NLP Foundations

FOR PRESENTER USE ONLY

Overview

NLP foundations introduces various ways to represent text numerically including word vectors and text embeddings.

Learning objectives

- Define common use cases for text data
- Describe the evolution of text representation from traditional methods to embeddings
- Apply CountVectorizer, TF-IDF to complete a sentiment analysis of customer reviews
- Compare two embedding

Skills: CountVectorizer, TF-IDF, Word2Vec, BERT

FOR PRESENTER USE ONLY

Teaching guide

Use the following quick tips for instructional guidance. Refer to the XXX for full guidance on content and delivery.

Title	What to Do
Our learning goals	**Learning objectives slide** Capture a screenshot of this slide and drop in the class discussion forum or Slack channel.
Group and partner exercises	**Creating breakout rooms — group and partner exercises** • Set up breakout rooms for student pairs/groups. Screenshot the slide and post it in the discussion forum or Slack channel before opening the rooms. • At the 60-second mark, send a message to all rooms: “Start to wrap up your discussions. Rooms will close in one minute!” • Move around from breakout room to breakout room to “walk around the classroom.”

FOR PRESENTER USE ONLY

Teaching guide (Cont.)

Title	What to Do
Discussions	**Whole-class discussions** Screenshot the slide and put in the discussion forum or Slack channel. Some students may not have two monitors. You should stop sharing your screen so they can see each other's faces during the discussion.
Solo activities	**Solo activities** • Screenshot the slide and drop it in the discussion forum or Slack channel. • For solo activities, you can choose to put people in group breakout rooms (to create the "table" group that they would be in in a physical setting) or allow students to work independently in the main session.
Homework	Share homework guidelines in the discussion forum or Slack channel.

FOR PRESENTER USE ONLY

Suggested Agenda

Time	Activity
0:00–0:15	**Intros, Use Cases for Text**
0:15-1:10	**Representing Text with Numbers through embeddings.ipynb activity**
1:10-1:20	**Break**
1:20-1.55	**Representing Text with Numbers (cont'd) to embeddings**
1:55- 2:20	**Impact of method selection, Wrap Up, Exit Ticket Completion**

Change log

FOR PRESENTER USE ONLY

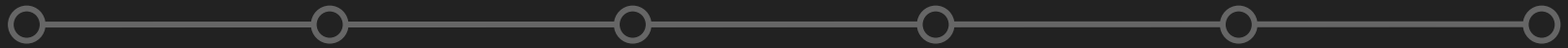
Customization opportunities

NLP and LLMs

The Foundations of NLP

GENERAL ASSEMBLY

Lesson roadmap



Welcome + warm-up

Use Cases for Text

Representing text with numbers

Text Embeddings

Impact of method selection

Bring It Home

Learning objectives

1. **Define** common use cases for text data
2. **Describe** the evolution of text representation from traditional methods to embeddings
3. **Apply** CountVectorizer, TF-IDF to complete a sentiment analysis of customer reviews
4. **Compare** two embedding methods, Word2Vec and BERT, using customer reviews
5. **Discuss** real-word considerations for embedding selection

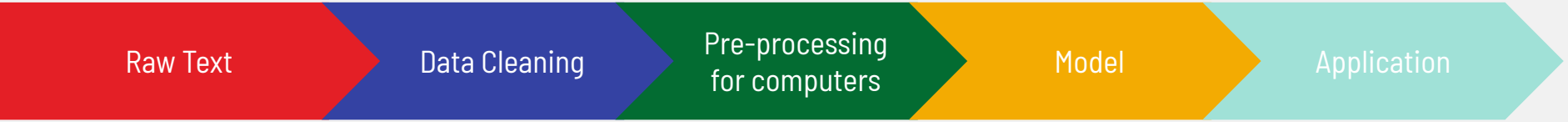
Warm Up

How do you use text?

How would you like to work with text data?

- ☐ Think about the types of data you work with that contain text
- ☐ Describe the types of tasks you would like to use text data to accomplish
- ☐ Brainstorm potential uses if you had a powerful AI system helping you parse through the data

NLP General Workflow



Use cases for text

Using text in the real-world

How do computers view text?

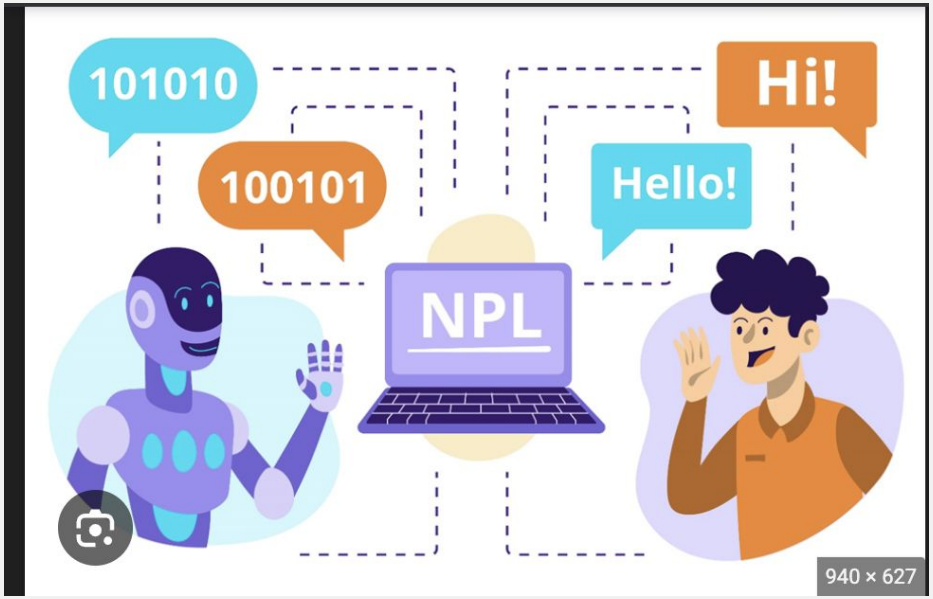
Natural Language Processing (NLP) is a specialized field of computer science and artificial intelligence primarily concerned with designing and building applications for facilitating interaction between machines and the natural language of humans – Definition Paraphrased from Sarkar

Let’s see a few examples of how this works.

