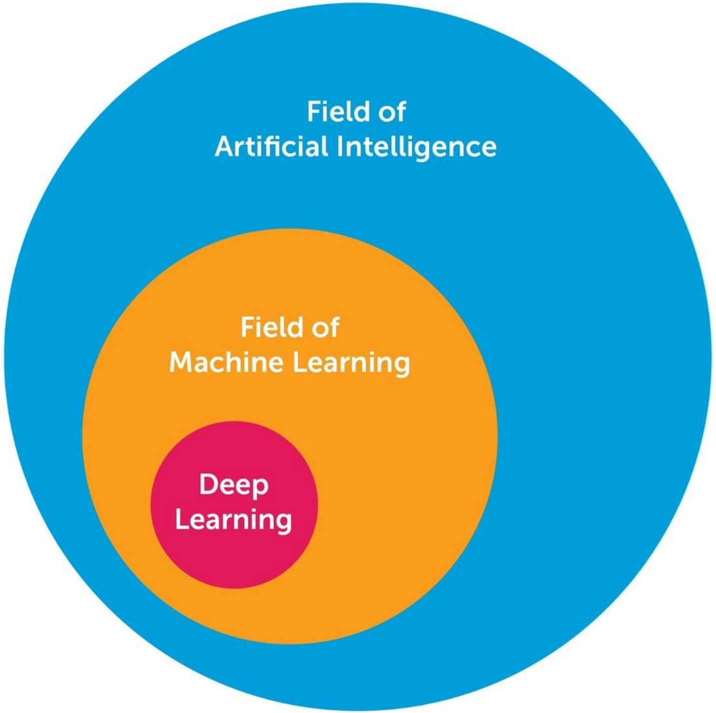
Natural Language Processing & Information Retrieval

Kwaku Ofosu-Tuffour

WILLIAM & MARY
CHARTERED 1693

Natural Language Processing: Background

Artificial Intelligence



- AI: Machines that perform jobs that mimic human behavior
- Machine Learning: Machines that get better at a task without explicit programming
- Deep Learning: Machines that have an artificial neural network to solve complex problem (inspired by the human brain

Natural Language Processing

Natural Language Processing (NLP) is a **machine learning technology** that gives computers the ability to interpret, manipulate, and comprehend human language.

https://aws.amazon.com/what-is/nlp/

Natural Language Processing: Important Tools

NumPy

NumPy (Numerical Python): Foundational package for scientific computation in Python

- Provides an n-dimensional array object, along with many important mathematical operations

Tensors

Tensors: generalizes the concept of vectors and matrices to higher dimensions

- Important framework for ML frameworks
- ability to run on GPUs for faster computation
- used to compute gradients

Pandas

Pandas: Python library that provides high-level data structures and other tools for data analysis. A common data structure is the dataframe

	Vtype	ivm				xyz		
	Coords	a	b	c	d	x	y	z
Shell	Ball							
layer0	0	0	0	0	0	0.000	0.000	0.000
layer1	1	0	1	1	2	0.000	-0.707	-0.707
	2	0	1	2	1	-0.707	0.000	-0.707
	3	0	2	1	1	-0.707	-0.707	0.000
	4	1	0	1	2	0.707	0.000	-0.707
	5	1	0	2	1	0.000	0.707	-0.707
	6	1	1	0	2	0.707	-0.707	0.000
	7	1	1	2	0	-0.707	0.707	0.000

Natural Language Processing: Important Terms

Tokenization

Tokenization: A preprocessing procedure where a text is split up into individual tokens

ex) “I study mathematics at W&M”

-> [“I”, “study”, “mathematics”, “at”, “W&M”]

Stop Words

Stop Words: Commonly used that are not taken into account during a preprocessing phase due to lack of semantic significance

ex) a, of, on, I, for, with, the, at, from, in, to

Stemming and Lemmatization

Stemming: The process of removing word endings

Lemmatization: The process of reducing a word to a base/root form

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Stemming and Lemmatization

ex) The boy’s cars are different colors

are, am, is ⇒ be

car, cars, car’s, cars’ ⇒ car

The boy car be differ color

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Normalization

The process of transforming a vector into a unit sphere. This process is necessary when data points within a vector are skewed.

