GenAI: Best Practices

Release 1.0

Wenqiang Feng and Di Zhen

December 11, 2024

CONTENTS

Welcome to our **GenAI: Best Practices**!!! The PDF version can be downloaded from HERE.

CHAPTER

ONE

PREFACE

Chinese proverb

Good tools are prerequisite to the successful execution of a job. – old Chinese proverb

1.1 About

1.1.1 About this book

This is the book for our Generative AI: Best practics [AutoFeatures] API. The PDF version can be downloaded from HERE. **You may download and distribute it. Please beaware, however, that the note contains typos as well as inaccurate or incorrect description.**

The `API` assumes that the reader has a preliminary knowledge of `python` programing and `Linux`. And this document is generated automatically by using sphinx.

1.1.2 About the authors

- **Wenqiang Feng**
  - Sr. Data Scientist and PhD in Mathematics
  - University of Tennessee at Knoxville
  - Webpage: http://web.utk.edu/~wfeng1/
  - Email: von198@gmail.com

- **Biography**

  Wenqiang Feng is Data Scientist within DST's Applied Analytics Group. Dr. Feng's responsibilities include providing DST clients with access to cutting-edge skills and technologies, including Big Data analytic solutions, advanced analytic and data enhancement techniques and modeling.

  Dr. Feng has deep analytic expertise in data mining, analytic systems, machine learning algorithms, business intelligence, and applying Big Data tools to strategically solve industry problems in a cross-functional business. Before joining DST, Dr. Feng was an IMA Data Science Fellow at The Institute for Mathematics and its Applications (IMA) at the University of Minnesota. While there, he helped startup companies make marketing decisions based on deep predictive analytics.

Dr. Feng graduated from University of Tennessee, Knoxville, with Ph.D. in Computational Mathematics and Master's degree in Statistics. He also holds Master's degree in Computational Mathematics from Missouri University of Science and Technology (MST) and Master's degree in Applied Mathematics from the University of Science and Technology of China (USTC).

- **Declaration**

  The work of Wenqiang Feng was supported by the IMA, while working at IMA. However, any opinion, finding, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the IMA, UTK and DST.

1.2 Feedback and suggestions

Your comments and suggestions are highly appreciated. I am more than happy to receive corrections, suggestions or feedback through email (Wenqiang Feng: von198@gmail.com and Di Zhen: dizhen318@gmail.com) for improvements.

CHAPTER TWO

PRELIMINARY

In this chapter, we will introduce some math and NLP preliminaries which is highly used in Generative AI.

2.1 Math Preliminary

2.1.1 Vector

A vector is a mathematical representation of data that has both magnitude and direction. Each data point is represented as a feature vector, where each component of the vector corresponds to a specific feature or attribute of the data.

```python
import numpy as np
import gensim.downloader as api
# Download pre-trained GloVe model
glove_vectors = api.load("glove-twitter-25")

# Get word vectors (embeddings)
word1 = "king"
word2 = "queen"

# embedding
king = glove_vectors[word1]
queen = glove_vectors[word2]

print('king:\n', king)
print('queen:\n', queen)
```

```
king:
[-0.74501  -0.11992   0.37329   0.36847  -0.4472   -0.2288    0.70118
0.82872   0.39486  -0.58347   0.41488   0.37074  -3.6906   -0.20101
0.11472  -0.34661   0.36208   0.095679 -0.01765   0.68498  -0.049013
0.54049  -0.21005  -0.65397   0.64556 ]
queen:
[-1.1266   -0.52064   0.45565   0.21079  -0.05081  -0.65158   1.1395
0.69897  -0.20612  -0.71803  -0.02811   0.10977  -3.3089   -0.49299
```

(continues on next page)

(continued from previous page)

```
-0.51375   0.10363  -0.11764  -0.084972  0.02558   0.6859   -0.29196
0.4594   -0.39955  -0.40371   0.31828 ]
```