When do we work with text data?

Natural Language Processing (NLP) is the field of computer science and artificial intelligence that uses principles of statistics and probability to translate human language into a form that computers can use and apply it to solve problems.

Let's see a few examples of when this is used.

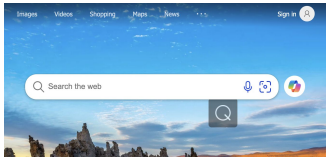


tktk **Create GA version of this image.** Source: https://blog.mitsde.com/what-is-nlp-how-it-aids-in-customer-satisfaction/

Common Use Cases

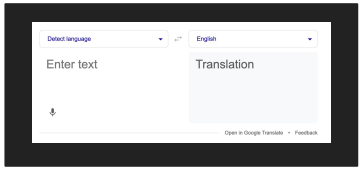
1

Searching the web



2

Translation



3

Summarization of documents

4

Identifying entities

Representing Text with Numbers

Translating text for computers

Computers see the world numerically

Computers understand the world in **numbers**. In order to use text and image data we have to first represent it in a way that computers can interpret.

In text, we do this with word or text embeddings, which are numerical representations of words or groups of words.

Text for humans

I'm learning NLP

Numbers for computers

[0, 1, 0, 0, 1, 0]
[1, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 1]

Methods for converting text to numbers

There are numerous methods for representing text with numbers.

They range from counts of every word used to complex embeddings that take into account the context of the word.

Let’s explore a few options

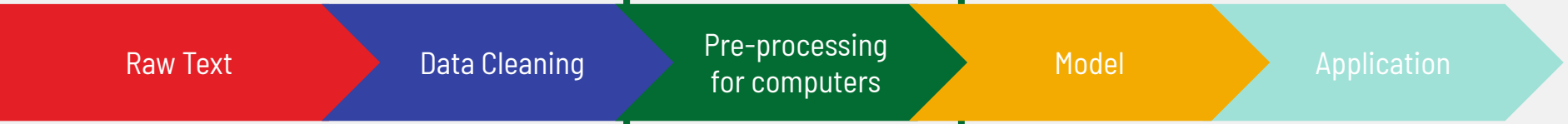
Text for humans

I’m learning NLP

Numbers for computers

[0, 1, 0, 0, 1, 0]
[1, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 1]

NLP General Workflow



We start with a collection of raw text this group of documents is your **corpus**

A **document** is a piece of text that represents one unit of analysis from the corpus

These steps will vary depending on your corpus common actions include: - **removing duplicates** to speed up models and reduce cost. -**removing identifying information** such as names for privacy - removing very common words (a, the, etc.) known as **stop words,** - simplifying text using methods such as **stemming/ lemmatization**

Tokens or **N-grams** are the unit of analysis. Often these are **words or pieces of words** or even characters. You must **use the same tokenization method your model used for training** (BERT Tokenizer → BERT model) These units are then **converted into numbers** the computer can use

Typically, you will use a **pre-trained model** to that you apply to your text. These models are built vast amounts of training data (think all of facebook,all of Github, all books, all of reddit, all scientific journal articles, the full web) You can also **build a model from scratch** but this **is very, very rare**

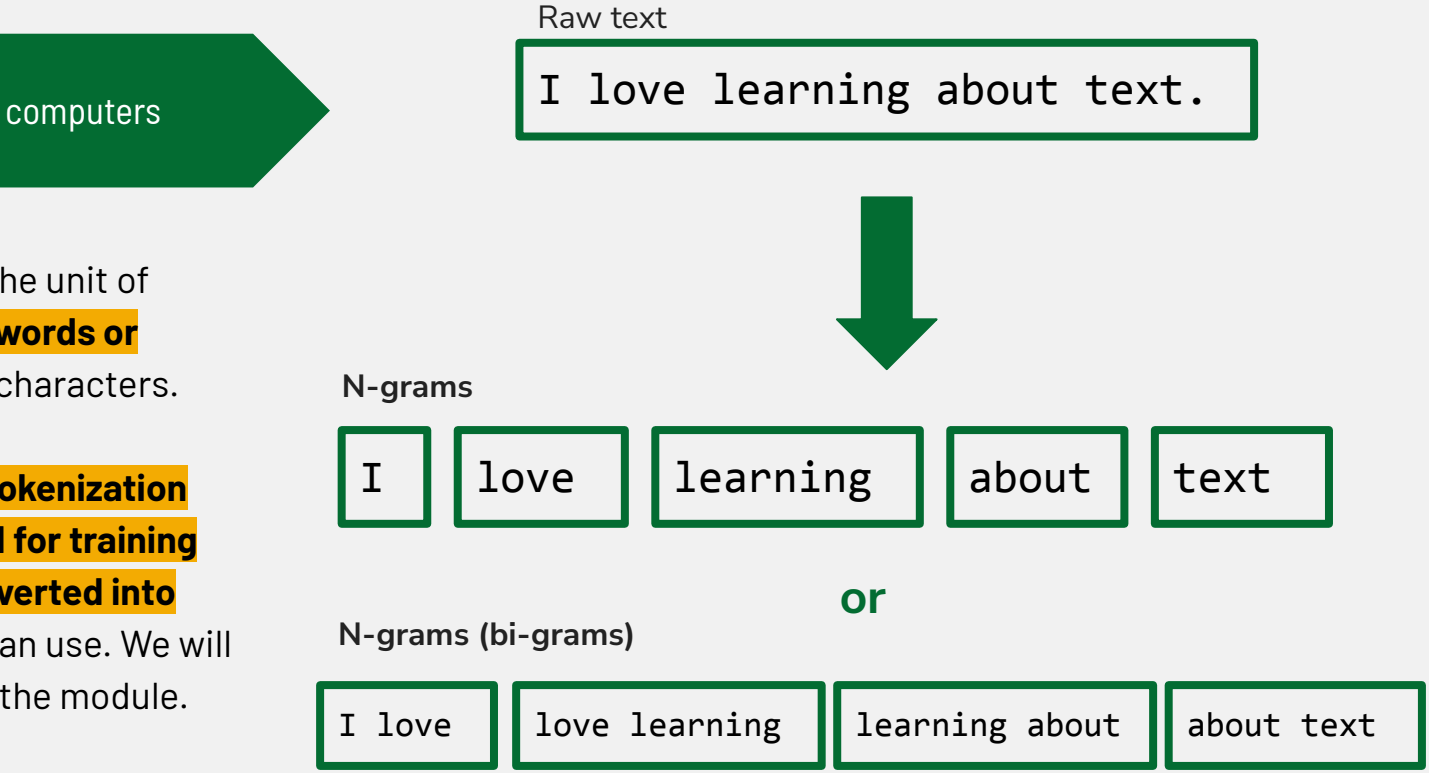
This is where you use or text for some purpose such as **summarization, translation, classification**

