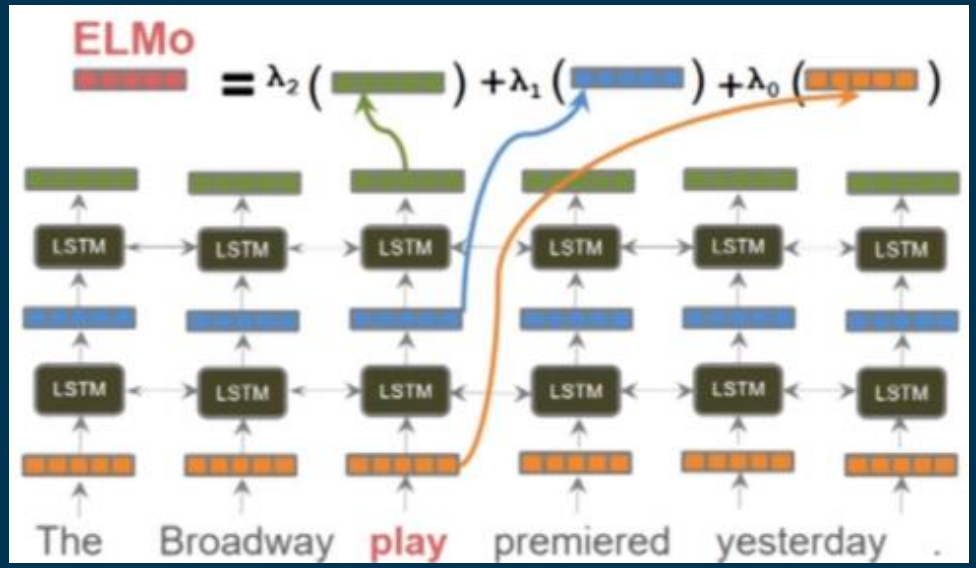


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

ELMo (2018): Combine different embeddings from LSTMs

Idea:

1. Start from word vectors computed via char-CNN
2. Values go through 2-layer bidirectional LSTM
3. The final embedding is a task-specific weighting of all the layer outputs

