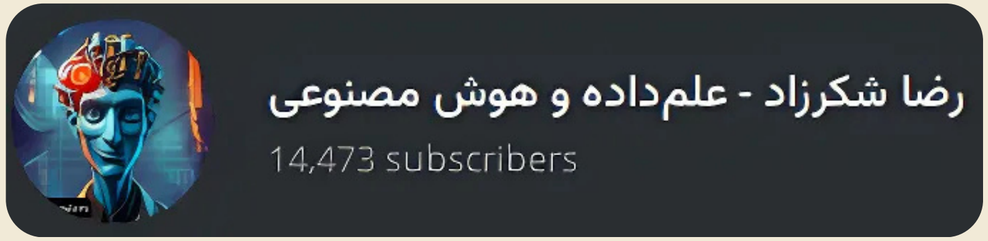
Reza Shokrzad

March 2025

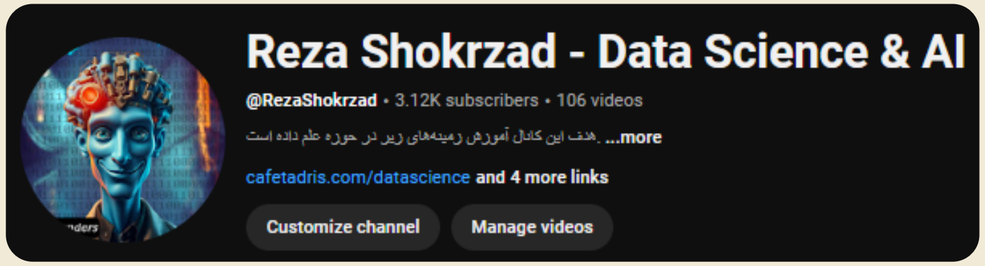
NLP Workshop

Session 2 - Token Embedding

Communities



@DSLanders



@RezaShokrzad

Content

- History of word embedding
- Bag-of-words Model (BOW)
- tf–idf (TF*IDF)
- One Hot Encoding
- Word2Vec
- GloVe
- FastText
- ELMo
- BERT
- GPT
- Claude

Word Embedding

Word embeddings are vector representations of words that capture semantic meaning.

Timeline

1954 BOW

1972 tf-idf

1980s One Hot Encoding

2013 Word2Vec

2014 GloVe

2016 FastText

2018 ELMo

2018 BERT

2018 GPT

2023 Claude

Bag-of-words Model (BOW)

- Zellig Harris's 1954 article on Distributional Structure
- Represents text as unordered word counts.
- An early, foundational NLP method.

https://www.tandfonline.com/doi/pdf/10.1080/00437956.1954.11659520

https://en.wikipedia.org/wiki/Bag-of-words_model

BOW Model

Corpus

A collection of text documents

↓

Tokenize

Divide the text into smaller units called tokens, usually words or phrases

↓

Count word frequencies

Create a vocabulary of unique words and count the number of times each word appears in each document.

↓

Encode the data

Encoding the text data as numerical values by creating a vector for each document, with each element of the vector representing the frequency count of a particular word in the document

Corpus:

The dog is happy. The child makes the dog happy. The dog makes the child happy

↓

Tokenization:

D1: [The] [dog] [is] [happy]
D2: [The] [child] [makes] [the] [dog] [happy]
D3: [The] [dog] [makes] [the] [child] [happy]

↓

Documents	Counting word frequencies
D1	the: 1, dog: 1, is: 1, happy: 1
D2	the: 2, dog: 1, makes: 1, child: 1, happy: 1
D3	the: 2, child: 1, makes: 1, dog: 1, happy: 1

↓

Encode	child	dog	happy	is	makes	the	BoW Vector representations
D1	0	1	1	1	0	1	**[0,1,1,1,0,1]**
D2	1	1	1	0	1	2	**[1,1,1,0,1,2]**
D3	1	1	1	0	1	2	**[1,1,1,0,1,2]**

Bag-of-words Model (BOW)

- **Pros:**
  - Simple to implement.