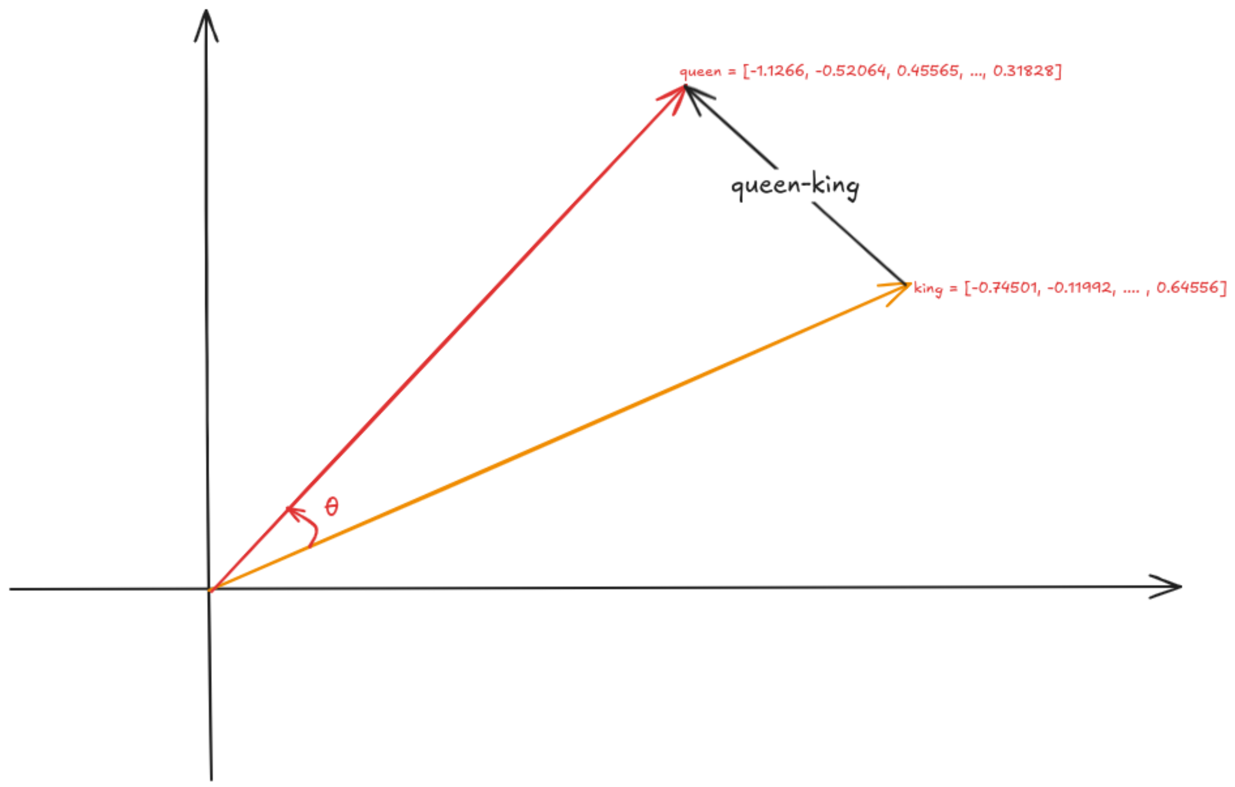
queen = [-1.1266, -0.52064, 0.45565, ..., 0.31828]
queen-king
king = [-0.74501, -0.11992, ... , 0.64556]
θ

Fig. 1: Vector

2.1.2 Norm

Norm is a function that maps a vector to a single positive value, representing its magnitude. Norms are used to calculate distances between vectors, which is vital for measuring prediction errors in models, performing feature selection, and applying regularization techniques.