NLP General Workflow

Pre-processing for computers

Tokens or **N-grams** are the unit of analysis.Often these are **words or pieces of words** or even characters.

You must **use the same tokenization method your model used for training** These units are then **converted into numbers** the computer can use. We will see this in action later in the module.



POINTS

What do we call a the collection of all our documents for analysis?

A N-gram

B Pre-trained model

C Corpus

D Tokens

Which of the following could be N-grams or tokens for the word “learning” from the phrase “Learning is fun”?

Raw text: `Learning is fun`

A learning

B learn

C learning is

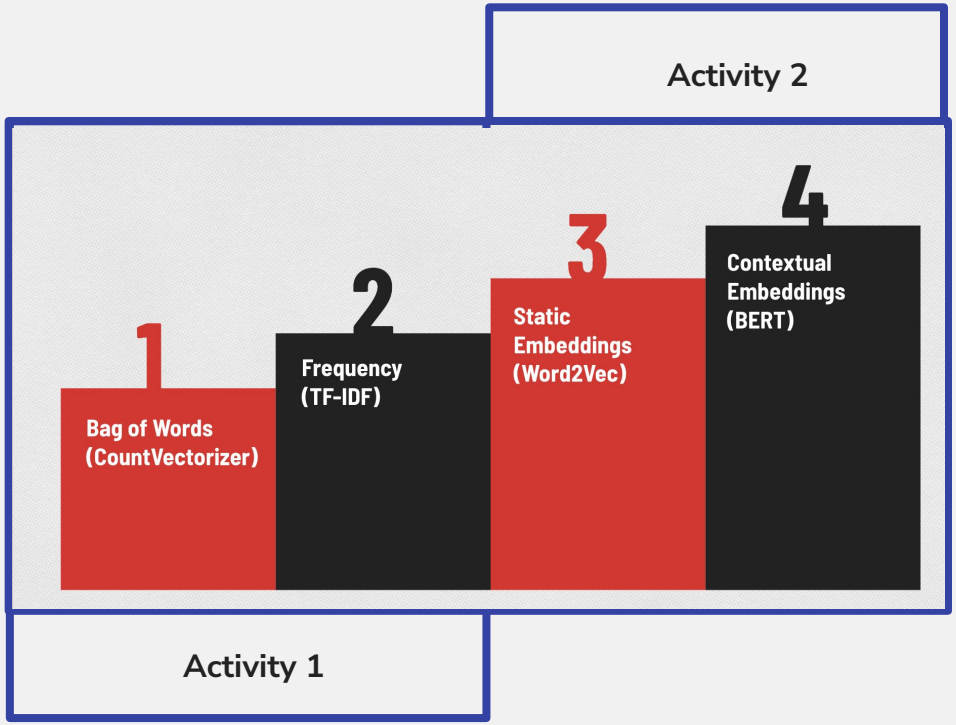
D All the above

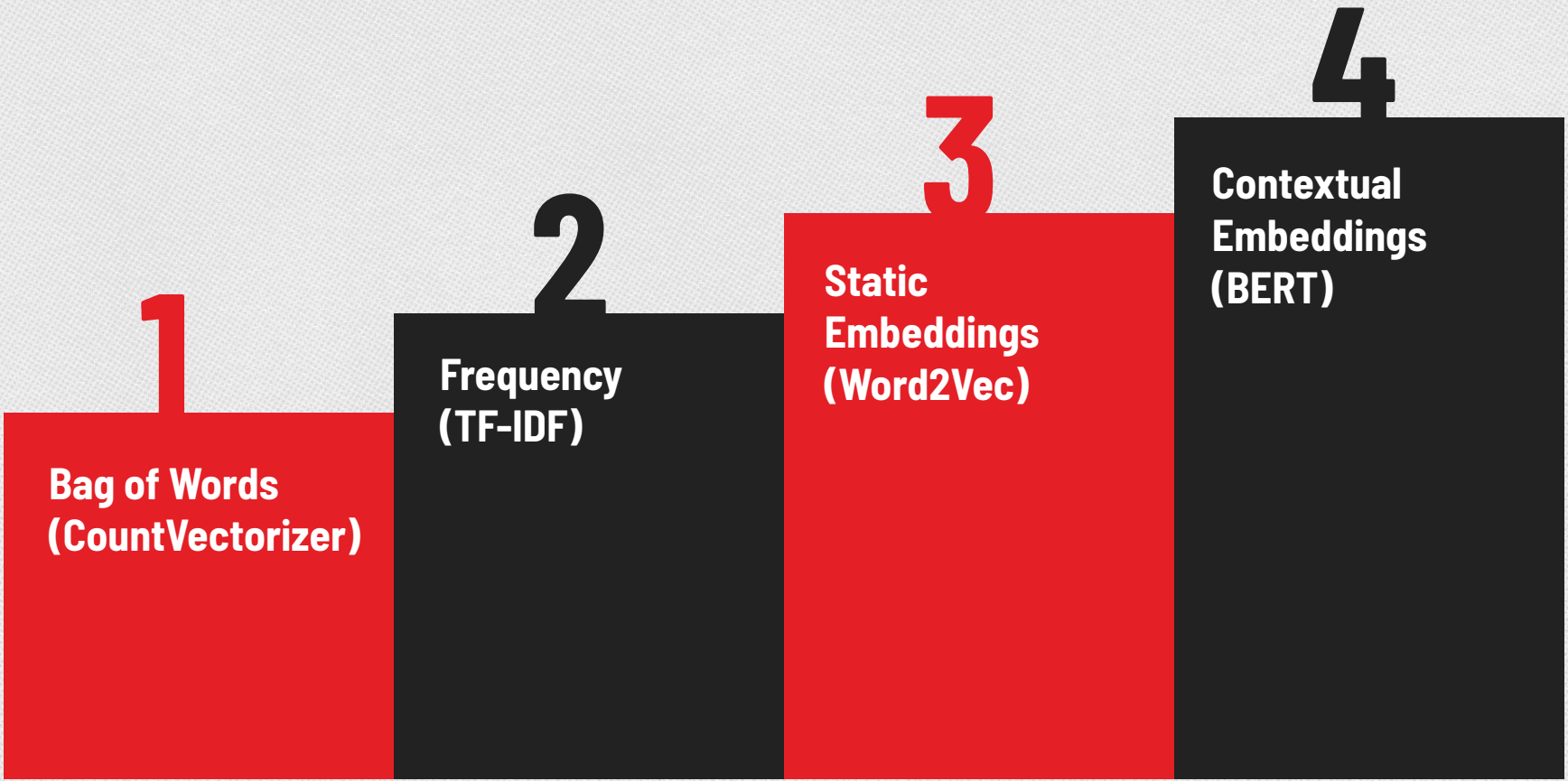
Exploring various text representation methods

During the remainder of this module **you will explore 4 different ways to represent text numerically**. They become progressively more complex and capture more, and more context as we move through each.

Even the simplest method (CountVectorizer) **can be extremely effective** at representing text, particularly when paired with a classification model.

When deciding which method to use, **consider starting with the simplest option and moving to progressively more complex** (and computationally expensive) only if you can't accomplish your goal with a simpler method.





Numerical Text Representation Methods

1

Bag of Words (CountVectorizer)

Counts how frequently a word is used

Let's look at one Bag of Words method: CountVectorizer

Suppose we have three short social media posts:

1. `I love learning about text.`
2. `Text models are fun.`
3. `I love learning at GA. Learning is fun!`

Bag of Words

Bag of words captures the frequency of words but not the order.

One bag of words implementation, CountVectorizer **creates a column for ever n-gram (word here)** that is used in our corpus - the collection of documents.

Then we **count how often words or n-grams were used in each document** (or post in this case) and organize them into a table like this one.