- **Cons:**
  - Ignores word order and syntactic context.
  - Lacks semantic understanding and nuance.

tf–idf (TF*IDF)

term frequency–inverse document frequency

- Developed in the 1970s as part of early information retrieval systems.
- Designed to quantify word importance by balancing term frequency with inverse document frequency, highlighting words that are more informative.
- It became a cornerstone for document ranking and text analysis, influencing modern search engines and NLP methodologies.

https://www.emerald.com/insight/content/doi/10.1108/eb026526/full/html

https://en.wikipedia.org/wiki/Tf%E2%80%93idf#:~:text=play%20it%20is.-,Definition,document%20or%20a%20web%20page.

tf–idf Model

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

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

Scikit-Learn

- $\text{IDF}(t) = \log\frac{1+n}{1+\text{df}(t)} + 1$

Standard notation

- $\text{IDF}(t) = \log\frac{n}{\text{df}(t)}$

tf–idf (TF*IDF)

- Pros:
  - Simple and efficient to compute and implement.
  - A strong baseline for document ranking and relevance in IR.

- Cons:
  - Ignores word order and semantic context.
  - Generates high-dimensional, sparse representations that may miss subtle nuances.

One Hot Encoding

- represents each word as a binary vector where one element is “hot” (set to 1) and all others are 0.
- It’s one of the simplest and earliest methods for converting categorical (or text) data into a numerical format for ML.

https://en.wikipedia.org/wiki/One-hot

One Hot Model

As an example, say that we were to use one-hot encoding for the following sentences:

s1. ***"This is sentence one."***
s2. ***"Now, here is our sentence number two."***

The vocabulary from the two sentences is:

vocabulary = {"here": 0, "is": 1, "now": 2, "number": 3, "one": 4, "our": 5, "sentence": 6, "this": 7, "two": 8}

The two sentences represented by one-hot vectors are:

indices **words**

```
      0  1  2  3  4  5  6  7  8
s1: [[0, 0, 0, 0, 0, 0, 0, 1, 0], - "this"
     [0, 1, 0, 0, 0, 0, 0, 0, 0], - "is"
     [0, 0, 0, 0, 0, 0, 1, 0, 0], - "sentence"
     [0, 0, 0, 0, 1, 0, 0, 0, 0]] - "one"

s2: [[0, 0, 1, 0, 0, 0, 0, 0, 0], - "now"
     [1, 0, 0, 0, 0, 0, 0, 0, 0], - "here"
     [0, 1, 0, 0, 0, 0, 0, 0, 0], - "is"
     [0, 0, 0, 0, 0, 1, 0, 0, 0], - "our"
     [0, 0, 0, 0, 0, 0, 1, 0, 0], - "sentence"
     [0, 0, 0, 1, 0, 0, 0, 0, 0], - "number"
     [0, 0, 0, 0, 0, 0, 0, 0, 1]] - "two"
```

One Hot Encoding

- Pros:
  - Simple to implement and understand.
  - Creates clear, non-ordinal representations for categorical data.

- Cons:
  - Leads to high-dimensional, sparse vectors, especially with large vocabularies.
  - Fails to capture semantic relationships or similarities between words.

One Hot Encoding

Gender	Location
Male	South
Female	North
Male	West
Male	East

Gender_Male	Gender_Female	Location_South	Location_North	Location_West	Location_East
1	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	0	0	0	0	1

Word2Vec

- **Developed by Mikolov et al. in 2013, Word2Vec marked a breakthrough in creating dense, low-dimensional word representations.**
- **It produces embeddings that capture semantic relationships between words.**
- **The method quickly became popular for its efficiency and effectiveness in various NLP tasks.**