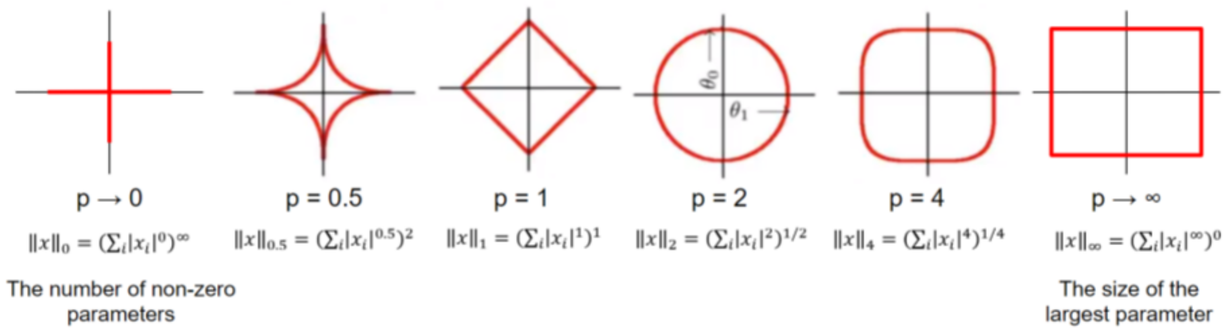
p → 0
$\|x\|_0 = (\sum_i |x_i|^0)^\infty$
The number of non-zero parameters
p = 0.5
$\|x\|_{0.5} = (\sum_i |x_i|^{0.5})^2$
p = 1
$\|x\|_1 = (\sum_i |x_i|^1)^1$
p = 2
$\|x\|_2 = (\sum_i |x_i|^2)^{1/2}$
θ_0
θ_1
p = 4
$\|x\|_4 = (\sum_i |x_i|^4)^{1/4}$
p → ∞
$\|x\|_\infty = (\sum_i |x_i|^\infty)^0$
The size of the largest parameter

Fig. 2: Geometrical Interpretation of Norm (source_1)

- Formula:

The ℓ^p norm for $\vec{v} = (v_1, v_2, \cdots, v_n)$ is

$$||\vec{v}||_p = \sqrt[p]{|v_1|^p + |v_2|^p + \cdots + |v_n|^p}$$

- ℓ^1 norm: Sum of absolute values of vector components, often used for feature selection due to its tendency to produce sparse solutions.

```
# l1 norm
np.linalg.norm(king,ord=1) #    max(sum(abs(x), axis=0))

### 13.188952
```

- ℓ^2 norm: Square root of the sum of squared vector components, the most common norm used in many machine learning algorithms.

```
# l2 norm
np.linalg.norm(king,ord=2)

### 4.3206835
```

- ℓ^∞ norm (Maximum norm): The largest absolute value of a vector component.

2.1.3 Distances

- Manhattan Distance (ℓ^1 Distance)

  Also known as taxicab or city block distance, Manhattan distance measures the absolute differences between the components of two vectors. It calculates the distance a point would travel along the grid lines in a Cartesian plane, as if navigating through a city.

  For two vector $\vec{u} = (u_1, u_2, \cdots, u_n)$ and $\vec{v} = (v_1, v_2, \cdots, v_n)$, the Manhattan Distance distance $d(\vec{u}, \vec{v})$ is

$$d(\vec{u}, \vec{v}) = ||\vec{u} - \vec{v}||_1 = |u_1 - v_1| + |u_2 - v_2| + \cdots + |u_n - v_n|$$

- Euclidean Distance (ℓ^2 Distance)

  Euclidean distance is the most common way to measure the distance between two points (vectors) in space. It is essentially the straight-line distance between them, calculated using the Pythagorean theorem.

  For two vector $\vec{u} = (u_1, u_2, \cdots, u_n)$ and $\vec{v} = (v_1, v_2, \cdots, v_n)$, the Euclidean Distance distance $d(\vec{u}, \vec{v})$ is

$$d(\vec{u}, \vec{v}) = ||\vec{u} - \vec{v}||_2 = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2}$$

- Minkowski Distance (ℓ^p Distance)

  Minkowski distance is a generalization of both Euclidean and Manhattan distances. It introduces a parameter p that allows you to adjust the sensitivity of the distance metric.