There are various types of normalization:

- scaling to a range
- clipping
- log scaling
- z-score

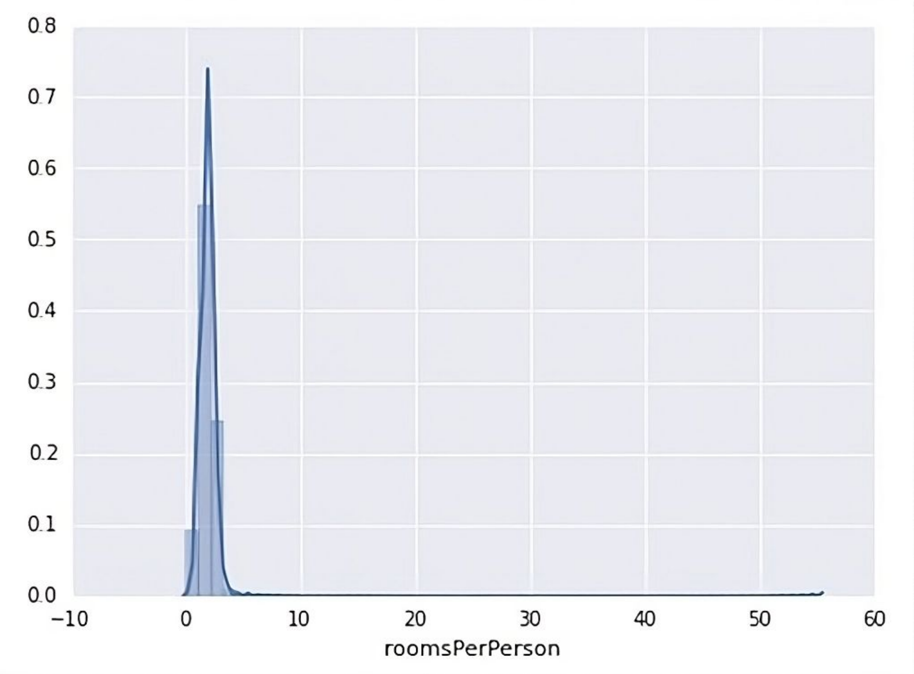
https://developers.google.com/machine-learning/data-prep/transform/normalization