Here we use a single word per column. You can also create bi-grams, which have two words per column.

1. I love learning about text.
2. Text models are fun.
3. I love learning at GA. Learning is fun!

	I	love	learning	about	text	models	are	fun	at	GA	is
I love learning about text	1	1	1	1	1	0	0	0	0	0	0
Text models are fun	0	0	0	0	1	1	1	1	0	0	0
I love learning at GA. Learning is fun!	1	1	2	0	0	0	0	1	1	1	1

Numerical Text Representation Methods

1

Bag of Words (CountVectorizer)
Counts how frequently a word is used

2

Frequency (TF-IDF)
Builds on CountVectorizer to account for how often words our used in your data

TF-IDF

Term Frequency- Inverse Document Frequency takes into account **how often is used with a given document** (or post in this example) **and across our full corpus.** It combines two metrics:

- **Term Frequency (TF):** How often a word appears in a document
- **Inverse Document Frequency (IDF)**: How unique that word is across all documents

This helps identify words that are both frequent and meaningful in specific documents. Full equation is available here

This is helpful for **removing** very rare **words that aren't meaningful** for our analysis as well as very common words that are used in every document many times. These very common words are also called **stop words**.

1. I love learning about text.
2. Text models are fun.
3. I love learning at GA. Learning is fun!

	about	are	at	fun	ga	is	learning	love	models	text
I love learning about text	0.60	0	0	0	0	0	0.46	0.46	0	0.46
Text models are fun	0	0.56	0	0.43	0	0	0	0	0.56	0.43
I love learning at GA. Learning is fun!	0	0	0.39	0.29	0.39	0.39	0.60	0.29	0	0

Text Frequency Jupyter Notebook (text-frequency.ipynb)

Let's walk through CountVectorizer and TF-IDF and use the results to predict customer star ratings based on their reviews of a product. This will serve as proxy for sentiment analysis. Remember even these simple methods can be surprisingly effective.