- Cos Similarity

  Cosine similarity measures the angle between two vectors rather than their straight-line distance. It is used to determine how similar two vectors are by focusing on their orientation rather than their magnitude. This makes it particularly useful for high-dimensional data, such as text, where the magnitude of the vectors may not be as important as the direction.

  The Cos similarity for two vector $\vec{u} = (u_1, u_2, \cdots, u_n)$ and $\vec{v} = (v_1, v_2, \cdots, v_n)$ is

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

  - 1 means the vectors point in exactly the same direction (perfect similarity).
  - 0 means they are orthogonal (no similarity).
  - -1 means they point in opposite directions (complete dissimilarity).

```
# Compute cosine similarity between the two word vectors
np.dot(king,queen)/(np.linalg.norm(king)*np.linalg.norm(queen))

### 0.92024213
```

```
# Compute cosine similarity between the two word vectors
similarity = glove_vectors.similarity(word1, word2)
print(f"Word vectors for '{word1}': {king}")
print(f"Word vectors for '{word2}': {queen}")
print(f"Cosine similarity between '{word1}' and '{word2}':
↪{similarity}")
```

```
Word vectors for 'king': [-0.74501  -0.11992   0.37329   0.36847  -
↪0.4472   -0.2288    0.70118
0.82872   0.39486  -0.58347   0.41488   0.37074  -3.6906   -0.20101
0.11472  -0.34661   0.36208   0.095679 -0.01765   0.68498  -0.049013
0.54049  -0.21005  -0.65397   0.64556 ]
Word vectors for 'queen': [-1.1266   -0.52064   0.45565   0.21079  -
↪0.05081  -0.65158   1.1395
0.69897  -0.20612  -0.71803  -0.02811   0.10977  -3.3089   -0.49299
-0.51375   0.10363  -0.11764  -0.084972  0.02558   0.6859   -0.29196
0.4594   -0.39955  -0.40371   0.31828 ]
Cosine similarity between 'king' and 'queen': 0.920242190361023
```

2.2 NLP Preliminary

2.2.1 Vocabulary

2.2.2 Tagging

2.2.3 Lemmatization

2.2.4 Tokenization

Out-of-vocabulary tokens Subword Tokenization

2.3 Platform and Packages

CHAPTER

THREE

WORD AND SENTENCE EMBEDDING

Word embedding is a method in natural language processing (NLP) to represent words as dense vectors of real numbers, capturing semantic relationships between them. Instead of treating words as discrete symbols (like one-hot encoding), word embeddings map words into a continuous vector space where similar words are located closer together.