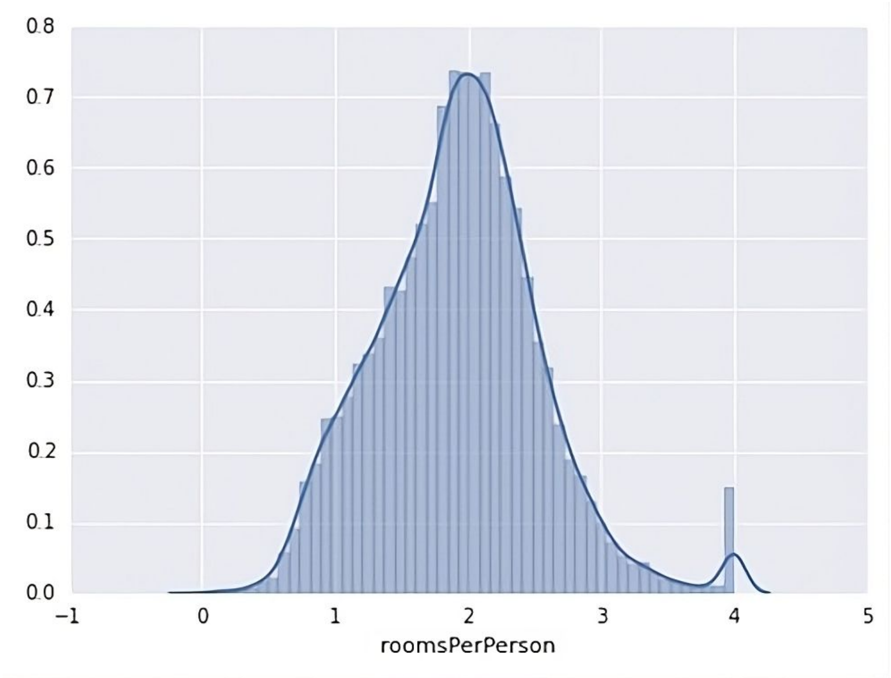
Normalization: Scaling

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

https://developers.google.com/machine-learning/data-prep/transform/normalization

Normalization: Clipping

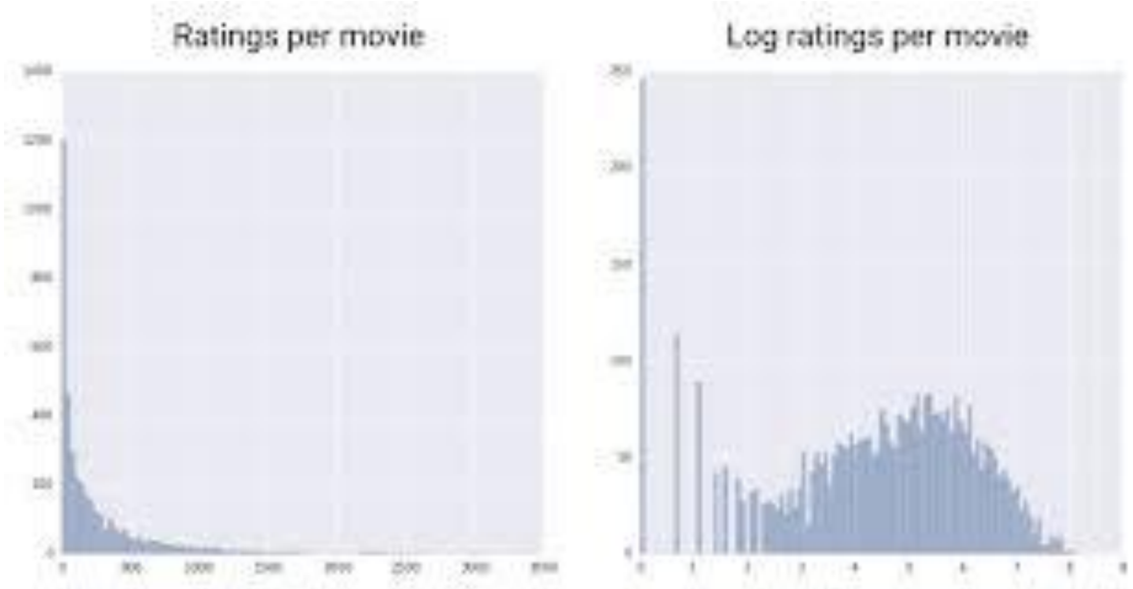




https://developers.google.com/machine-learning/data-prep/transform/normalization

Normalization: Log Scaling

$$x' = \log(x)$$



https://developers.google.com/machine-learning/data-prep/transform/normalization

Normalization: Z-Score

$$x' = \frac{x - \mu}{\sigma}$$

μ = mean / average

σ = standard deviation

https://developers.google.com/machine-learning/data-prep/transform/normalization

Normalization: Use Cases

Linear Scaling: When a feature is uniformly distributed across a certain range

Clipping: When a feature contains extreme outliers

Log Scaling: When a feature adheres to a logarithmic pattern

Z-Score: When a feature does NOT contain extreme outliers

https://developers.google.com/machine-learning/data-prep/transform/normalization

Vectorizer

A vectorizer is a tool that converts textual data into numerical format. This concept is crucial for many machine learning concepts, as many algorithms typically deal with numerical input.

This process can involve multiple steps, including:

- tokenization
- counting word frequencies
- normalization

Vector Semantics

Important Terms

- Word Similarity: a measure of semantic similarity between a given set of words
- **Embedding**: Defining the meaning of a word as a vector

Embeddings

The method of defining a word as a vector.

- This process is a standard way of representing meaning in NLP
- Called an "embedding" because it's embedded into a space

Words (vectors) in the vector space that are similar in a vector context tend to have similar semantic meaning

https://web.stanford.edu/~jurafsky/slp3/

Embeddings

Which of these words are similar?

a) [18, 27, 41]
b) [-4, -1, 30]
c) [16, 25, 45]

