PCSE 595
Special Topics in Machine Learning

Dr. Sam Henry

samuel.henry@cnu.edu

Luter 325

Office Hours

- Please stop by!
- This is an **Open Question Answering Time**

Monday, Wednesday, Friday

11:00-12:00 , 1:00-1:30

Or by appointment

Office hours are in-person (just come by my office, LUTR 325)

Pizza My Mind

- You can get extra credit
  - Up to two extra points on your final grade for one PCSE course
- Attend them! They're fun, informative, and employers present
  - Don't wait until you need a job or internship, go now!
- Thursdays at 12:20

… and you get free pizza!!

Students in PCSE classes can get extra credit if they attend at least 10 events. 10-11 events: 1 extra point; 12-13 events: 2 extra points.

Natural Language Processing (NLP)

- Humans communicate primarily through language
- Vast amounts of knowledge is stored as machine readable natural language text
  - This information is inaccessible to traditional computing methods which require structured data
- NLP provides a method to process textual information and therefore a method to access this human knowledge

- Speech and text is becoming an important form of human-computer interaction

NLP Applications

- Information Retrieval
  - Accessing the documents/text that are relevant to you
- Question Answering
  - Providing answers to questions you ask
- Chat-bots
  - Conversational agents for a variety of purposes (question answering, etc)
- Sentiment Analysis
  - Determining the sentiment (positive, negative, neutral) of text
- Machine Translation
  - Automatically Translating from one language to another
- Text Summarization
  - Creating a summary of multiple and/or lengthy text

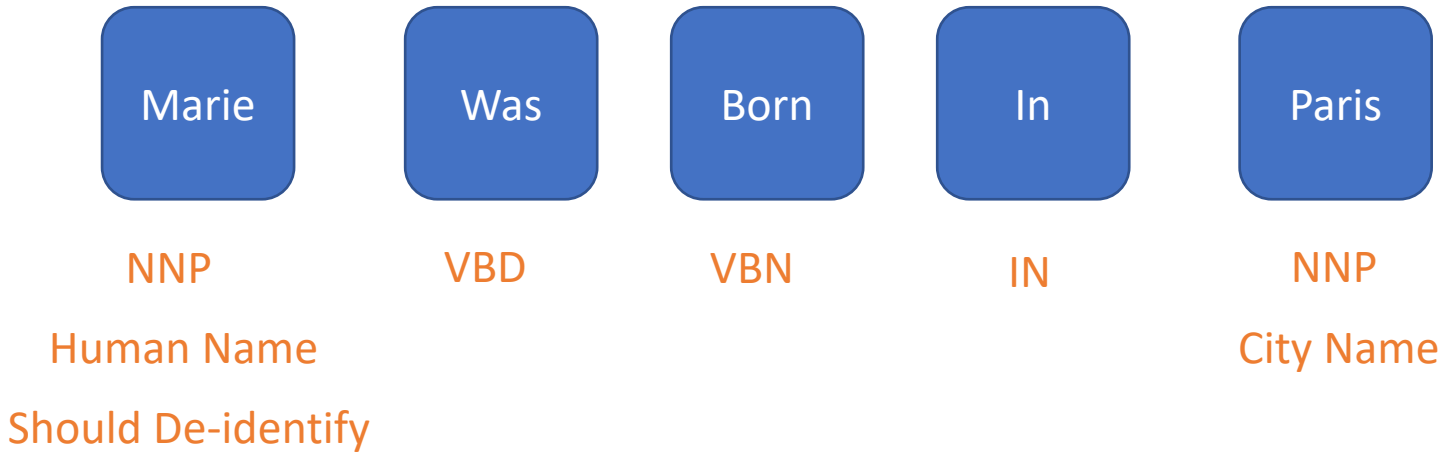
NLP Tasks

- Applications can be solved as pipeline of NLP tasks
  - Question Answering
    - Can pose as Causal Language Modeling (Generative/Abstractive QA)
      - Just one step – predict the next most probable token
    - Or, a pipelined approach: (Extractive QA)
      - Information Retrieval to retrieve relevant documents
      - Snippet Extraction to retrieve relevant sentences
      - Span detection to retrieve the answer within a sentence

Token Classification Tasks

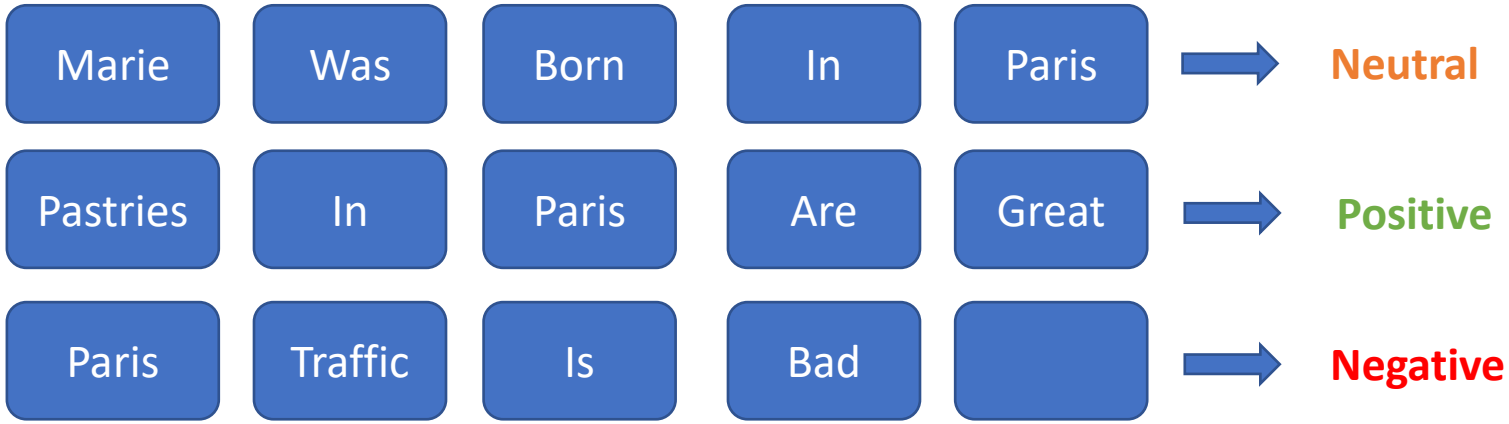
- The goal of token classification tasks is to generate labels for each input token
- Examples include:
  - Part of speech tagging, named entity recognition, word sense disambiguation, de-identification, etc.