Jupyter Notebook

- ☐ Create a frequency word vector with CountVectorizer
- ☐ Compare methods to modify n-grams with stop words and length
- ☐ Explore the impact of incorporating TF-IDF
- ☐ Apply methods to predict customer rating based on review text (a proxy for sentiment analysis)

Frequency Methods- Bag of Words (Countvectorizer) and TF-IDF

- Represents text numerically.
- Fast and inexpensive
- Tailored to the data being used
- Capture how frequently words are used in our documents
- Can be used to approximate sentiment and in predictive models as features

❌ Doesn't account for different definitions of words

❌ Can't be used for word comparisons

❌ Doesn't account for the order of words in a phrase or sentence

❌ Can miss important information, such as negations. For example:

- I'm happy!
- I'm **not** happy!

Break time

Text Embeddings

Translating text for computers

Embeddings

Now we have seen **2 methods for representing text CountVectorizer and TF-IDF**. These methods are **surprisingly useful** despite having **no context** to how the word is used in the document.

But sometimes we do **need the context** of how the word is used and even where in a sentence that word falls. These more complex numericals representations of text are called **embeddings**.

Embeddings use vectors of numbers to represent text

Feature Vectors are numerical representations of information that may contain multiple dimensions (can vary in size and shape).

Countvectorizer and TF-ID

use a **single value** to represent a token/n-gram.

```
#countvectorizer
interest =  1

#TF-IDF
interest = 0.23
```

Embeddings

use a **vector of numbers** to represent a token. The dimensions of the vectors vary from model to model.

```
interest =  [  0.43, 0.28, 0.002,
               0.03, 0.02, 0.14,
               0.05, 0.10, 0.14 ]
```


Embeddings uniquely represent text

Embeddings use a vector of numbers to represent a token. These additional numbers allow you to **capture more meaning about the word.**

- What is the definition of the word?
- How is it used in this context?
- How does that change the meaning of text?

Single representation

```
#countvectorizer
interest =  1
#TF-IDF
interest = 0.23
```

Interest (noun)

Definition 1: the state of wanting to know or learn about something or someone.

Definition 2: money paid regularly at a particular rate for the use of money lent, or for delaying the repayment of a debt.

Unique representation

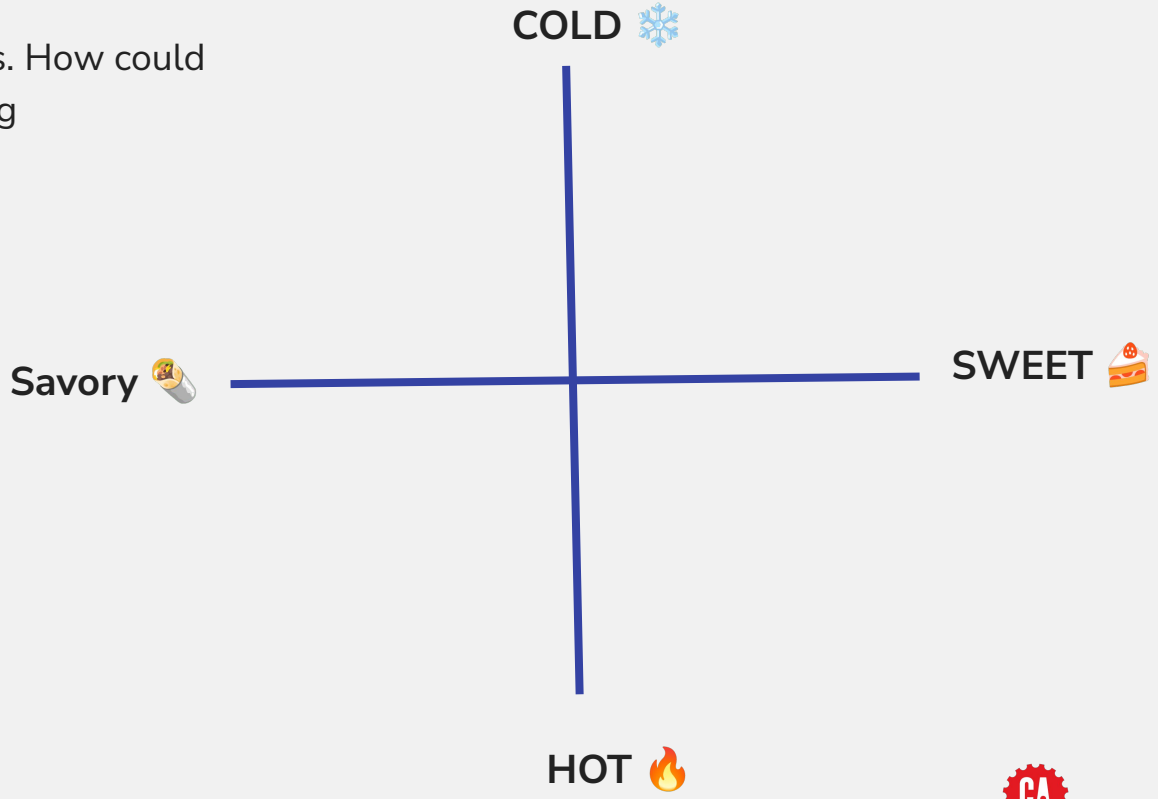
```
interest =  [  0.43, 0.28, 0.002,
               0.03, 0.02, 0.14,
                  0.05, 0.10, 0.14 ]

interest =  [  0.03, 0.15, 0.002,
               0.22, 0.04, 0.06,
                  0.14, 0.01, 0.08 ]
```


Where do the vector numbers come from?

Let's say you had a group of food words. How could you represent those on graph comparing Sweet/Savory and Hot/Cold?

- Steak
- Soup
- Salad
- Coffee
- Pie
- Ice Cream
- Sorbet



Where do the vector numbers come from?

We can do this on multiple dimensions and get a numerical representation based on where words are plotted. This becomes the basis for our vectors. The exact formula varies by model.