https://arxiv.org/abs/1301.3781

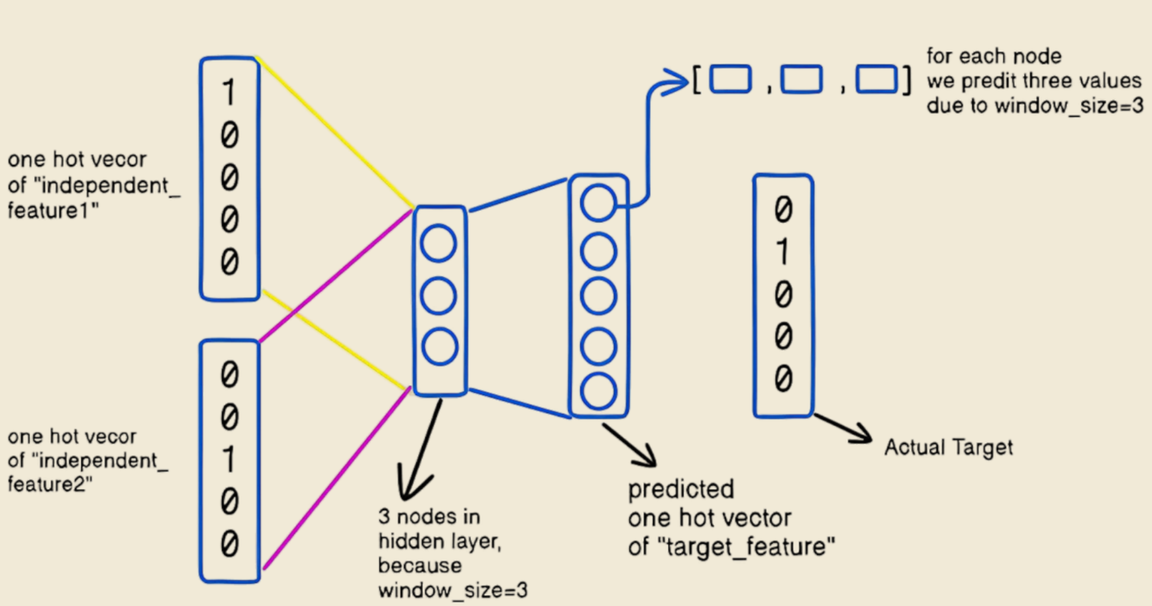
https://en.wikipedia.org/wiki/Word2vec#:~:text=Word2vec%20is%20a%20technique%20in,text%20in%20a%20large%20corpus.

Word2Vec

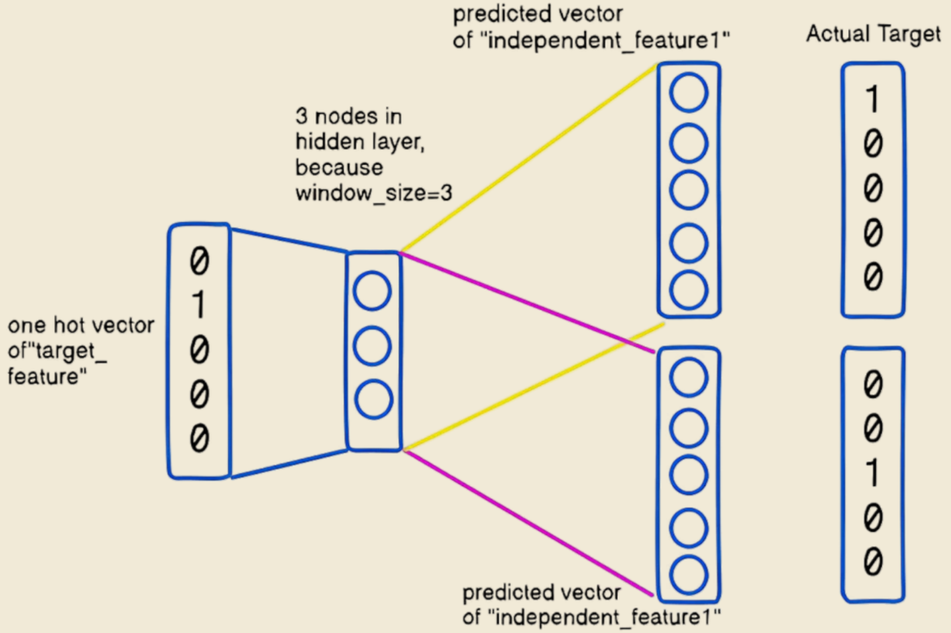
- **Architecture Variants:**
  - **CBOW (Continuous Bag-of-Words):** Predicts a target word based on its surrounding context words.
  - **Skip-gram:** Predicts surrounding context words given a target word.
- **Training Techniques:**
  - Utilizes a shallow neural network with one hidden layer.
  - Employs optimization strategies like negative sampling or hierarchical softmax to improve efficiency.

Word2vec

CBOW



Skip-gram



Example

Cyrus the Great, the legendary king of Persia, founded the vast Achaemenid Empire. As a wise and just ruler, he conquered Babylon and freed the Jews, earning a place in history as a liberator. His policies of tolerance and governance set a model for future leaders. Unlike tyrants, he ruled with respect for diverse cultures, strengthening his vast dominion. His legacy as an emperor of Persia, wisdom and justice endures in history and legend.

Word2Vec

- Pros:
  - **Efficient & Scalable:** Can be trained on large datasets quickly.
  - **Semantic Richness:** Effectively captures relationships and similarities between words.
  - **Simplicity:** The model's straightforward architecture makes it accessible and easy to implement.