Word2Vec

An approach for representing a word in a vector space

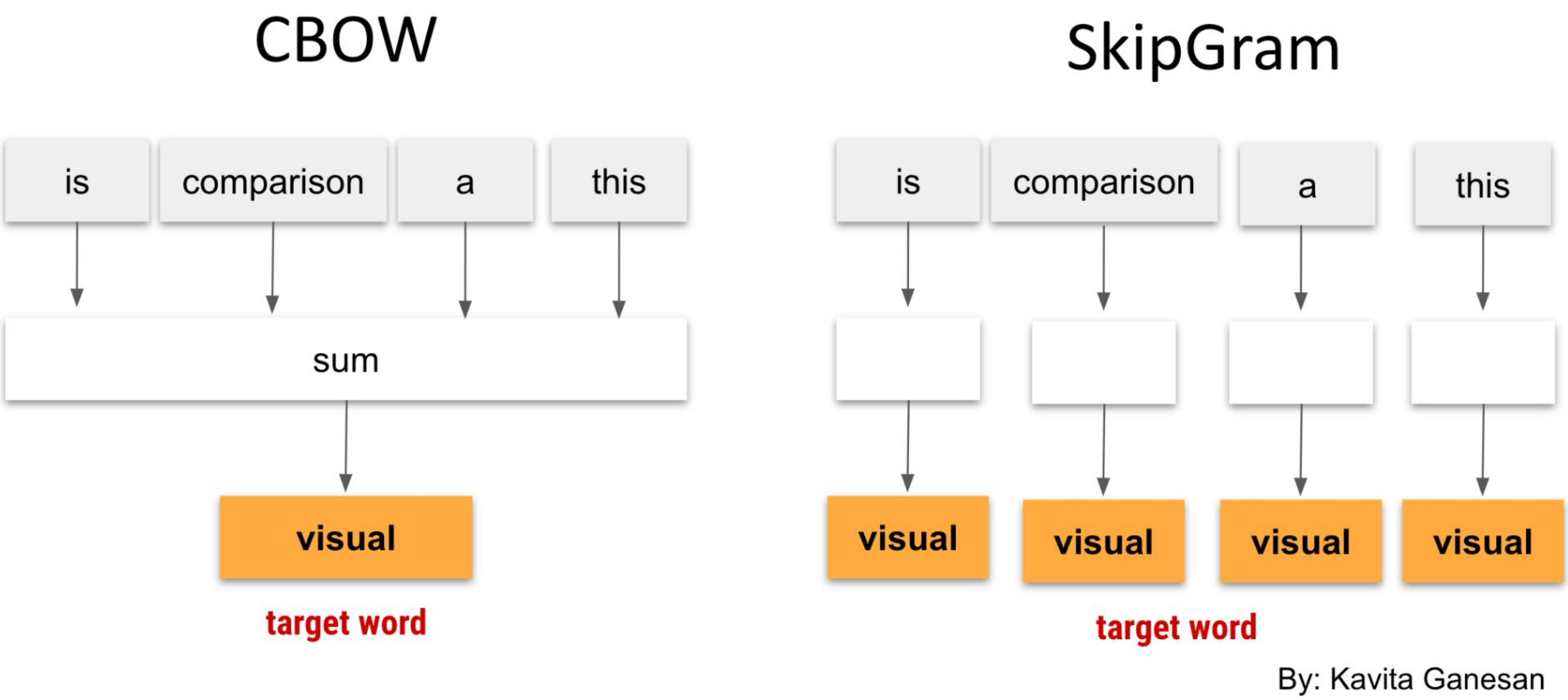
“not only will similar words tend to be close to each other, but that words can have multiple degrees of similarity” (Mikolov et al., 2013)

Word2Vec

Word2Vec can use two approaches:

- Continuous Bag-of-Words Model (CBOW)
  a. predicts current token based on the context
- Continuous Skip-gram Model
  a. predicts context from current token

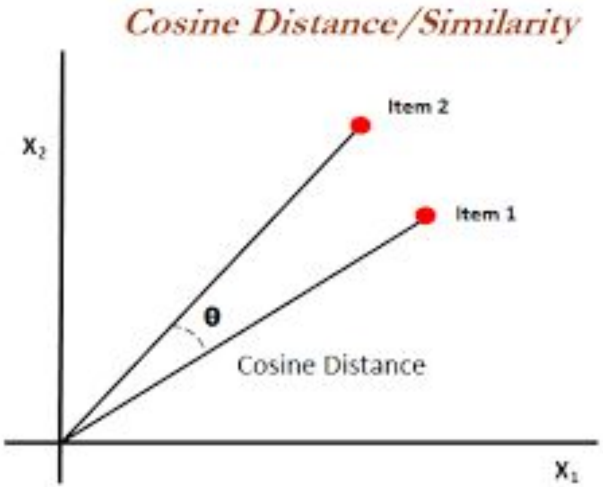
Drawback: Frequency of words are disregarded



This is a visual comparison

Cosine Similarity

A method of computing the similarity between two words (as vectors)



Dot Product

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i = |x||y| \cdot \cos(\theta)$$

Cosine Similarity

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

Cosine Similarity

$\mathbf{cos}(\mathbf{x}, \mathbf{y}) \rightarrow 1$: similar

$\mathbf{cos}(\mathbf{x}, \mathbf{y}) \rightarrow 0$: orthogonal (not-related)

Cosine Similarity

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

cos(cherry, information) = 0.017

cos(digital, information) = 0.996

Information Retrieval

Information Retrieval

Process of scoring various documents in terms of relevancy to a user's query.

Commonly used to create **search engines.**

https://web.stanford.edu/~jurafsky/slp3/14.pdf

Information Retrieval

Important Terms:

- **document**: any unit of text the system indexes and retrieves
- **collection**: a set of documents
- **term**: a word in a collection
- **query**: the user’s information need expressed as a set of terms

https://web.stanford.edu/~jurafsky/slp3/14.pdf

TF-IDF (Term Frequency-Inverse Document Frequency)

A statistical measure used to evaluate the importance of a word in a document, which is part of a corpus (a collection of documents).