Token classification Pipelines can build on each other. For instance, only proper nouns (NNP's) may need to be input into a named entity recognition system



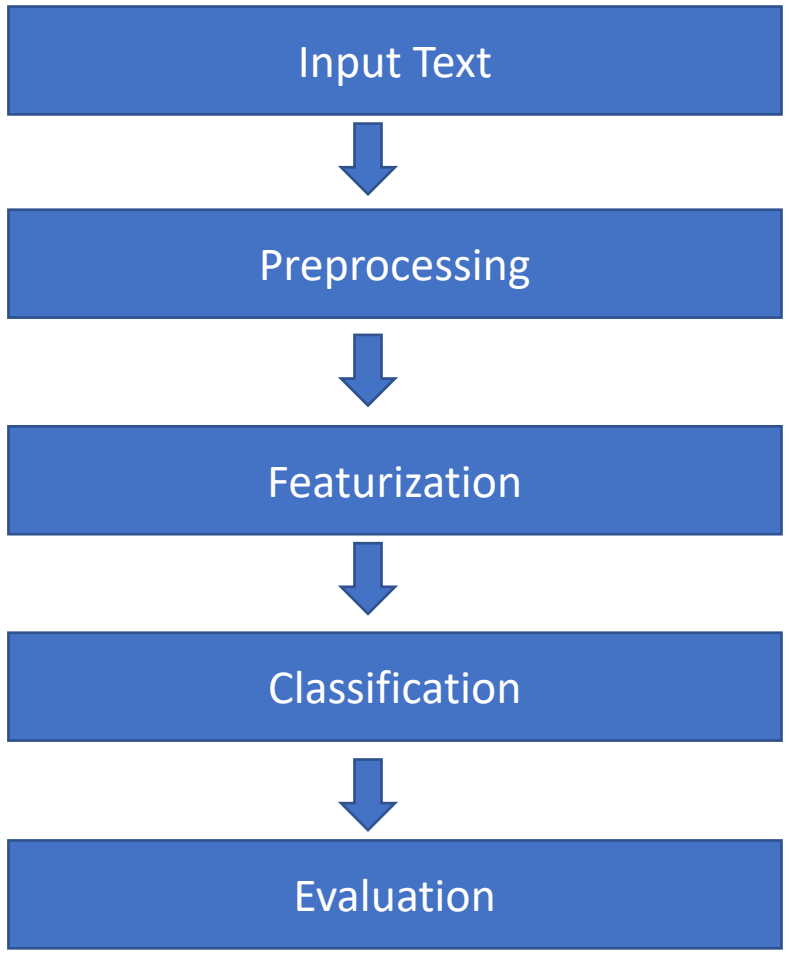
Text/Document Classification Tasks

- The goal of text classification tasks is to generate labels for text.
- “Text” may consist of multiple words and may be phrases, sentences, Tweets, posts, paragraphs, documents, etc..
- Examples include:
  - Information retrieval, sentiment analysis, etc..



Text classification tasks are typically represented as a combination of their constituent tokens. Therefore they often incorporate token classification tasks within their pipeline

NLP Pipeline



Reduce variability and ambiguity

- **Text Cleaning (e.g. spelling correction, stop word removal)**
- **Text Normalization (e.g. stemming, concept normalization)**
- **Adds layers of knowledge (e.g. POS tags, concept normalization) - reduces ambiguity**

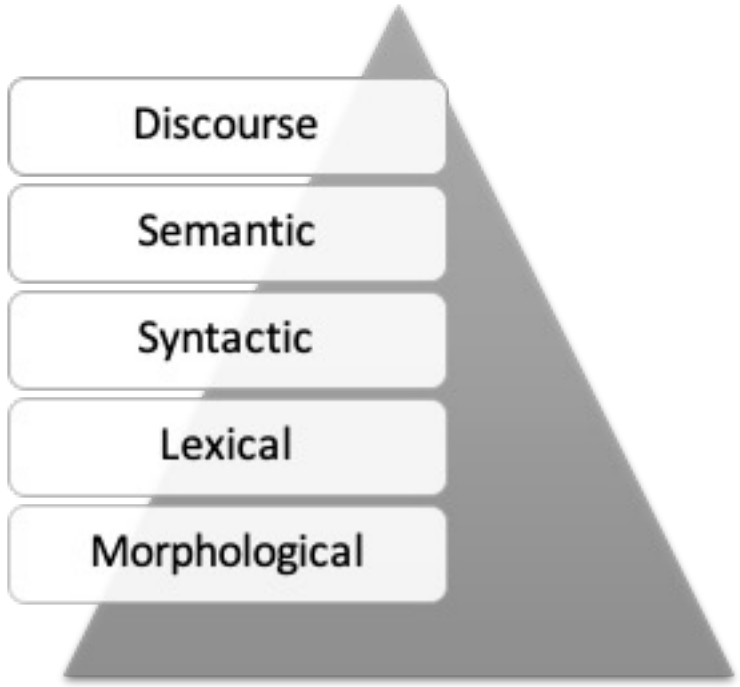
Converts text into numeric values

- **“contains the word smoking”**
- **“count of nouns”**
- **“count of adjectives, etc”**
- **“contains a negated term”**