- Cons:
  - **Context Insensitivity:** Generates a single embedding per word, ignoring different meanings in varied contexts.
  - **Data Requirement:** High-quality embeddings require large amounts of training data.
  - **Loss of Order:** The model overlooks the sequential order of words within the context window.

GloVe

- Introduced by researchers at Stanford in 2014 (Pennington et al.) as Global Vectors for Word Representation.
- **Core Idea:** Leverages global word-word co-occurrence statistics from large corpora to capture semantic relationships.
- **Approach:** Combines benefits of count-based methods with predictive models to produce dense word embeddings.

https://github.com/stanfordnlp/GloVe

https://aclanthology.org/D14-1162/

https://en.wikipedia.org/wiki/GloVe

GloVe

- **Objective:** Uses a weighted least squares regression to factorize a word co-occurrence matrix, modeling the logarithm of co-occurrence probabilities.
- **Mechanism:** Learns word vectors such that their differences approximate the log ratios of co-occurrence probabilities, capturing meaningful linear substructures.
- **Data Dependency:** Relies on aggregated global statistics, rather than solely on local context windows, to build its embeddings.

GloVe

- Pros:
  - **Global Context:** Effectively captures overall statistical patterns from the corpus.
  - **Semantic Richness:** Produces embeddings that perform well on analogy and similarity tasks.
  - **Efficiency:** Can be computed relatively efficiently once the co-occurrence matrix is built.

- Cons:
  - **Static Representations:** Generates one embedding per word, ignoring context-dependent meanings.
  - **Memory Intensive:** Building and storing the full co-occurrence matrix can be resource-heavy for very large corpora.
  - **Hyperparameter Sensitivity:** Performance can vary based on the choice of weighting functions and other parameters.

FastText

- Introduced by Facebook AI Research in 2016.
- **Innovation:** Extends Word2Vec by incorporating character n-grams, capturing subword and morphological information.
- **Advantage:** Improves handling of rare and out-of-vocabulary words, especially in morphologically rich languages.

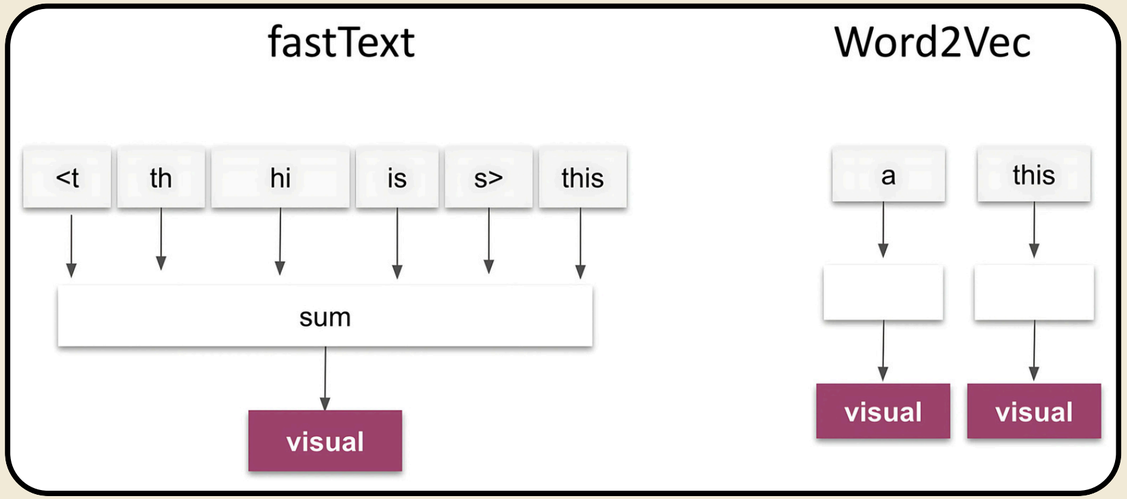
https://github.com/facebookresearch/fastText

https://arxiv.org/abs/1607.04606

https://en.wikipedia.org/wiki/FastText

FastText

- **Architecture:** Builds on the skip-gram model, representing each word as a sum of its character n-gram vectors.
- **Mechanism:** Decomposes words into overlapping character n-grams; these subword embeddings are aggregated to form the final word representation.
- **Efficiency:** Retains scalability and efficiency similar to Word2Vec while enhancing linguistic richness.



FastText

- Pros:
  - **Robustness:** Effectively handles rare words and out-of-vocabulary terms.
  - **Linguistic Nuance:** Captures subword information, beneficial for morphologically complex languages.
  - **Scalable:** Maintains efficient training similar to traditional Word2Vec.