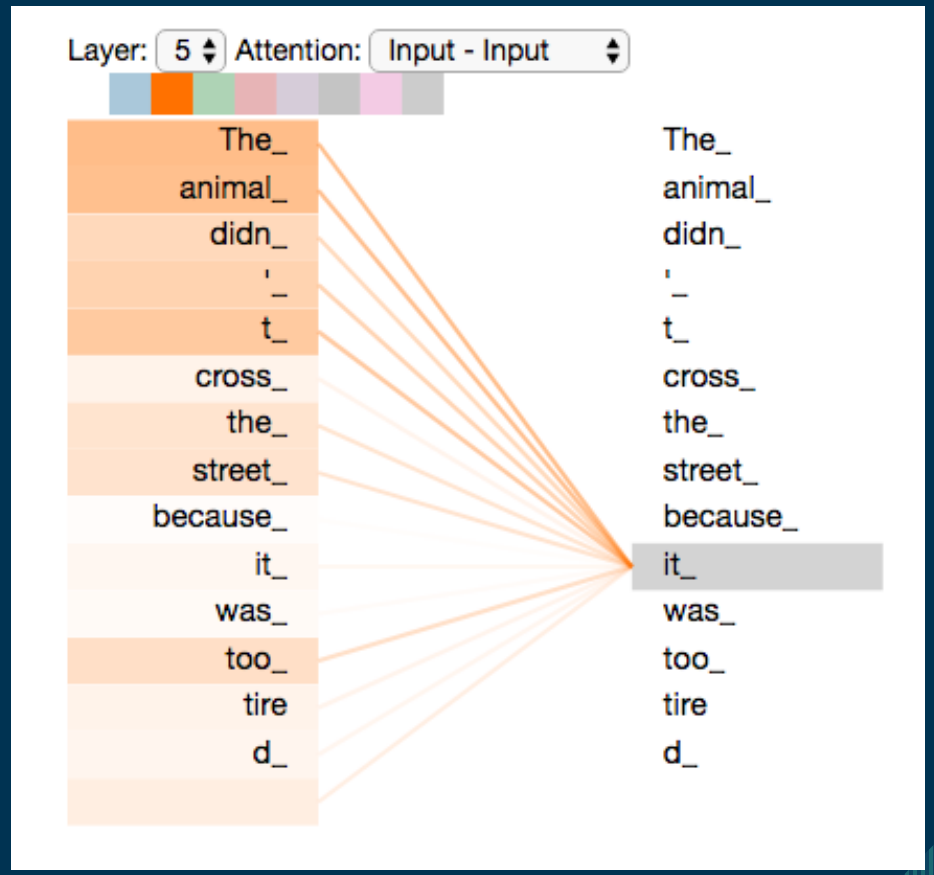


The attention and self-attention mechanisms

💡 Idea: focus on specific parts of the input sequences, assigning a weight to each word

But:

- **Attention** is used to attend to different parts of another sequence when making predictions generating another sentence
- **Self-attention** is used to attend to different parts of the input sequence when making predictions (refers to the same sequence which is currently being embedded)



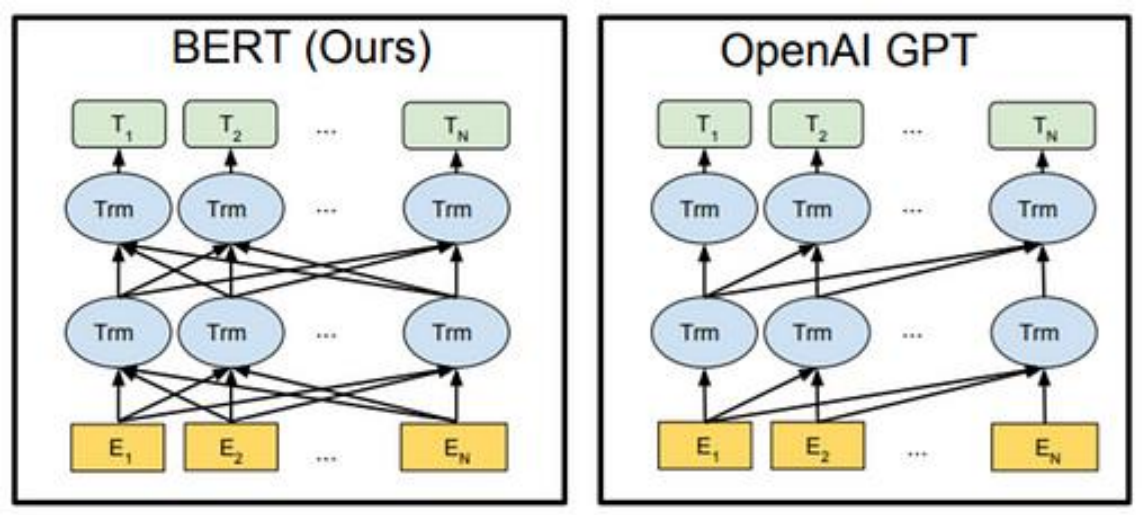
BERT vs GPT (2018): Transformer-Based Architectures

BERT

- Trained on **Masked Language Modeling** (MLM), where some tokens are randomly masked and the model is trained to predict the original value based on the context
- Use **bidirectional flow**, which allows it to use context from both directions during processing

GPT

- Trained on **Causal Language Modeling** (CLM), where the model is trained to predict the next token in a sequence given all previous tokens
- Use **unidirectional flow**, which allows it to use only the context from the previous tokens



Word representation: Contextualized embeddings

Idea: are generated by considering the context in which words are present

☑ Pros:

- Capture the meaning of words in context
- Can handle polysemy, ambiguity, homonymy, …
- Can be easily used for transfer learning

✗ Cons:

- Computationally expensive to generate
- Require large amounts of training data
- May work worse than static embeddings when context is not important

When to use: tasks where a deep understanding of nuances and context-specific differences is essential