Machine learning classifier of your choice

A Pyramid of Language Meaning

- Morphological
  - Captures sub-word meaning, such as prefixes and suffixes
- Lexical
  - Captures individual words (called *unigrams*), pairs of words (called *bi-grams*) or sequences of n words (referred to as *n-grams*)
- Syntactic
  - Syntactic information captures how words are structured within a sentence or phrase, such as *part of speech (POS) tags* indicating whether a word is a noun, verb, adjective, etc.
- Semantic
  - Captures the meaning of a word, such as whether it refers to a person's name, a location, a disease, etc.
- Discourse
  - Captures information about how sentences, paragraphs, or documents are structured, such as document sections or document types



A Pyramid of Language Meaning

Syntactic – create a parse tree to determine that "I" is a noun, "walked" is a verb, store is a "noun" in a subject-predicate-object triplet relationship

Subject – Predicate - Object

Noun Verb Noun

Semantic - Map the string "bread" to the concept (abstract meaning) of bread in a dictionary or ontology

"I walked to the grocery store and bought some bread. It was stale"

Morphological – ends in "ed", so past tense

Lexical – the bigram "grocery store" consisting of the tokens "grocery" and "store" is observed frequently, so "grocery store" may be a single term

Discourse: use coreference resolution to determine "It" refers to "bread"

Preprocessing Example

1) The patient displayed symptoms of borderline personality disorder and was diagnosed with BPD

2) ~~The~~ patient displayed symptoms ~~of~~ borderline personality disorder ~~and~~ ~~was~~ diagnosed ~~with~~ BPD

3) ~~The~~ ~~patient~~ displayed symptoms ~~of~~ borderline personality disorder ~~and~~ ~~was~~ diagnosed ~~with~~ BPD

4) ~~The~~ ~~patient~~ display~~ed~~ symptom~~s~~ ~~of~~ borderline personality disorder ~~and~~ ~~was~~ diagnos~~ed~~ ~~with~~ BPD

5) ~~The~~ ~~patient~~ display~~ed~~ symptom~~s~~ ~~of~~ borderline personality disorder ~~and~~ ~~was~~ diagnos~~ed~~ ~~with~~ BPD

6) ~~The~~ ~~patient~~ display~~ed~~ symptom~~s~~ ~~of~~ borderline personality disorder ~~and~~ ~~was~~ diagnos~~ed~~ ~~with~~ BPD

C0006012 C0006012

7) display symptom C0006012 diagnos C0006012

Line (1) shows the original sentence, (2) after stop word removal, (3) after domain specific stop words removal, (4) after stemming, (5) after multi-word terms identified, (6) after concept mapping, and (7) the final preprocessed text. This cleaner and simplified text may make downstream tasks more effective

Featurization

- The goal of featurization is to represent text as numeric values which can be used for classification
- What a “sample” is depends on the application
- Typical samples are:
  1. Words (Terms, Tokens)
     - A sample is a token which may be a word or multi-word term
     - For token classification tasks
  2. Text Snippets, Sentences, Documents
     - Consists of multiple words
     - For text classification tasks

Word Vectors

- One-hot Vectors (Sparse - Vocabulary Size)
  - Vectors are all zero except for a single one at the word's index
- Co-occurrence vectors (Sparse - Vocabulary Size)
  - Iterate over a corpus and collect co-occurrence information. At each co-occurrence increment co-occurrence count at word-word index
- Word Embeddings (Dense – Fixed Size)
  - Iterate over a corpus and use a neural network to create vector representations based on word co-occurrences

"You shall know a word by the company it keeps" ~Firth

One-hot Vectors

Assign each word an index and encode vector with a one at that index

	I	walked	the	dog	in	morning	cat	evening
I	1	0	0	0	0	0	0	0
walked	0	1	0	0	0	0	0	0
the	0	0	1	0	0	0	0	0
dog	0	0	0	1	0	0	0	0
in	0	0	0	0	1	0	0	0
morning	0	0	0	0	0	1	0	0
cat	0	0	0	0	0	0	1	0
evening	0	0	0	0	0	0	0	1

Basic word representation – very large and very sparse vector space. No representation of word meaning

Co-occurrence Vectors

"I walked the dog in the morning"

Window size 4, order matters. Focus word is "the". Increment the co-occurrence matrix for all words in the window ("dog", "in", "the", "morning")

	I	walked	the	dog	in	morning	cat	evening
I	0	1	1	1	1	0	0	0
walked	0	0	2	1	1	0	0	0
the	0	0	1	1	1	1	0	0
dog	0	0	1	0	1	1	0	0
in	0	0	1	0	0	1	0	0
morning	0	0	0	0	0	0	0	0
cat	0	0	0	0	0	0	0	0
evening	0	0	0	0	0	0	0	0

Iterate over a huge corpus to generate a representation of meaning based on word's co-occurrences

Text Segment Vectors

- To represent text segments, sentences, documents, etc.. We typically assume that the text is the combined meaning of its constituent words
  1. Bag of Words Representation = take the sum or average constituent one-hot vectors
  2. Term Frequency – Inverse Document Frequency (TF-IDF) vector = TF-IDF value of each of its constituent terms. TF-IDF value is the number of times a word occurs in that document divided by the number of times it occurs in all documents
  3. Mean or sum of word embeddings

TF-IDF Vectors

Term Frequency – Inverse Document Frequency (TF-IDF) Vectors

Goal is to reduce the weight of frequently occurring terms that convey little meaning and increase the weight of rarer terms that convey more meaning

“The” conveys very little meaning in a bag of words vector

1. Calculate the number of times a term occurs across all documents (document frequency)
2. Calculate the number of times a term occurs in this particular document (term frequency)
3. Weight at that term index = TF-IDF = $\frac{Term\ Frequency}{Document\ Frequency}$