- commonly used in information retrieval and text mining

Term Frequency

Measures how frequent a word appears in a document. With this method, all terms are given the same weight/importance

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

Inverse Document Frequency (IDF)

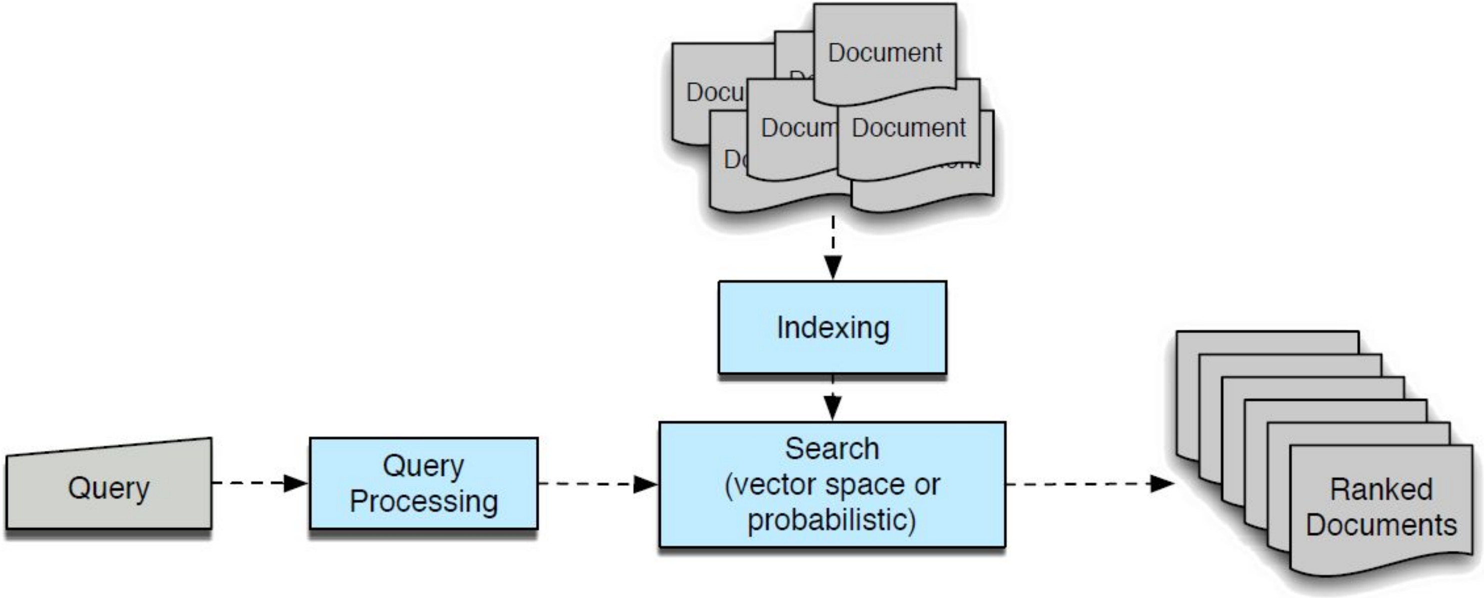
Measures **how important** a term is within a set of documents. Terms that are common across multiple documents are given a lower IDF score.

IDF(t) = log(Total number of documents / Number of documents with term t)

TF-IDF

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

Information Retrieval: ad hoc



Information Retrieval: ad hoc

- uses a vector-space model to map queries & documents to vectors
- uses cosine similarity between the vectors to rank the documents based on the query
- can use Word2Vec Bag-Of-Words

Information Retrieval

Suppose we have a query q and a set of documents D = {d1, d2, d3}.

After vector embedding, we can find the cosine similarity between the query and each document:

$$score(\boldsymbol{q},\ \boldsymbol{d}_{\mathrm{i}}) = \cos ine(\boldsymbol{q},\ \boldsymbol{d}_{\mathrm{i}}) = \frac{\boldsymbol{q} \cdot \boldsymbol{d}_{\mathrm{i}}}{|q||d_i|}$$

Information Retrieval

$$score(\boldsymbol{q},\ \boldsymbol{d}_{\mathrm{i}}) = \text{cosine}(\boldsymbol{q},\ \boldsymbol{d}_{\mathrm{i}}) = \frac{\boldsymbol{q} \cdot \boldsymbol{d}_{\mathrm{i}}}{|q||d_i|}$$

Information Retrieval

ex)

Query: sweet love

Document 1: Sweet sweet nurse! Love?

Document 2: Sweet sorrow

Document 3: How sweet is love?

Document 4: Nurse!