- Cons:
  - **Complexity:** Increased computational overhead due to n-gram processing.
  - **Hyperparameter Sensitivity:** Performance depends on careful tuning of n-gram size and other parameters.
  - **Aggregation Limitations:** The summing of n-gram vectors may sometimes obscure word order nuances.

ELMo

- **Contextualized Embeddings:** Introduced in 2018 by the Allen Institute for AI, ELMo (Embeddings from Language Models) generates context-sensitive word representations.
- **Deep Representations:** It leverages deep, bidirectional LSTM language models to capture rich, dynamic word meanings.
- **Performance Boost:** ELMo significantly improved NLP task performance by providing embeddings that adapt to the context in which words appear.

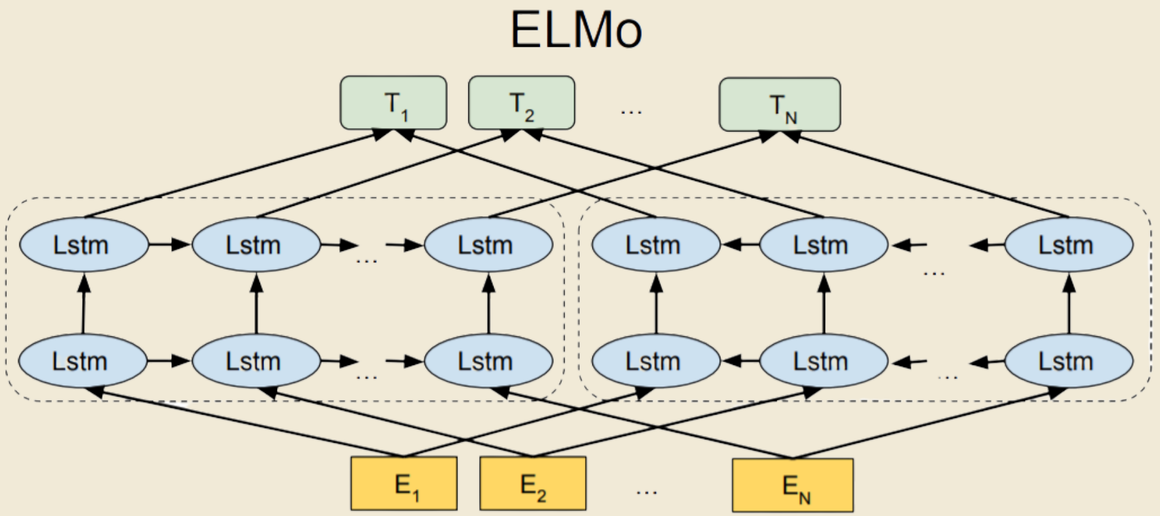
https://github.com/allenai/allennlp-models

https://arxiv.org/abs/1802.05365v2

https://en.wikipedia.org/wiki/ELMo

ELMo

- **Architecture:** Uses a multi-layer bidirectional LSTM to process text both forwards and backwards.
- **Representation:** Generates word embeddings as a weighted combination of internal LSTM states, making them sensitive to the full sentence context.
- **Training Approach:** Trained on large corpora with a language modeling objective, capturing both syntax and semantics in context.



ELMo

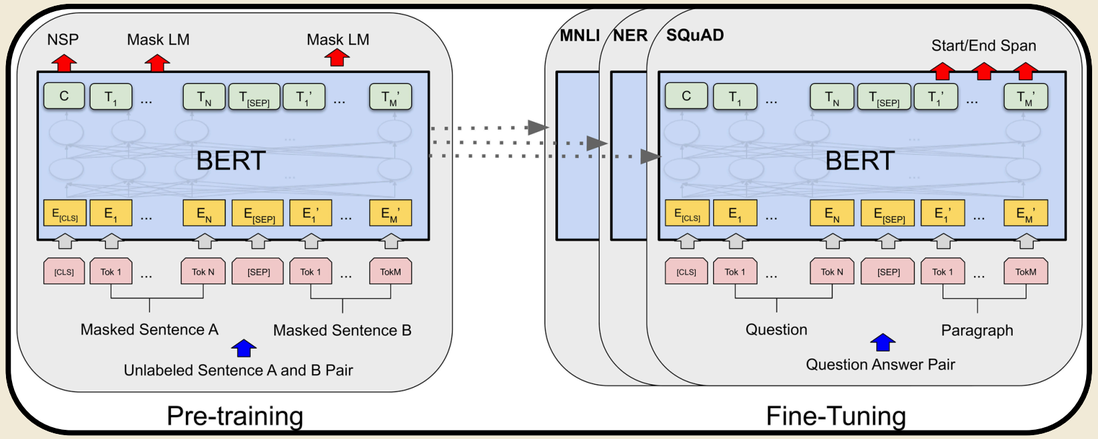
- Pros:
  - **Context-Aware:** Produces embeddings that vary with context, capturing nuanced word meanings.
  - **Task Versatility:** Enhances performance across diverse NLP tasks like sentiment analysis, question answering, and text classification.
  - **Rich Representations:** Utilizes deep architectures to incorporate multiple levels of linguistic information.