TF-IDF Vectors

"The" occur in 9,000 of 10,000 Tweets

"is" occurs in 5,000 of 10,000 Tweets

"good" occurs in 1500 of 10,000 Tweets

"bad" occurs in 3000 of 10,000

"War" occurs in 100 of 10,000 Tweets

"Garden" occurs in 50 of10,000 Tweets

a= "is the war good?"

is	the	war	good	garden	bad		Word n-2	Word n-1	Word n
1	1	1	1	0	0	…	0	0	0
$\frac{1}{5000}$	$\frac{1}{9000}$	$\frac{1}{100}$	$\frac{1}{1500}$	0	0	…	0	0	0

b="the garden is good good"

is	the	war	good	garden	bad		Word n-2	Word n-1	Word n
1	1	0	1	2	0	…	0	0	0
$\frac{1}{5000}$	$\frac{1}{9000}$	0	$\frac{2}{1500}$	$\frac{1}{50}$	0	…	0	0	0

Bag of Words vector

TF-IDF Vector

c = "the war is bad"

is	the	war	good	garden	bad		Word n-2	Word n-1	Word n
1	1	1	0	0	1	…	0	0	0
$\frac{1}{5000}$	$\frac{1}{9000}$	$\frac{1}{100}$	0	0	$\frac{1}{3000}$	…	0	0	0

Cosine similarity = 1- cosine distance

BOW a,b = 0.5669467095138409

BOW a,c = 0.75

TF-IDF a,b = 0.0012387094595330828

TF-IDF a,c = 0.997232738091998 1

Word Embeddings

- Co-occurrence vectors are large, sparse, and noisy. They contain integer values
- Word embeddings are small(er), dense, and less noisy. They contain real number values

Word embeddings are dense vectors of real-values rather than sparse vectors of binary or integer values. Dimensionality varies, but typically between 50 and 500 dimensions

One-Hot Vectors

	Cat	Dog	Fish	Tree	⋮	House	Home
Cat	1	0	0	0	…	0	0
Dog	0	1	0	0	…	0	0
Fish	0	0	1	0	…	0	0
Tree	0	0	0	1	…	0	0
⋮	…	…	…	…	…	…	…
House	0	0	0	0	…	1	0
Home	0	0	0	0	…	0	1

Word Embeddings

	dim 1	dim 2	dim 3	dim 4	dim 5
Cat	0.5	0.4	0	0.1	0
Dog	0.6	0.3	0.1	0	0
Fish	0.4	0.1	0	0.5	0
Tree	0.2	0	0.7	0.1	0
⋮	…	…	…	…	…
House	0	0	0	0.2	0.8
Home	0.1	0	0.1	0.1	0.7

Word Embeddings

- Word Embeddings are vectors which are the learned internal weights of an artificial neural network
- The network is trained to predict either:
  1. the context given a word (skip-gram)
  2. a word given the context (continuous bag of words (cbow))

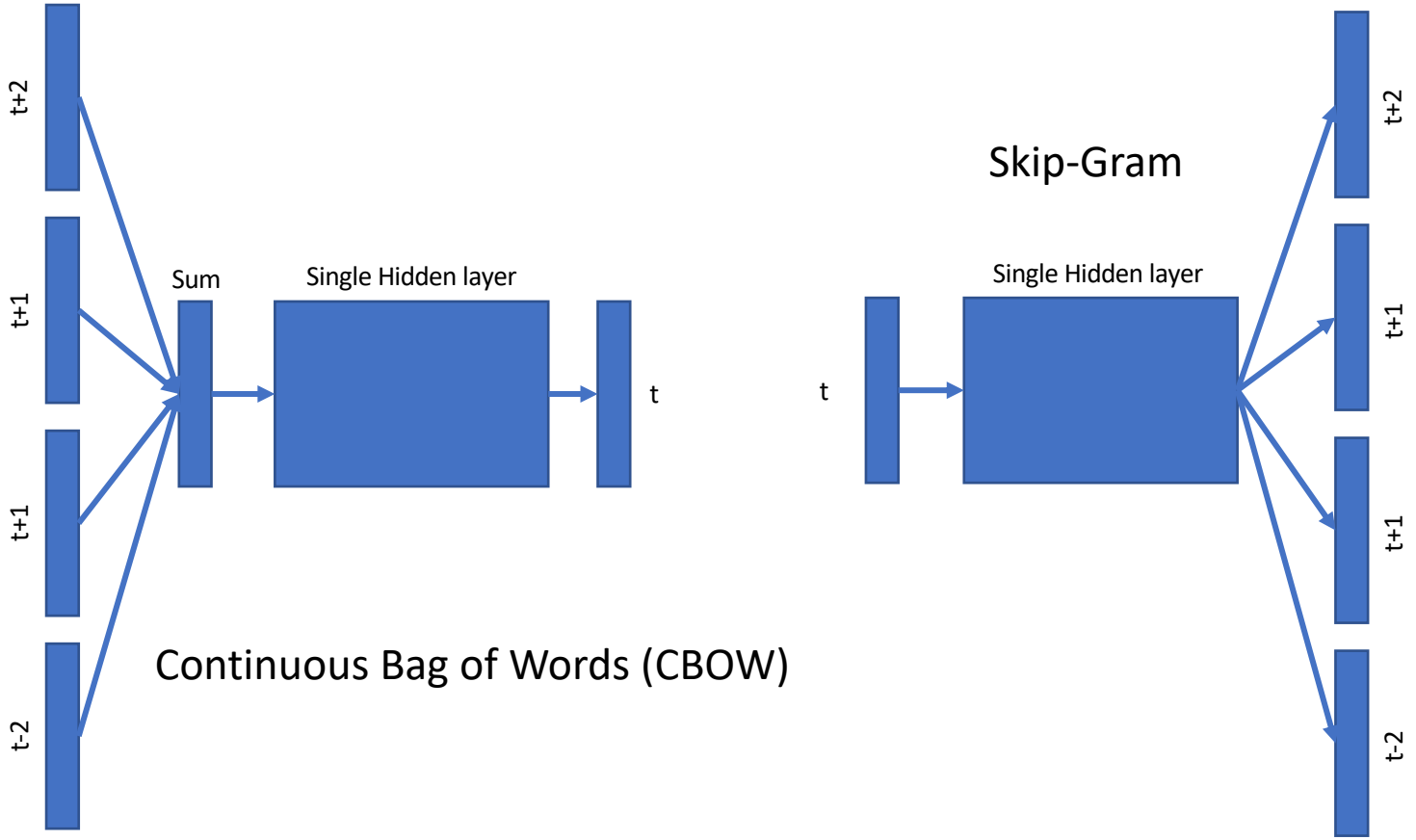
"You shall know a word by the company it keeps" ~Firth