Information Retrieval

ex)

Preprocessing:

- tokenize query and documents
- create a set of all tokens
- perform stemming and lemmatization

['sweet', 'nurse', 'love', 'how', 'sorrow', 'is]

Information Retrieval

ex)

Sort Documents:

- Embed query/documents into vectors (TF-IDF)
- Find cosine similarity between query and documents; order in descending order

Query						
word	**cnt**	**tf**	**df**	**idf**	**tf-idf**	**n'lized** = tf-idf/$\lvert q \rvert$
sweet	1	1	3	0.125	0.125	0.383
nurse	0	0	2	0.301	0	0
love	1	1	2	0.301	0.301	0.924
how	0	0	1	0.602	0	0
sorrow	0	0	1	0.602	0	0
is	0	0	1	0.602	0	0
$\lvert q \rvert = \sqrt{.125^2 + .301^2} = .326$						

	Document 1					Document 2				
word	**cnt**	**tf**	**tf-idf**	**n'lized**	**× q**	**cnt**	**tf**	**tf-idf**	**n'lized**	**×q**
sweet	2	1.301	0.163	0.357	**0.137**	1	1.000	0.125	0.203	**0.0779**
nurse	1	1.000	0.301	0.661	0	0	0	0	0	0
love	1	1.000	0.301	0.661	**0.610**	0	0	0	0	**0**
how	0	0	0	0	0	0	0	0	0	0
sorrow	0	0	0	0	0	1	1.000	0.602	0.979	0
is	0	0	0	0	0	0	0	0	0	0
$\lvert d_1 \rvert = \sqrt{.163^2 + .301^2 + .301^2} = .456$						$\lvert d_2 \rvert = \sqrt{.125^2 + .602^2} = .615$				
		Cosine: $\sum$ of column:			**0.747**		Cosine: $\sum$ of column:			**0.0779**

Evaluation of Information Retrieval

We can measure the performance of the ad hoc algorithm using two metrics: precision and recall

- *TP* + *FP* represents all documents returned from a query
- *TP* represents all documents that are truly relevant to the query
- *TP+FN* represents all documents in the collection that are relevant to the request

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Information Retrieval and Dense Vectors

Normal embeddings are ***static*** and do not take semantic meaning of tokens into account.

ex)

“I went to the bank to deposit some money”

“The player made a bank shot in the last quarter”

“They met me at the river bank”

Information Retrieval and Dense Vectors

Solution: Use Contextualized Embeddings

- ELMo
- ULMFiT
- OpenAI ChatGPT
- BERT: Bidirectional Encoder Representations from Transformers

Project Implementation

Market app for W&M

The goal of the project is to create a mobile application where W&M students can sell and buy various products from other students. Similar to Facebook marketplace, except catered and limited to W&M students

HuggingFace

A machine learning (ML) and data science platform and community that helps users build, deploy and train machine learning models.

Offers many trained, open-source ML models

HUGGING FACE

Tools/Frameworks Used

- Front-end: React Native (TypeScript/JavaScript)
- Back-end: Django (Python)
  - ML model: Sentence Transformers via *HuggingFace*
- Deployment: Amazon Web Services (AWS), App Store (internal-testing), Google Play Store (internal testing)

Search Algorithm

```python
def get_embeddings(documents):
    """
    Transforms the each document in the list into an embedded vector
    """
    response = requests.post(api_url, headers=headers, json={"inputs": documents, "options":{"wait_for_model":True}})
    return response.json()

def rank_similarity(input):
    """
    Ranks the similarity between the input vector and the document vectors using cosine similarity.
    In other words, it ranks the posts based on the user's search input
    """
    posts = Post.objects.all()
    documents = [post.product for post in posts]
    documents.insert(0, input)
    documents_ids = [post.id for post in posts]

    embeddings = get_embeddings(documents)
    query = embeddings.pop(0)
    documents.pop(0)
    return similarity(query, embeddings, documents, documents_ids)
```

Search Algorithm

```python
def similarity(query, embeddings, documents, documents_ids):
    """
    Function that calculates the cosine similarity between the query and document vectors.
    Return a list of posts in descending order by their search ranking
    """
    data = []
    for vector in embeddings:
        dot_product = np.dot(query, vector)
        query_magnitude = np.linalg.norm(query)
        vector_magnitude = np.linalg.norm(vector)
        data.append((dot_product) / (query_magnitude * vector_magnitude))
    df = pd.DataFrame({
        "Post_id": documents_ids,
        "Name": documents,
        "Score": data
    })
    df = df.sort_values(by=['Score'], axis=0, ascending=False)
    return list(df['Post_id'])
```

Results

Query: “Bottle”

Precision:

$(5)/(5+1) = 0.833$

Recall:

$(5)/(5+0) = 1$

Post_id	Name	Score
23	Shaker Bottle	0.72448
35	Black water bottle	0.717623
45	Purple water bottle	0.63708
34	W&M water bottle	0.604581
1	Cups	0.457705
48	V8 Juice	0.413679
40	Helmet	0.380746
6	Bird	0.37847
15	Spicy Sauce	0.371038
37	TV	0.369128

Results

Query: “Sports”

Precision:

(5)/(5+0) = 1.0

Recall:

(5)/(5+0) = 1.0

Post_id	Name	Score
37	TV	0.509026
12	Tennis Balls	0.465537
40	Helmet	0.456399
7	Purple Soccer Ball	0.447629
8	Soccer sweatshirt	0.414176
1	Cups	0.386867
6	Bird	0.355266
5	Dog	0.306227
38	Road Bike	0.277349
33	Boots	0.276298

Results

Query: “Pencils”

Precision:

(1)/(1+0) = 1.0

Recall:

(1)/(1+1) = 0.5

Post_id	Name	Score
18	BIC Mechanical Pencils	0.687582
1	Cups	0.365652
22	Five Star Paper	0.357549
36	Printer	0.347954
19	Mini Stapler	0.345633
25	Coat	0.324235
12	Tennis Balls	0.302151
46	Coffee creamer	0.286429
37	TV	0.272269
33	Boots	0.271016

Results

Query: “Instrument”

Precision:

(2)/(2+0) = 1.0

Recall:

(2)/(2+0) = 1.0

Post_id	Name	Score
13	Oboe	0.688479
50	Oboe Reed	0.646337
24	Handheld vacuum	0.337073
12	Tennis Balls	0.320299
37	TV	0.315224
40	Helmet	0.30879
19	Mini Stapler	0.308068
18	BIC Mechanical Pencils	0.305089
6	Bird	0.281814
5	Dog	0.263972

Results

Query: “Cooking”

Precision:

(3)/(3+1) = 0.75

Recall:

(3)/(3+5) = 0.375

Post_id	Name	Score
39	Pressure Cooker	0.49696
15	Spicy Sauce	0.42063
1	Cups	0.415951
37	TV	0.412203
6	Bird	0.362832
47	Frozen vegetables	0.346326
5	Dog	0.323628
36	Printer	0.322787
25	Coat	0.316749
49	Blanket	0.312194

Results

Query: “Technology”

Precision:

(2)/(2+0) = 1

Recall:

(2)/(2+2) = 0.5

Post_id	Name	Score
37	TV	0.440683
36	Printer	0.409767
5	Dog	0.312691
1	Cups	0.286745
40	Helmet	0.243689
22	Five Star Paper	0.24185
6	Bird	0.237662
46	Coffee creamer	0.234185
44	Stationeries	0.220467
10	Twix candy	0.22043

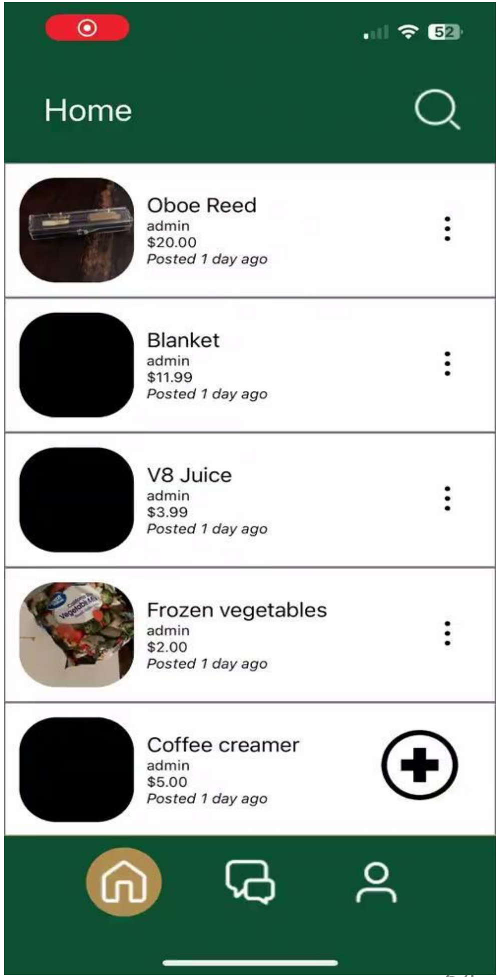
Results

Query: “School”

Post_id	Name	Score
37	TV	0.481606
5	Dog	0.434241
6	Bird	0.353587
1	Cups	0.352532
40	Helmet	0.334864
36	Printer	0.3288
25	Coat	0.313415
46	Coffee creamer	0.283103
33	Boots	0.27802
38	Road Bike	0.270293