- Cons:
  - **Computationally Intensive:** Requires significant computational resources and memory due to its deep, recurrent architecture.
  - **Slower Inference:** LSTM-based models can be slower compared to newer transformer-based methods.
  - **Complex Integration:** Extracting and weighting multiple LSTM layers can add complexity when integrating into downstream models.

BERT

Bidirectional encoder representations from transformers

- **Processes context from both left and right simultaneously.**
- **Masked Language Modeling:** Pre-trained by predicting randomly masked tokens in a sentence.
- **Next Sentence Prediction:** Learns relationships between sentences to capture document-level context.
- **Versatile:** Excels across various NLP tasks with fine-tuning on specific datasets.
- **Pre-training Paradigm:** Sets a new standard for transfer learning in NLP.



https://github.com/google-research/bert

https://arxiv.org/abs/1810.04805

https://en.wikipedia.org/wiki/BERT_(language_model)

GPT

Generative pre-trained transformer

- **GPT-1 (2018):** Introduced the transformer-based generative pre-training paradigm, laying the groundwork for unsupervised learning in NLP. (117 million parameters)
- **GPT-2 (2019):** Scaled up model size to showcase strong text generation capabilities, sparking discussions on ethical implications. (1.5 billion parameters)
- **GPT-3 (2020):** Leveraged 175 billion parameters to excel in few-shot learning, enabling a wide range of language tasks with minimal fine-tuning.
- **GPT-4 (2023):** Improved context understanding and reasoning, with potential multimodal capabilities and enhanced safety measures. (1.7 trillion parameters)

https://en.wikipedia.org/wiki/Generative_pre-trained_transformer

https://en.wikipedia.org/wiki/GPT-2

https://en.wikipedia.org/wiki/GPT-3

https://en.wikipedia.org/wiki/GPT-4

Claude

- **Anthropic's Model:** Developed by Anthropic as a next-generation conversational AI.
- **Constitutional AI:** Emphasizes ethical alignment and safety through constitutional principles.
- **Contextual Proficiency:** Designed for robust, context-aware dialogue and coherent responses.
- **Market Competitor:** Positioned as a reliable alternative to other large-scale language models.

https://paperswithcode.com/paper/the-claude-3-model-family-opus-sonnet-haiku

https://en.wikipedia.org/wiki/Claude_(language_model)

Word Embedding

جلسه دوم - کارگاه NLP

تعبیه (برداری) کلمات

(Jurafsky کتاب فصل ۳، ۶)

Recap:

- lower case
- stem / lemma } Computation Issue
- Tokenization → Input
- POS Tagging
- Hypernyme, Hyponyme } → meaning (Sense)

Taxonomy

color hyper

hyperbrown hypo

hypo dark brown

- word/Token Embedding
- word representation

feature → (across the columns); label ↓ (Price column); Tabular data → (row labels)

	x_1 Area (m^2)	x_2 # rooms	x_3 Zipcode	x_4 Parking lot	Price (K$)
H1	120	3	5	1	300
H2	100	2	12	0	210
H3	300	5	1	3	420
H4	200	4	3	2	?

S ↓ (down the rows)

0.7   100   0.1   2   0.05   12   0.2   0

Rule_based : $y = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$

Scale
محدودیت ها : ۱- تعداد سمپل کم
limitation
۲- تعداد فیچر کم

ML : train کردن w ها در یک فرآیند iterative

ML (Input): Tabular Data (Structral)

DL (Input): Raw Data (unstructed)

modvality: Text, Image, Video, Audio

Motivation:

	authority	event	rich	tail	beauty	gender
battle	0	1	0	0	0	0
horse	0.1	0	0.05	1	0.2	0
King	1	0	1	0	0.5	-1
man	0.2	0	0.2	0	0.2	-1
queen	1	0	1	0	1	1
woman	0.2	0	0.2	0	0.8	1