Training Architecture

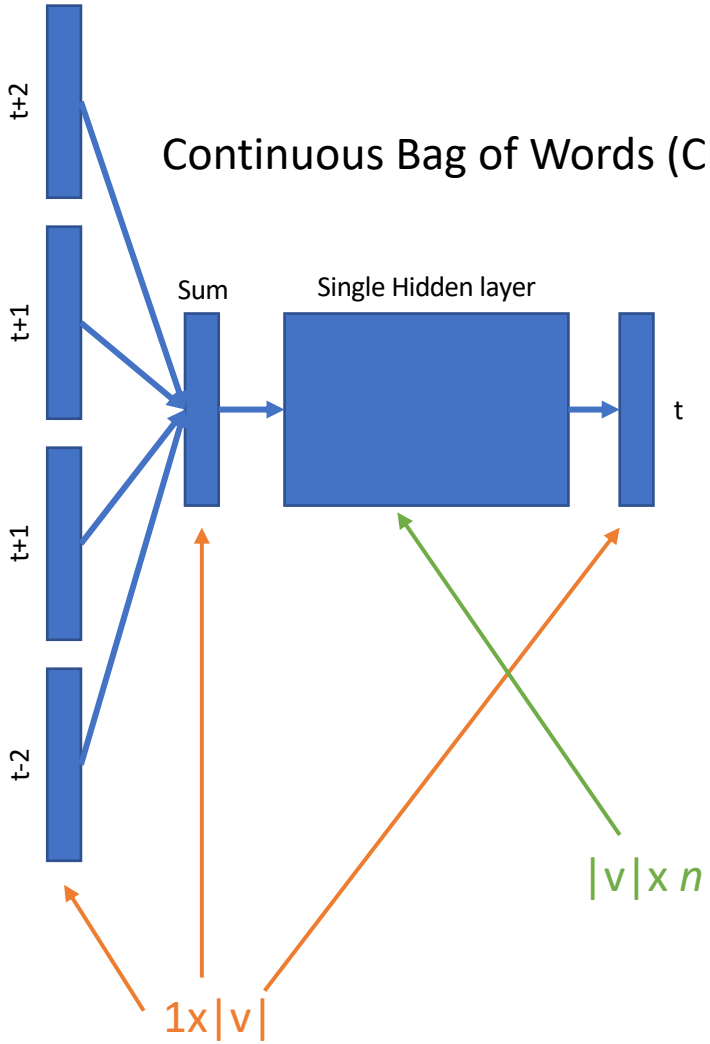
the quick **brown** fox jumped over …

t-2 t-1 t t+1 t+2

Each word is represented by its one-hot vector

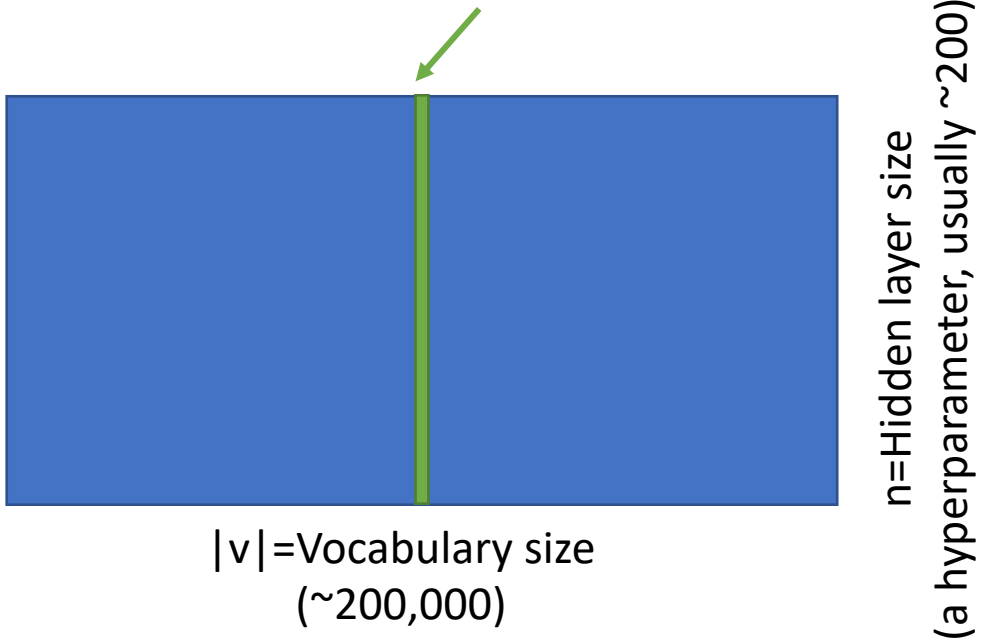


Word Embeddings



The learned weights (input to hidden) encode word meaning in a dense, lower dimensional vector

So, the vector at this index might encode the meaning of “brown”



- Similar words have similar vectors:
  - e.g. Cow and Steer will be more similar than
    Cow and Goat will be more similar than
    Cow and Donut
- Word2vec popularized this, but many other variations (Glove, FastText, etc..)