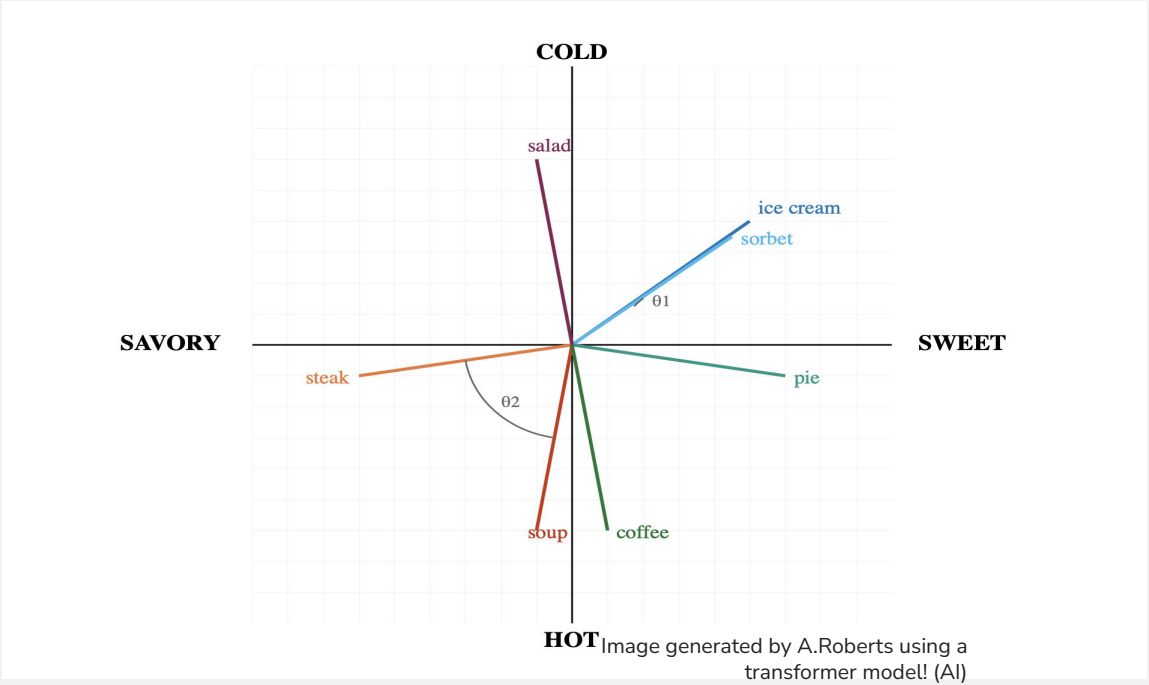


Image generated by A.Roberts using a transformer model! (AI)

Numerical Text Representation Methods

1

Bag of Words (CountVectorizer)

Counts how frequently a word is used

2

Frequency (TF-IDF)

Builds on CountVectorizer to account for how often words our used in your data

3

Static Embeddings (Word2Vec)

An embedding method that accounts for how a word is used

Static Embeddings with Word2Vec

Text embeddings aim to capture the meaning of words by assigning them vectors, where **similar words have similar vectors.** These vectors reflect the meanings or contexts in which the words are used. A given word will have a set vector regardless of the words around it. This is a **static embedding.**

Word2Vec is **one common example of the this type of embedding** that was developed by Google to help with translation tasks. It makes us possible to capture different definitions/uses of words and compare words to each other.

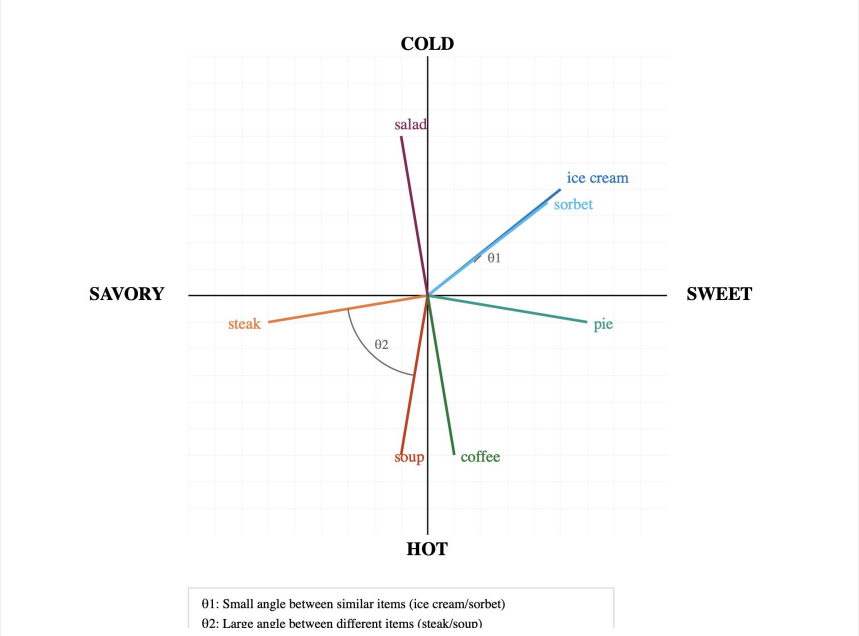


Image generated by A.Roberts using a transformer model! (AI)

Word similarities with Word2Vec

Notice that angles between words can be used to see how similar two items are to each other. **Ice cream and sorbet are very similar to each other, while salad and coffee are very different from one another**. In mathematical terms, we can measure these differences using a cosine similarity.

We can capture this mathematical relationship using a metric such as, **Cosine similarity.**

Cosine Similarity **ranges in values from -1 to 1.**

- Where 1 is an angle (θ) that is close to 0° or perfectly aligned (ice cream and sorbet are close in our example)
- And -1 is 180° or are opposites (coffee and salad in our example).