$\overrightarrow{King} - \overrightarrow{man} = (0.8, 0, 0.8, 0, 0.3, 0)$

$\vec{K} - \vec{m} + \overrightarrow{woman} = (1, 0, 1, 0, 1.1, 1)$

$\overrightarrow{queen} = (1, 0, 1, 0, 1, 1)$

1. BoW's

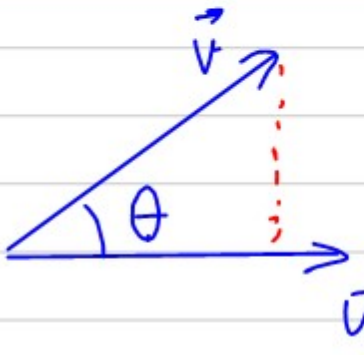
1 _ unordered

man bites cat

cat bites man

2 _ Search engine (Information Retrieval / IR)

Google | Query (dog is happy) |



$$\vec{u} \cdot \vec{v} = |\vec{u}||\vec{v}|\cos\theta$$

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}||\vec{v}|}$$

2. tf/idf: term frequency

inverse document frequency

$$tf = \frac{\#w_t}{\#w_D}, \quad IDF = \log \frac{N+1}{df}$$

N: تعداد کل داکیومنت‌ها

df: داکیومنت‌هایی که شامل آن توکن هستند.

emprical (تجربی)

---

3. One_hot encoding:

limitation: Sparsity → { Computation ↑ ; inefficient multiplication }

Advantage: non-Ordinal

کلمه‌ها در → Categorical f → numerical f

	color
i16	blue
i15Pro	red
i15s	gray

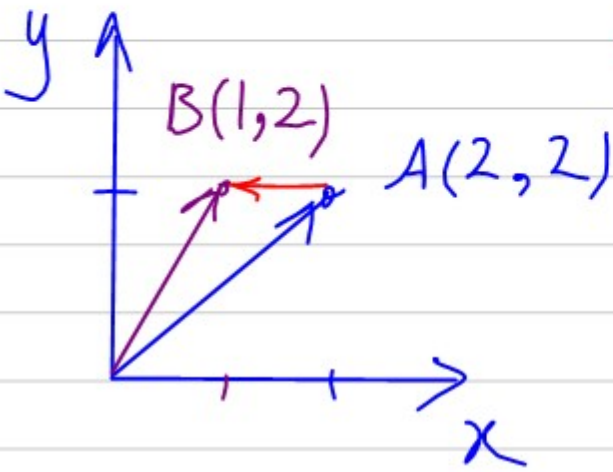
↑ categorical

label encoding

color
1
2
3

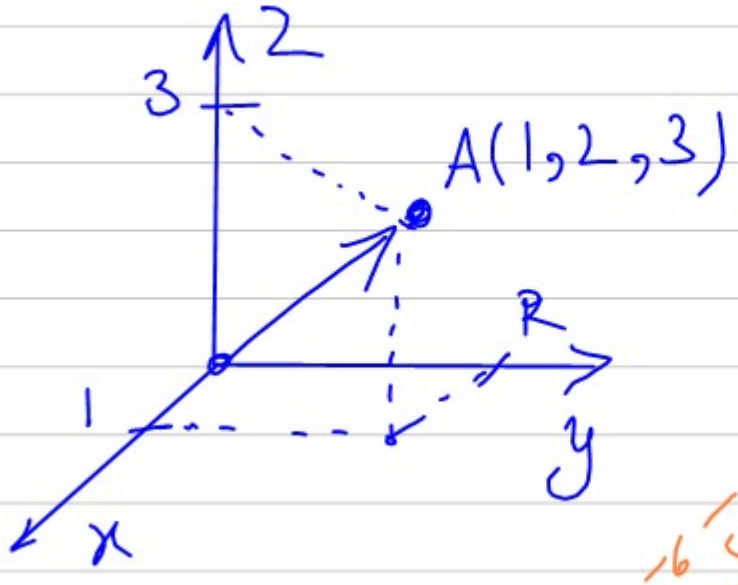
One_hot encoding

	color_b	color_r	color_g
1	1	0	0
2	0	1	0
3	0	0	1



$$\vec{AB} = \sqrt{(2-1)^2 + (2-2)^2} = 1$$

Euclidean (اقلیدسی)