Recap

- Words can be represented as:
  - One-hot encoding – no meaning is encoded, high dimensional, sparse
  - Co-occurrence vectors – some meaning is encoded, high dimensional, sparse
  - Word embeddings – meaning is encoded, low dimensional, dense

- Text Segments can be represented as
  1. Bag of Words Representation = take the sum or average constituent one-hot vectors
  2. Term Frequency – Inverse Document Frequency (TF-IDF) vector = TF-IDF value of each of its constituent terms. TF-IDF value is the number of times a word occurs in that document divided by the number of times it occurs in all documents
  3. Mean or sum of word embeddings

Token Classification Example: Named Entity Recognition

- Named Entity Recognition is a token classification task
- For each token, decide if it is an entity of interest (e.g. a person's name)

Cindy met Juan Diego-Veracruz at Frank's Hot Dog Stand

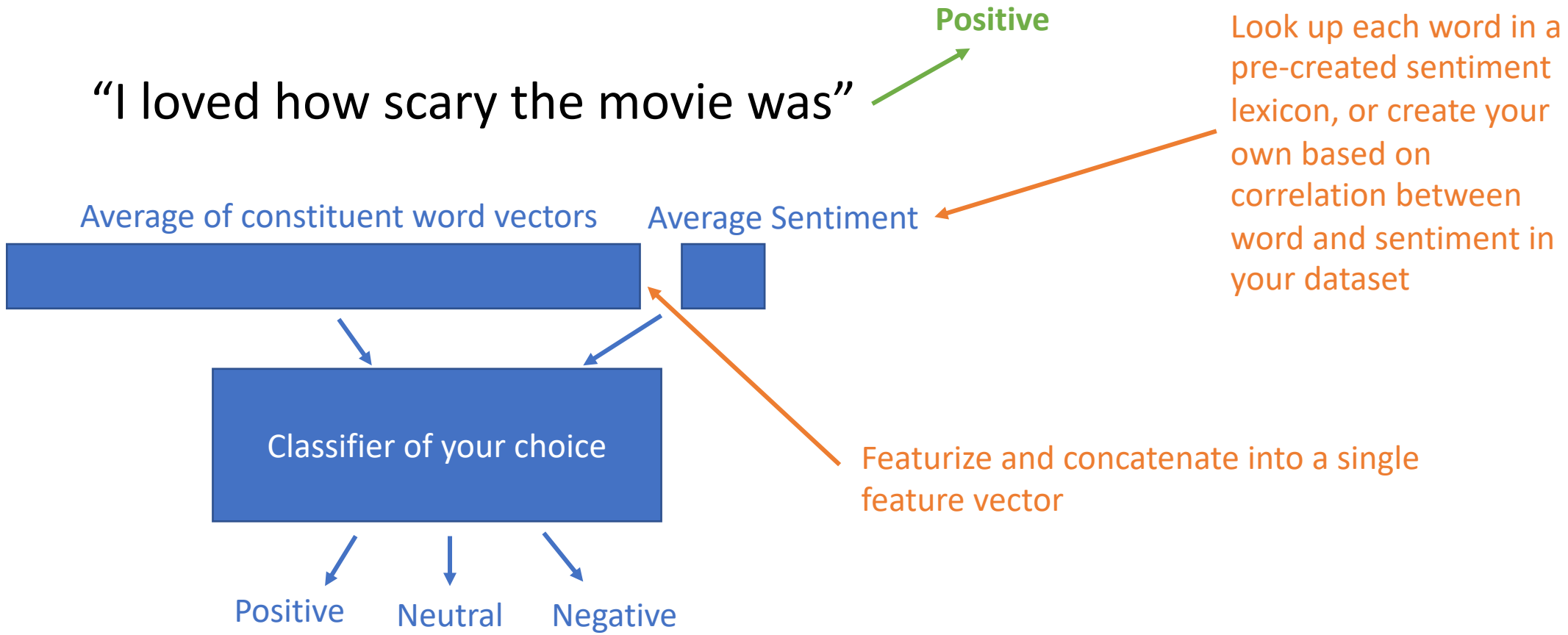
Person's Name

Person's Name

Not a Person's Name

Text Classification Example: Sentiment Analysis Example

- Sentiment Analysis is a text classification task



Lexicon is a list of words – dictionary is a list of words and their definition