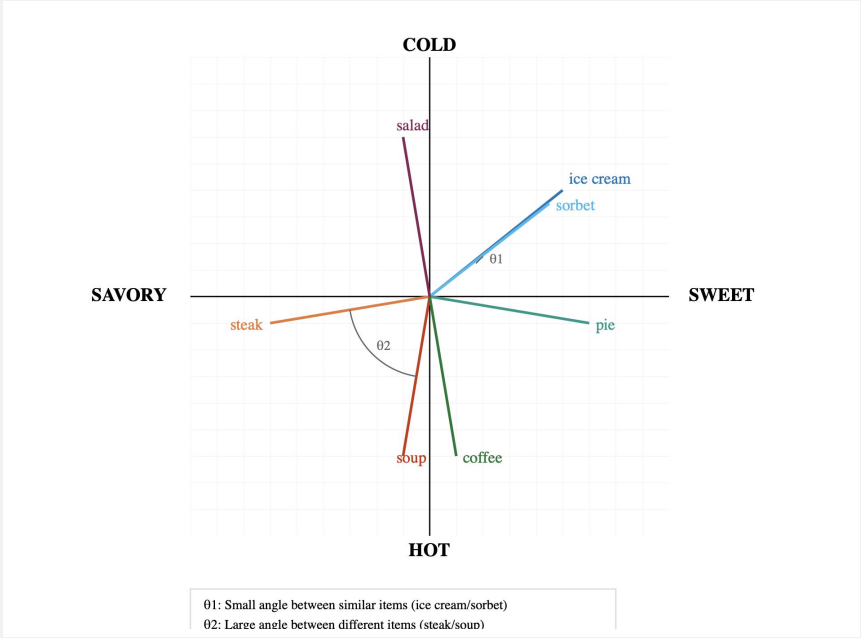


Image generated by A.Roberts using a transformer model! (AI)

Numerical Text Representation Methods

1

Bag of Words (CountVectorizer)

Counts how frequently a word is used

2

Frequency (TF-IDF)

Builds on CountVectorizer to account for how often words our used in your data

3

Static Embeddings (Word2Vec)

An embedding method that accounts for how a word is used

4

Contextual Embeddings (BERT)

A method that additionally accounts for how the word is used and in what sequence

Consider the following two examples

- I'm happy!
- I'm not happy!

➔ "The supplier agreed to pay the manufacturer"

➔ "The manufacturer agreed to pay the supplier"

In both cases **the order of the words impacts the meaning significantly**. Static embeddings, like Word2Vec, can't capture these changes. You will need to a more advanced model to determine the full meaning of the words given the context of the words that were used around it.

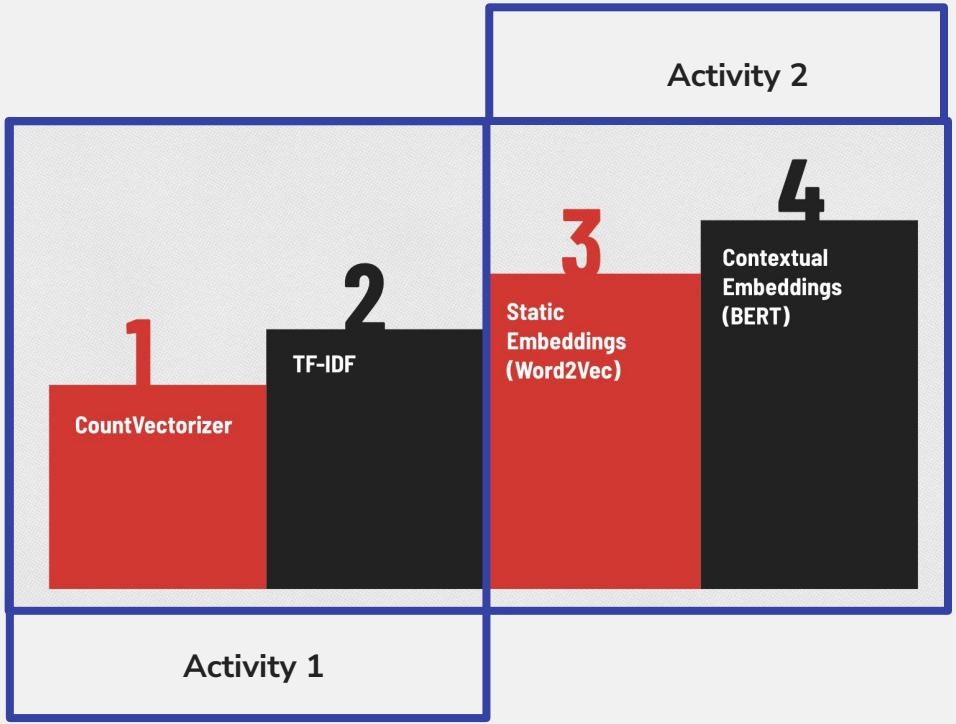
Contextual Embeddings with BERT

BERT was designed to capture the impact of word order. It is used **to create contextual embeddings**

BERT, is a famous language model developed by Google that uses transformer encoder architecture. We will explore this architecture in more detail in future modules. Let's compare the embeddings produced by Word2Vec and BERT in the next activity.

Exploring various text representation methods

In our next activity we compare static contextual embeddings (word2vec) with contextual embeddings (BERT). We will use 2 models that represent each of these (Word2Vec and BERT) but there are other options to explore in the future.



Embeddings jupyter notebook (embeddings.ipynb)

XX MINUTES

You will compare two embedding methods, Word2Vec and BERT.

These are more complicated models that may be necessary for a given task (such as complex sentiment analysis).

In this notebook, you will focus on how each model represents text as vectors.

Jupyter Notebook

- ☐ What's the difference between static and contextual embeddings?
- ☐ Why might you choose static word2vec embeddings over contextual ones?