Next Steps

- Refine current search engine algorithm
- Do more research in contextualized embeddings
- Develop our own contextualized embedding ML model



Bibliography

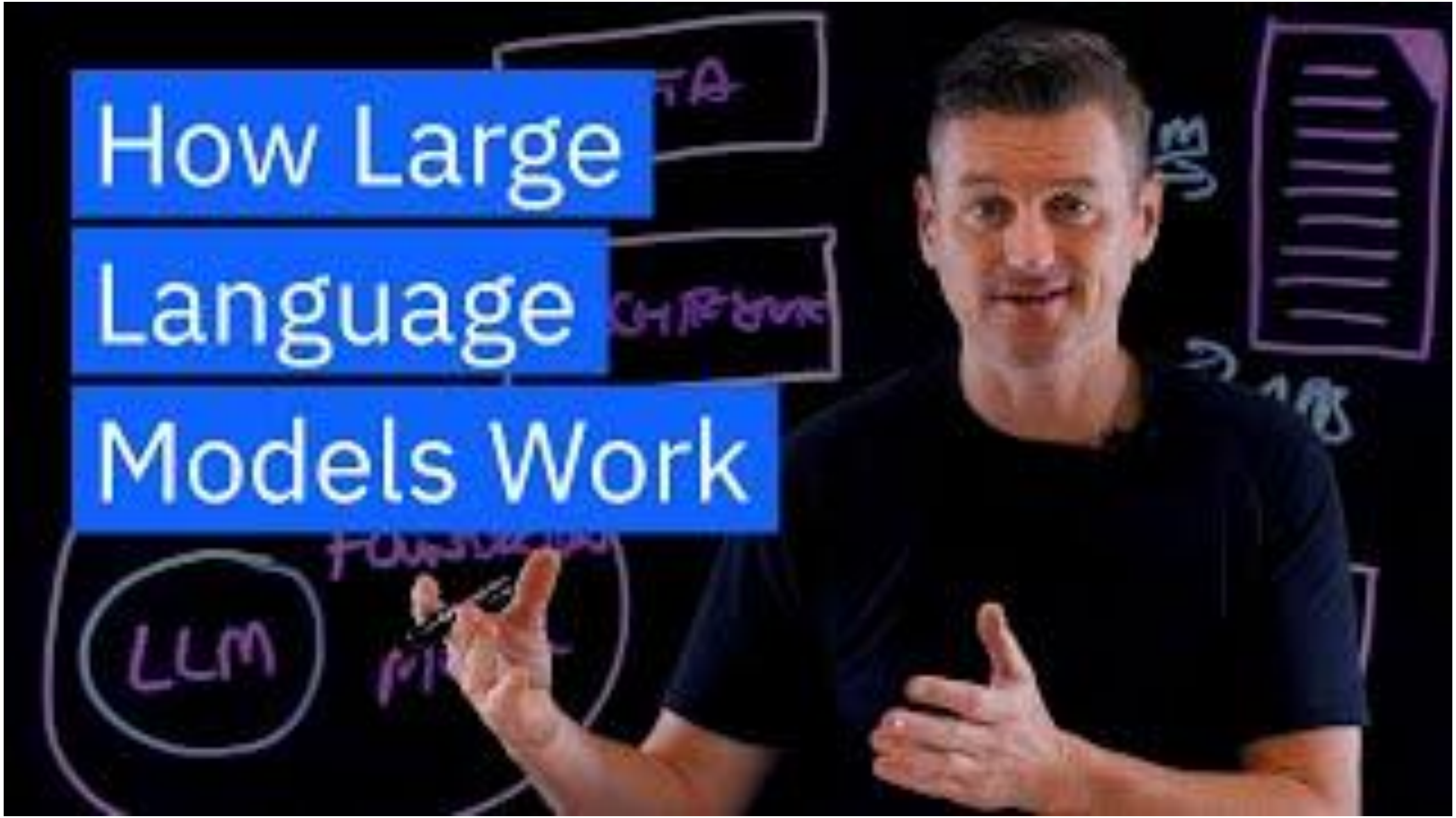
Jurafsky, Dan, and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson, 2022.

Manning, Christopher D., et al. *An Introduction to Information Retrieval*. Cambridge University Press, 2022.

“Normalization.” *Google*, Google, developers.google.com/machine-learning/data-prep/transform/normalization. Accessed 19 Mar. 2024.

“What Is NLP? - Natural Language Processing Explained - AWS.” *Amazon Web Services*, aws.amazon.com/what-is/nlp/. Accessed 20 Mar. 2024.

James M. Tucker, DATA 340 - Natural Language Processing (https://github.com/JamesMTucker/DATA_340_NLP/tree/master)


How Large
Language
Models Work
LLM