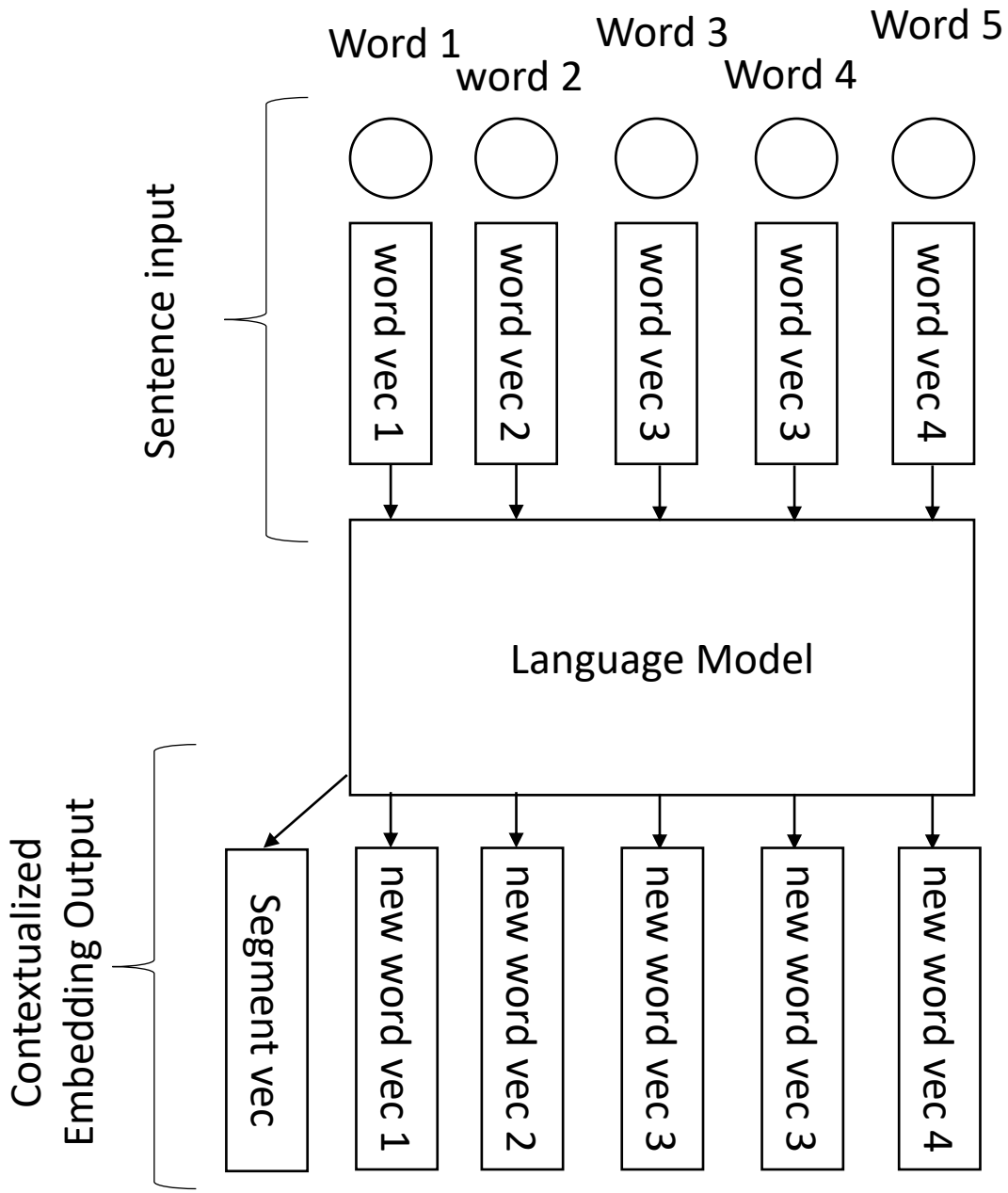
Language Models

- Language models have become ubiquitous in NLP
  - Language models are deep neural networks trained on massive corpora
- Language models generate contextualized word embeddings and (sometimes) text segment embeddings
  - These embeddings consider both the meaning of a word (using word embeddings) and its surrounding context to generate a vector representation of the word's meaning in its current context
  - I ate by the river **bank** vs. I deposited money in the **bank**
  - "bank" has the same word embedding for both contexts, language models add context to the embedding of a word
- A language model is therefore a trained deep neural network that generates a vector representation of a sequence of words

Language Models

- A sentence is input as a series of static word vectors (e.g. word embeddings)
- The language model does a transformation based on its learned weights
- The sentence is output as a series of contextualized embeddings

Then, you can use whatever classifier you want using the contextualized embeddings as input with or without other features