Static Embeddings - Word2Vec

- Represents text numerically.
- Fast and inexpensive
- Tailored to the data being used
- Accounts for different definitions of words
- Can be used for word comparisons
- Helpful for many tasks including translation, sentiment analysis

❌ Doesn't account for the order of words in a phrase or sentence

❌ Can miss important information, such as negations. For example:

- I'm happy!
- I'm **not** happy!

❌ Doesn't work well on small datasets

Contextual Embeddings - BERT

- Represents text numerically
- Fast and inexpensive
- Tailored to the data being used
- Accounts for different definitions of words
- Can be used for word comparisons
- Account for the order of words in a phrase or sentences
- Because the model is pre-trained on vast amounts of data, works ok in small datasets similar to the training data
- Captures important information, such as negations. For example:
  - I'm happy!
  - I'm **not** happy!

❌ Can be slow and expensive to run on large datasets

❌ Extra complexity may not be required when a simpler, faster tool can accomplish the task equally well

❌ As a pre-trained model, it will contain, biases that may be difficult to detect

Impact of Method Selection

THINK-PAIR-SHARE: Real-World Impacts

What are the impacts of your how you represent your text numerically?

- Outline the +/- of the methods we discussed
- Describe when you may choose a simpler method over a complex contextual one
- Consider the role of data quality and setup choices (such as ,stop word removal) on your final results

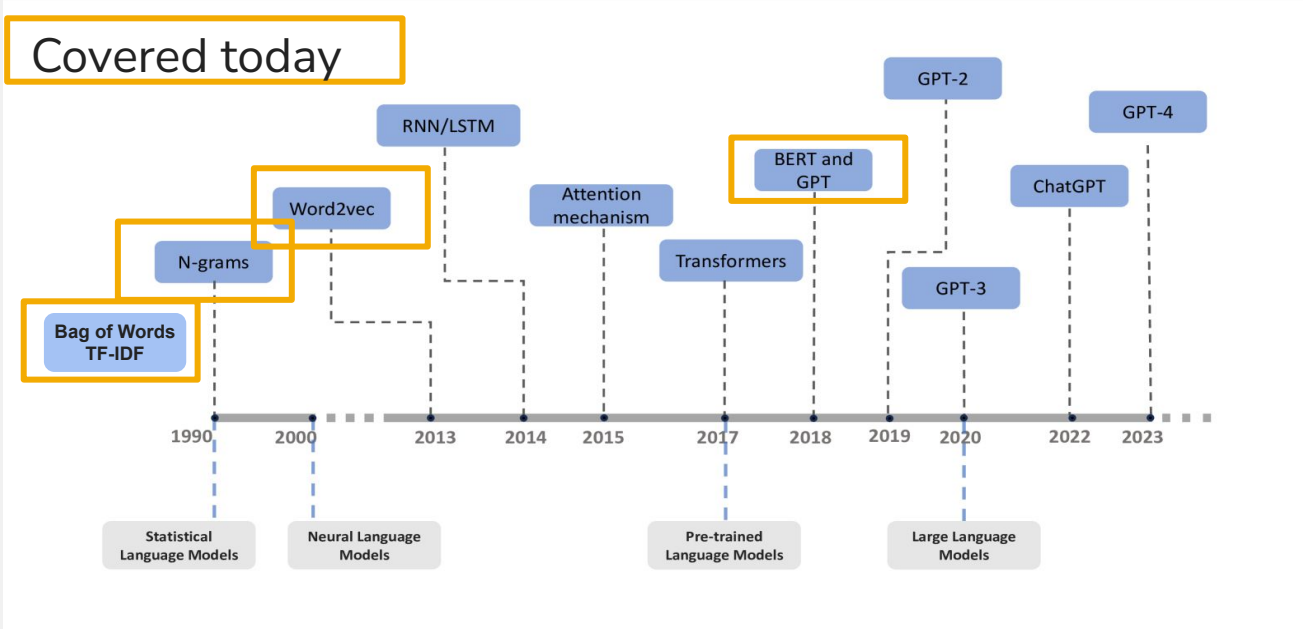
- How do you decide which model to use?
- What is the impact on cost? Processing time?
- Are any methods more prone to bias than others?

Bring It Home

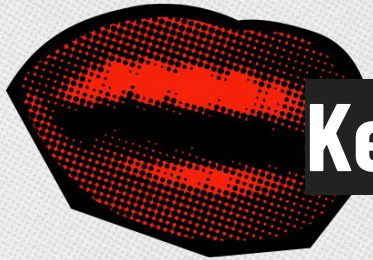
What did we learn?

Timeline of the models you used today

This is a rapidly developing space with new models coming out frequently, each new model building on the foundations of the ones that came before.

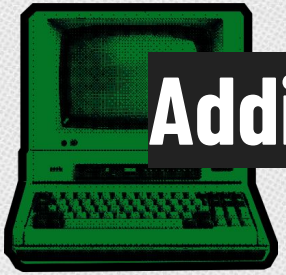


Modified from Source: Wang, Z., Chu, Z., Doan, T. V., Ni, S., Yang, M., & Zhang, W. (2024). History, development, and principles of large language models: an introductory survey. *AI and Ethics*, 1-17. (preprint)



Key takeaways

1. **Define** common use cases for text data
2. **Describe** the evolution of text representation from traditional methods to transformer embeddings
3. **Apply** CountVectorizer, TF-IDF, Word2Vec, and BERT
4. **Summarize** impact of method selection on your analysis



Additional Resources

DIGGING DEEPER

Bag-of-Words

- https://www.ibm.com/think/topics/bag-of-words

Word2Vec

- https://arxiv.org/abs/1301.3781

BERT

- https://arxiv.org/abs/1810.04805

DON'T FORGET:

Exit tickets

Homework Deliverables

What to Do:

- [ ] TBD

What to Turn In:

- [ ] TBD


GA
GENERAL ASSEMBLY