CMSC 473/673
Natural Language Processing

Instructor: Lara J. Martin (she/they)

TA: Duong Ta (he)

Slides modified from Dr. Frank Ferraro

Learning Objectives

Correct common misconceptions about machine learning

Define a language model

Understand the use & creation of dense vector embeddings

Calculate the distance between vector embeddings

Misconceptions

Continual/Lifelong Learning vs “Regular” Machine Learning

Baselines

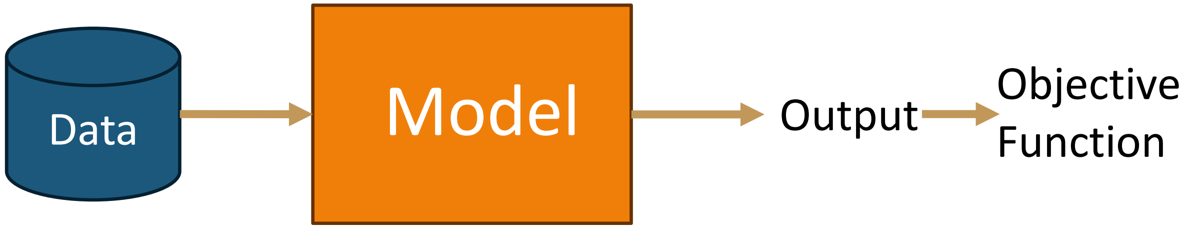
Determining a goal vs evaluation metrics