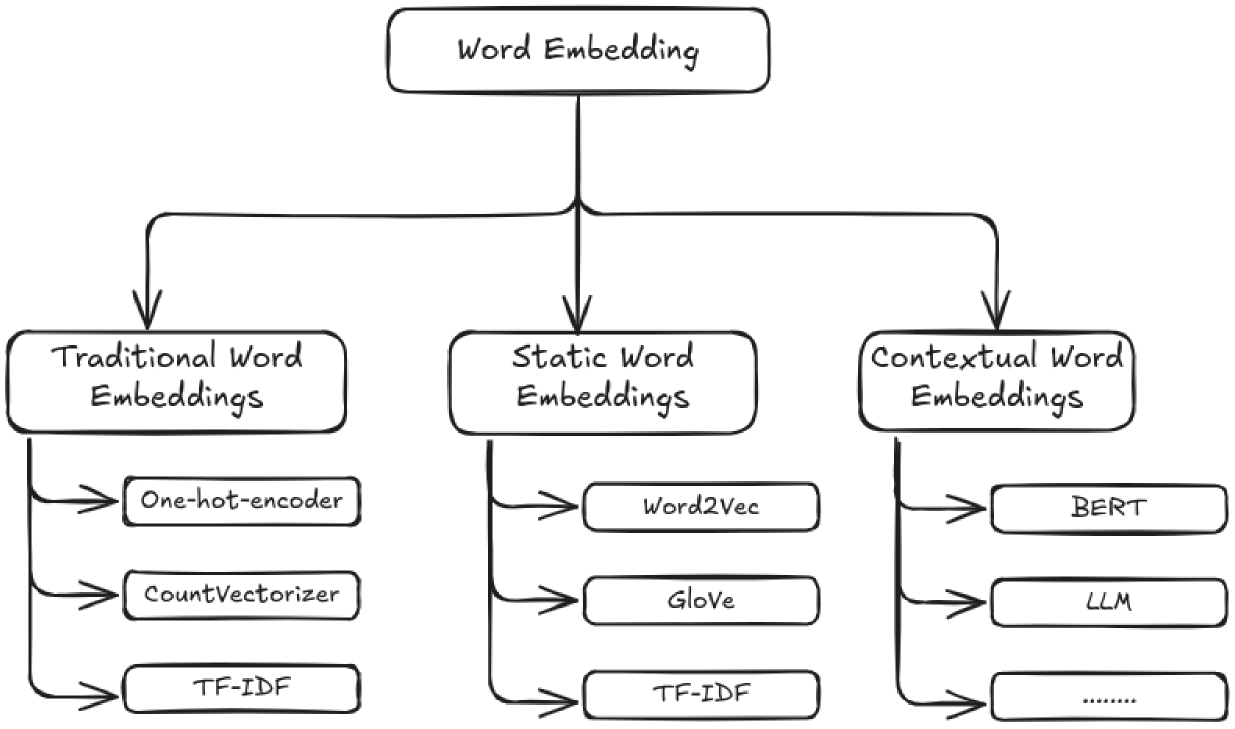

Fig. 1: Embedding Diagram

3.1 Bag-of-Word

Bag of Words (BoW) is a simple and widely used text representation technique in natural language processing (NLP). It represents a text (e.g., a document or a sentence) as a collection of words, ignoring grammar, order, and context but keeping their frequency.

Key Features of Bag of Words:

1. **Vocabulary Creation**: - A list of all unique words in the dataset (the “vocabulary”) is created. - Each word becomes a feature.
2. **Representation**: - Each document is represented as a vector or a frequency count of words from the vocabulary. - If a word from the vocabulary is present in the document, its count is included in the vector. - Words not present in the document are assigned a count of zero.
3. **Simplicity**: - The method is computationally efficient and straightforward. - However, it ignores the sequence and semantic meaning of the words.

Applications:

- Text Classification
- Sentiment Analysis
- Document Similarity

Limitations:

1. **Context Ignorance**: - BoW does not capture word order or semantics. - For example, “not good” and “good” might appear similar in BoW.
2. **Dimensionality**: - As the vocabulary size increases, the vector representation grows, leading to high-dimensional data.
3. **Sparse Representations**: - Many entries in the vectors might be zeros, leading to sparsity.

3.1.1 One Hot Encoder

3.1.2 CountVectorizer

To overcome these limitations, advanced techniques like **TF-IDF**, **word embeddings** (e.g., Word2Vec, GloVe), and contextual embeddings (e.g., BERT) are often used.

3.2 TF-IDF

3.3 Word2Vec

3.4 GloVE

3.5 Fast Text

3.6 BERT

CHAPTER

FOUR

PROMPT ENGINEERING

4.1 Background about LLM and Prompt

4.2 Prompt Engineering Basics

4.2.1 Prompt Components

4.2.2 Prompt Engineering Principles

4.3 Advanced Prompt Engineering

CHAPTER

FIVE

RETRIEVAL-AUGMENTED GENERATION

5.1 Overview

5.2 Indexing

5.3 Retrieval

5.4 Generation

CHAPTER

SIX

FINE TUNING

CHAPTER

SEVEN

PRE-TRAINING

CHAPTER

EIGHT

MAIN REFERENCE

BIBLIOGRAPHY

[AutoFeatures] Wenqiang Feng and Ming Chen. Python Data Audit Library API, 2019.