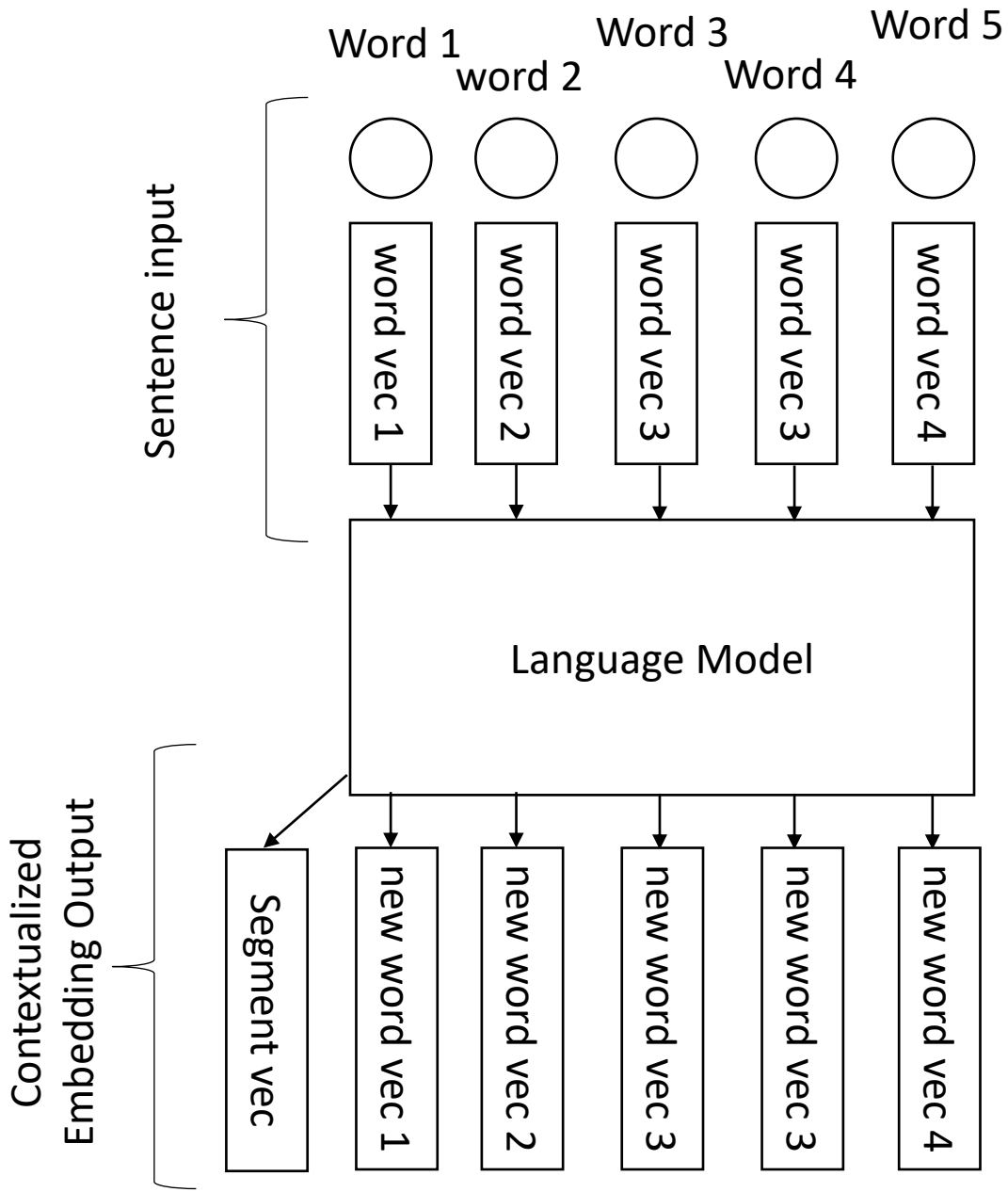
Language Model

Popular language models include:

- eLMO – BiLSTM-based
- BERT – Transformer-based
- GPT – Transformer-based
- LLaMa – Transformer-based

Lots of Others:

- ERNIE
- XLNEt
- Megatron



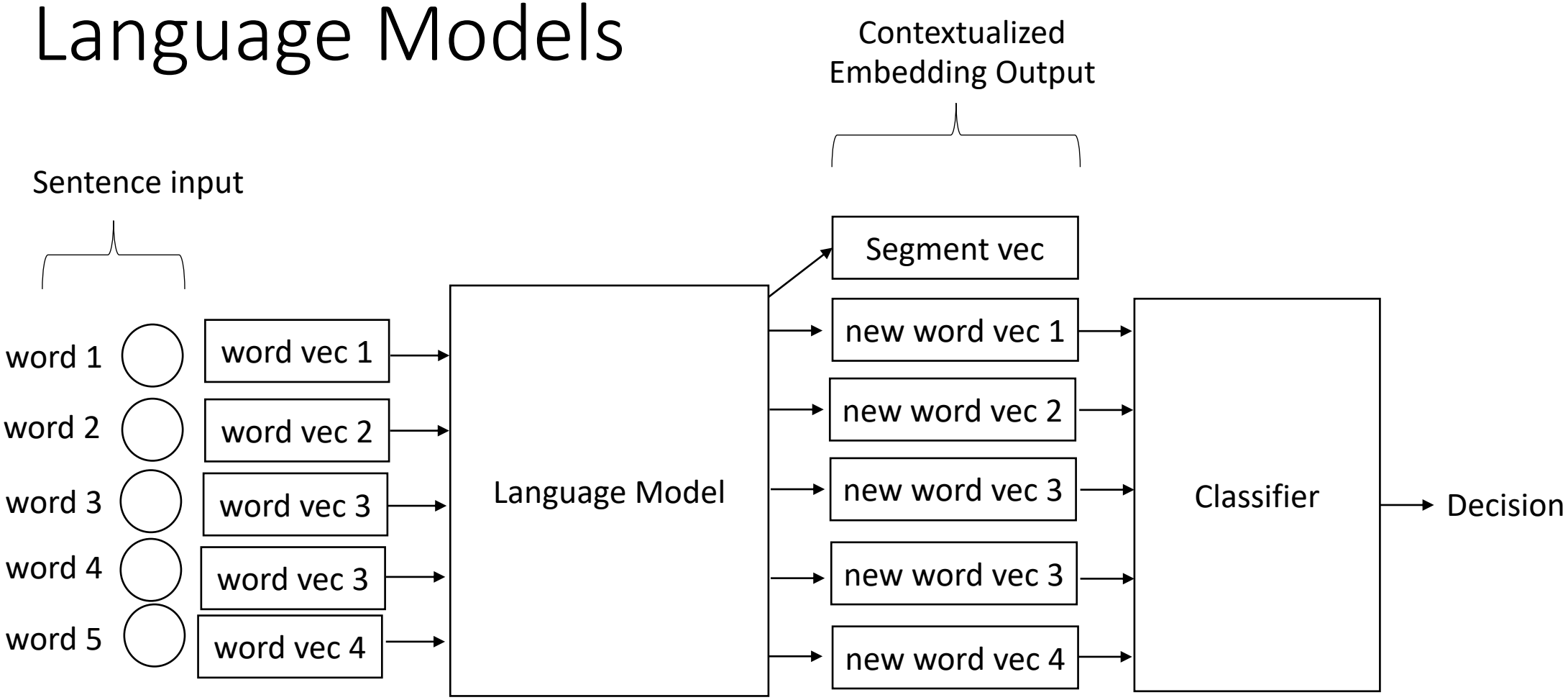
How do the models learn?

Self-Supervised Learning

- These models are pre-trained to predict the next word in a sequence
  - And also sometimes the next sentence in a sequence

- This is really clever and it means that any text can be used as labeled training data

Language Models



You can use whatever classifier you want. The embeddings are just features.

Typically though, neural networks are used, since it allows backpropagation through the whole network (although this is possible with many other classifiers as too)