انجام کار در زمان — انجام درست کار

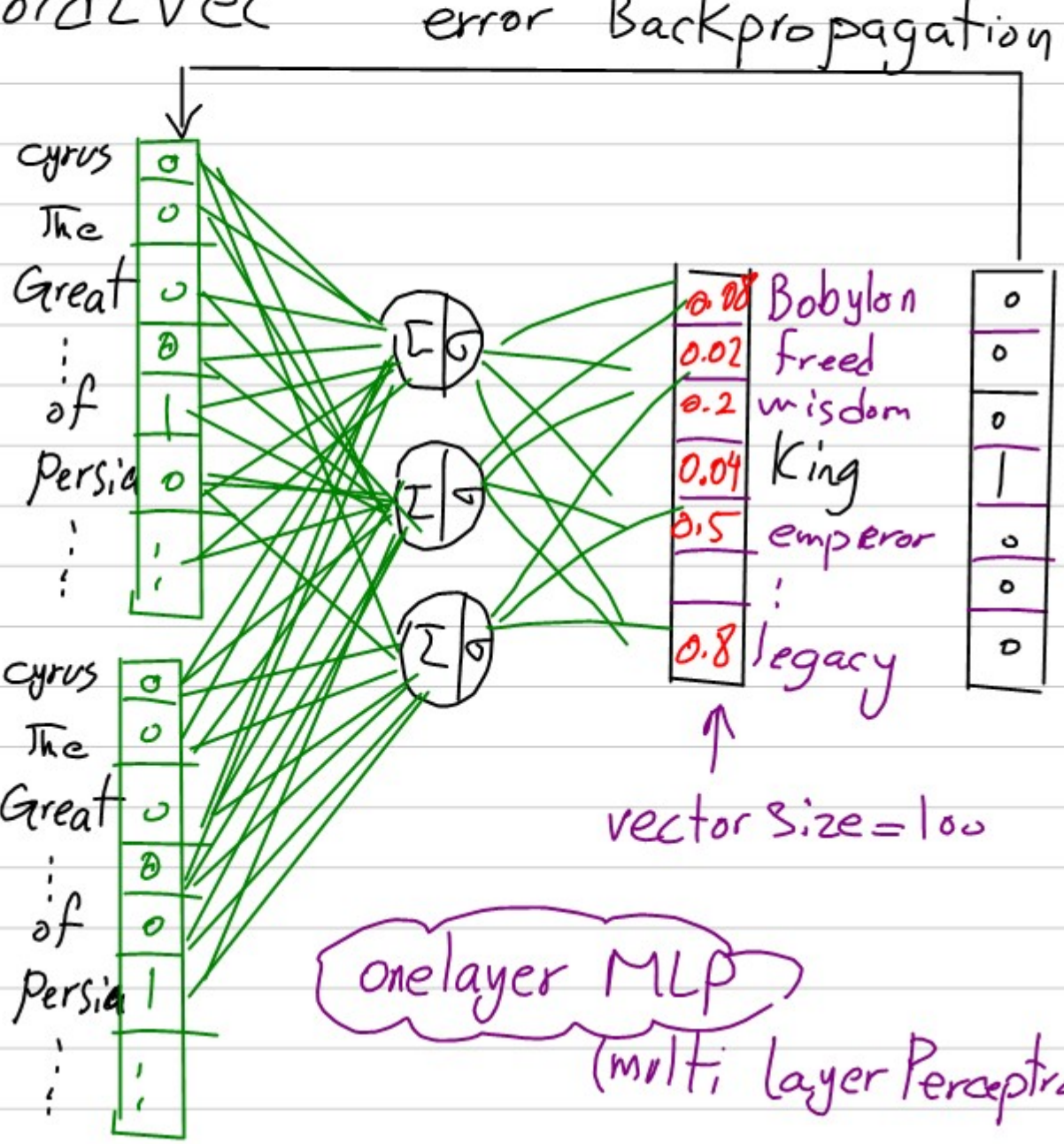
Productivity = Efficiency + Effectiveness

بهره‌وری — راندمان (کارایی) — اثربخشی

---

n_gram : I like NLP because ...

4. Word2Vec



Legendary ~~King~~ of persia

3_gram → # sliding window = 3

P(w | Context) CBow

P(C | word) Skip_gram

- word2Vec
- GloVe
- FastText

→ Context-free (meaning of word)

بردار → 2 4 6

ماتریس → (2, 4, 6)

	x_1	x_2	x_3
s_1	2	4	6
s_2	8	0	1
s_3	-1	2	1

تنسور →