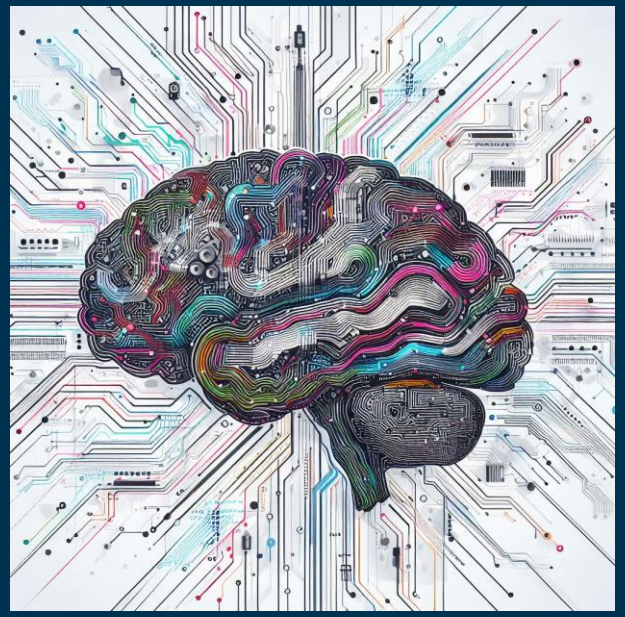
2020s

Advancements in Word Embedding Techniques and Interpretability

2020s: Current lines of research

Neurosymbolic approaches

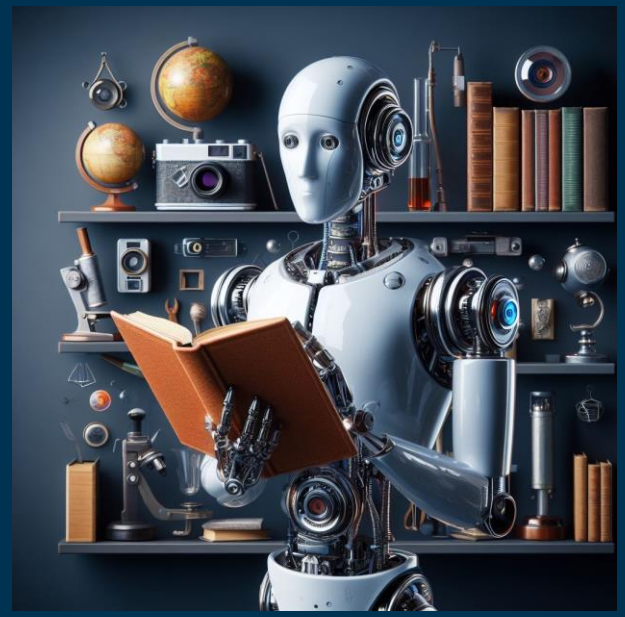
Neural strategies + Older symbolic representations



- Improvement: higher interpretability and generalization capabilities

Multimodal embeddings

Textual information + Visual information + …

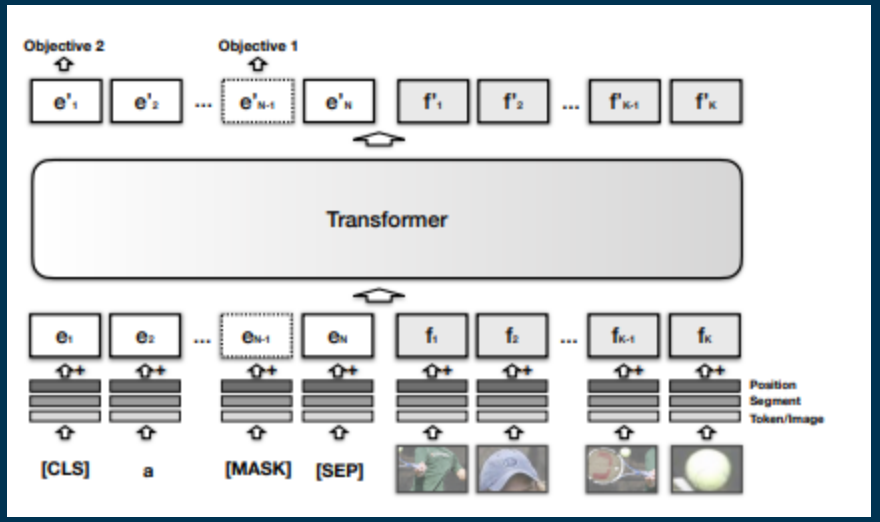


- Improvement: enhance the contextualization of the textual embeddings

VisualBERT (2020)

Idea:

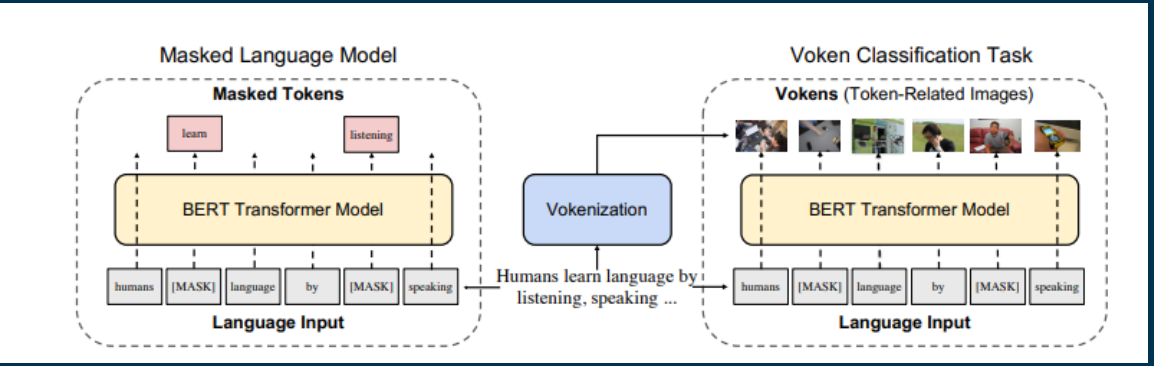
1. Obtain output from BERT and from an object detector
2. Combine them together
3. Pass the result through a transformer to learn alignment
4. Take the output as embedding



Vokenization (2020)

Idea:

1. Split images into vokens: token-related subparts
2. Train a BERT model on Masked Language Modeling and Voken Classification
3. Take final output as embeddings



Word representation: Multimodal embeddings

💡 Idea: are generated by considering other modalities (images, video, audio, …) to enhance their meaning

☑ Pros:

- Better performance (usually)
- Can handle multimodal data
- Via transfer learning can be fine-tuned for other tasks

❌ Cons:

- Really computationally expensive
- Require a huge amount of (coherent) data between modalities
- May not work well on abstract concepts

🤓 When to use: when we have coherent data from multiple sources available (and a lot of computational power)

Recap

Word representation: Comparisons

Symbolic representations

Statistical representations:

Bag of words

Statistical representations:

Co-occurrence matrixes

Neural representations:

Static

Neural representations:

Contextualized

Neural representations:

Multimodal

References

https://www.samyzaf.com/ML/nlp/nlp.html
https://arxiv.org/pdf/1607.04606.pdf (Fasttext)
https://aclanthology.org/D14-1162/ (GloVe)
https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b
http://ai.stanford.edu/blog/contextual/
https://arxiv.org/pdf/1708.00107.pdf (CoVe)
https://arxiv.org/pdf/1802.05365.pdf (ELMo)
https://arxiv.org/pdf/1810.04805.pdf (BERT)
https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (GPT)
Embed2Sym - Scalable Neuro-Symbolic Reasoning via Clustered Embeddings (kr.org) (Emb2Sym)
1908.03557.pdf (arxiv.org) (VisualBERT)
arxiv.org/pdf/2010.06775.pdf (Vokenization)
https://www.icts.res.in/sites/default/files/media/media-library/NLPIntro.pdf (historical overview)
https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbcb937ebce (historical overview)
https://aclanthology.org/www.mt-archive.info/90/MTNI-1999-Hutchins.pdf (Weaver's memorandum)
https://aclanthology.org/www.mt-archive.info/Garvin-1967.pdf (Georgetown experiment)
https://dl.acm.org/doi/pdf/10.1145/365153.365168 (ELIZA)
https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/abs/on-the-generality-of-parry-colbys-paranoia-model/CAEAAACBB8894DF9BB0ABC7CD7519D7C# (PARRY)
https://www.rfc-editor.org/rfc/rfc439.html (Eliza-Parry conversation)
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/ (statistical word representation)
https://arxiv.org/abs/1301.3781 (Word2Vec)

Thank You

Presented by:

Alex Astolfi
Filippo Momentè
Massimo Stefan