Language Models

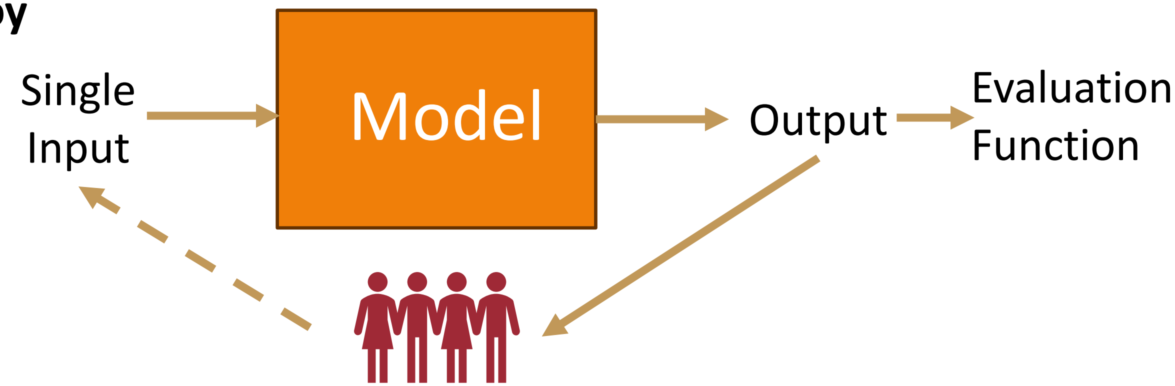
Continual Learning vs Machine Learning

“STATIC” MACHINE LEARNING

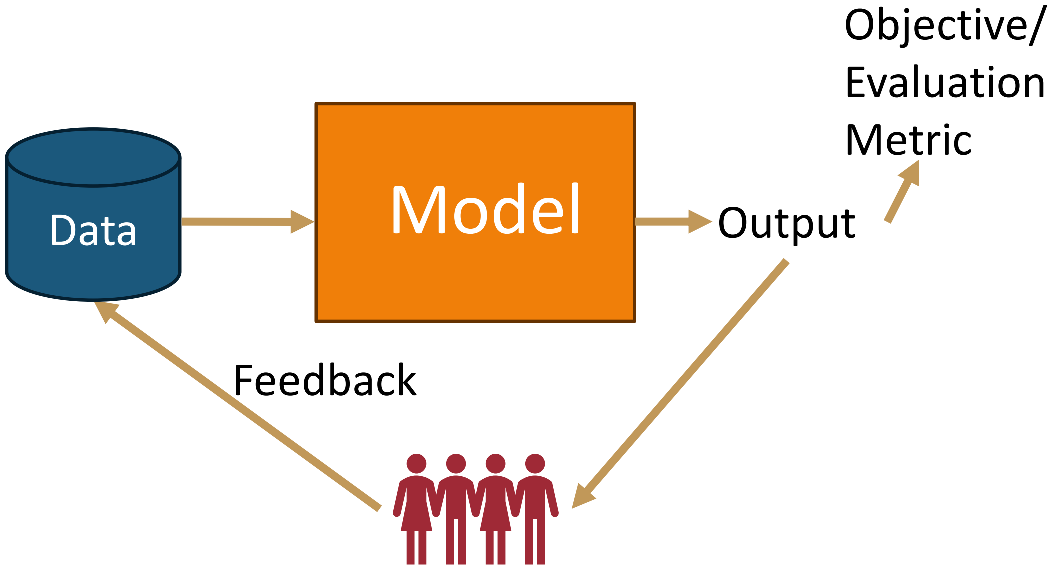
1) Train



2) Test/Deploy

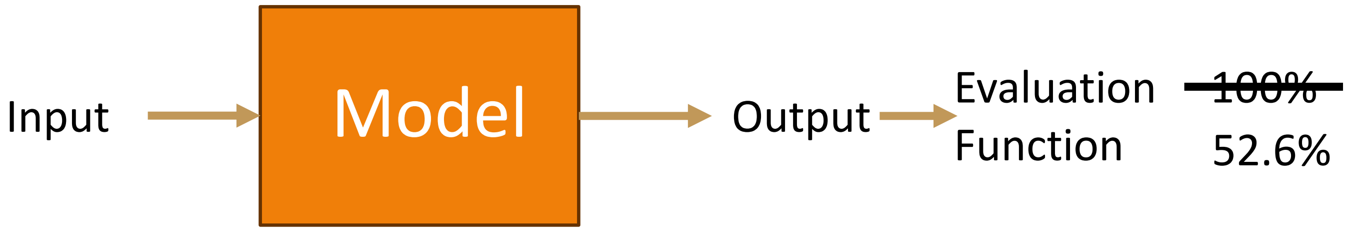


CONTINUAL MACHINE LEARNING

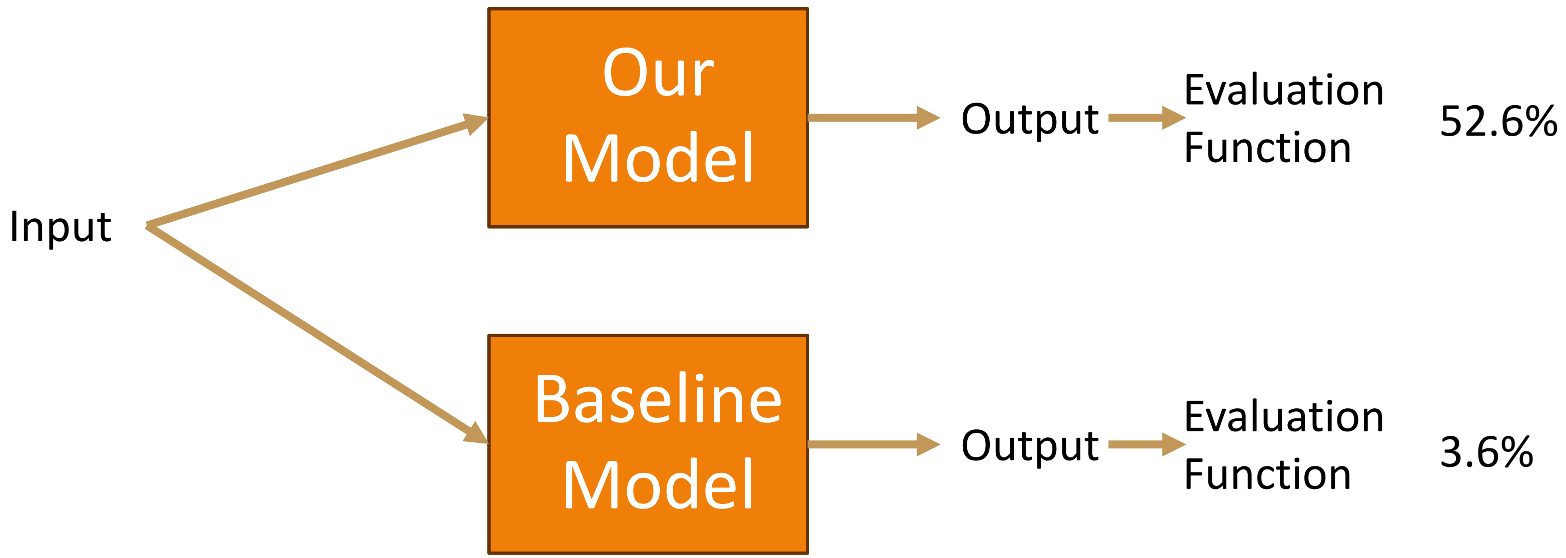


Determining how good a model is

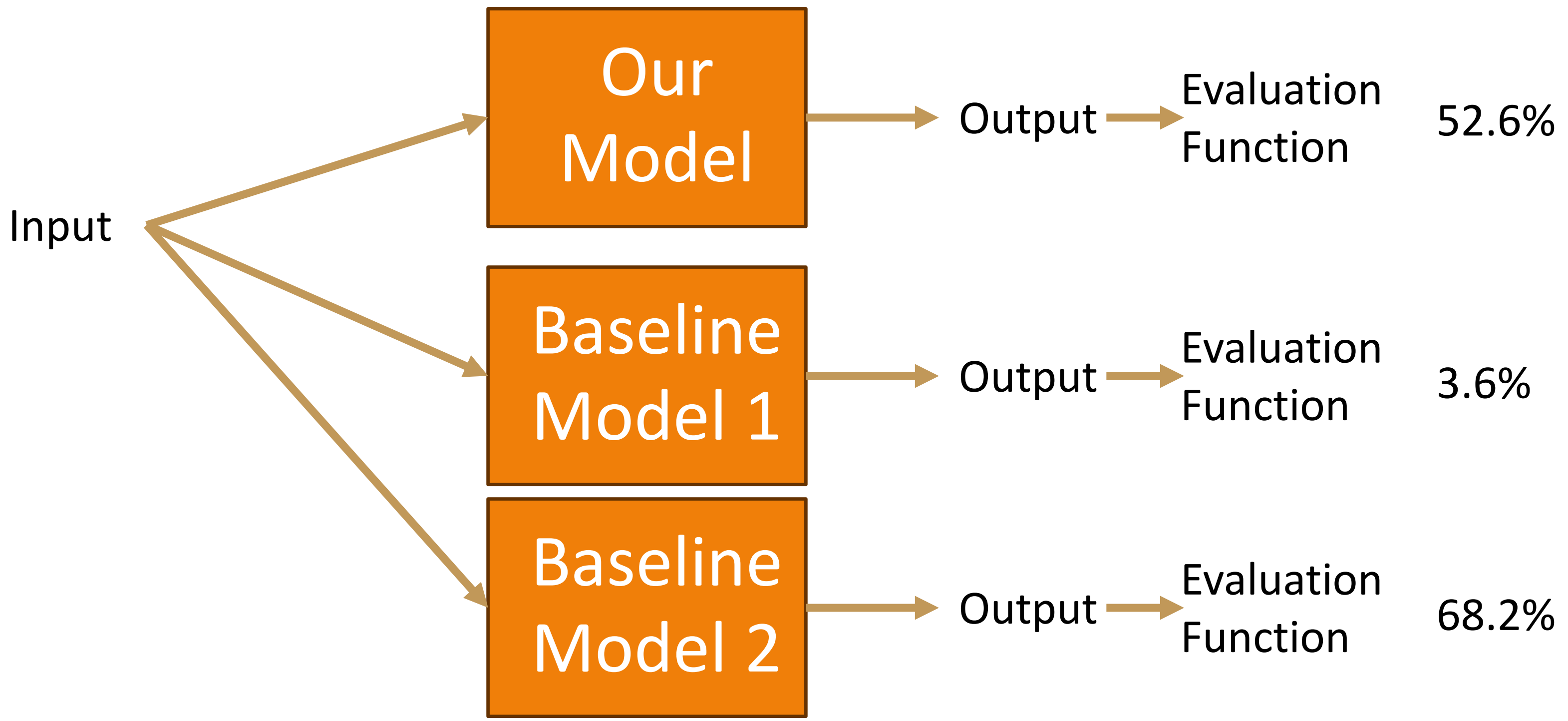
2) Test



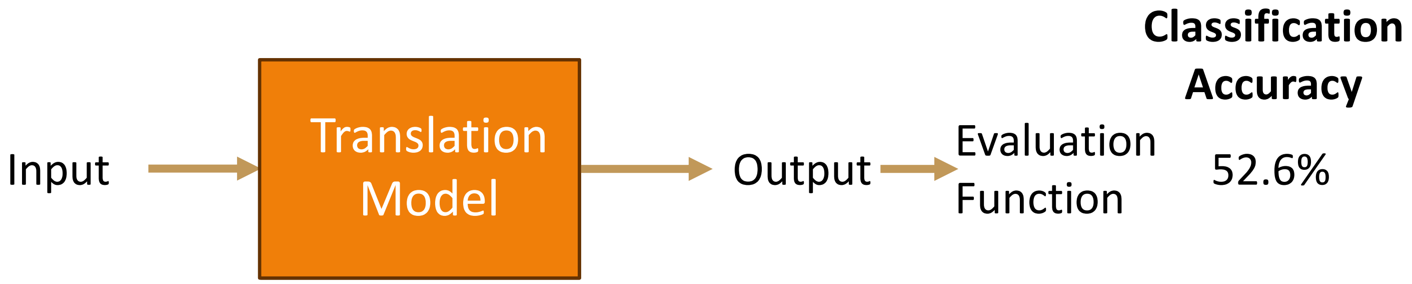
Determining how good a model is: Baselines



Determining how good a model is: Baselines



Determining how good a model is: Evaluation Metric vs Goal



What are you evaluating?

- How good is the model at translating from Mandarin to Twi?
- How accurate is the model at translating the word “potato” across languages?

- How good is this model at classifying correct grammatical form?
- How good is the model at translating new terms?

Bonus Misconception: Data References

If it's cited in a paper:

In Text

In this paper, we use ROC Stories (Mostafazadeh et al., 2016), which is a dataset…

Reference

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., & Allen, J. (2016). A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 839–849. http://www.aclweb.org/anthology/N16-1098

Bonus Misconception: Data References

If it's not cited in a paper (i.e., just online/on Github/on 🤗):

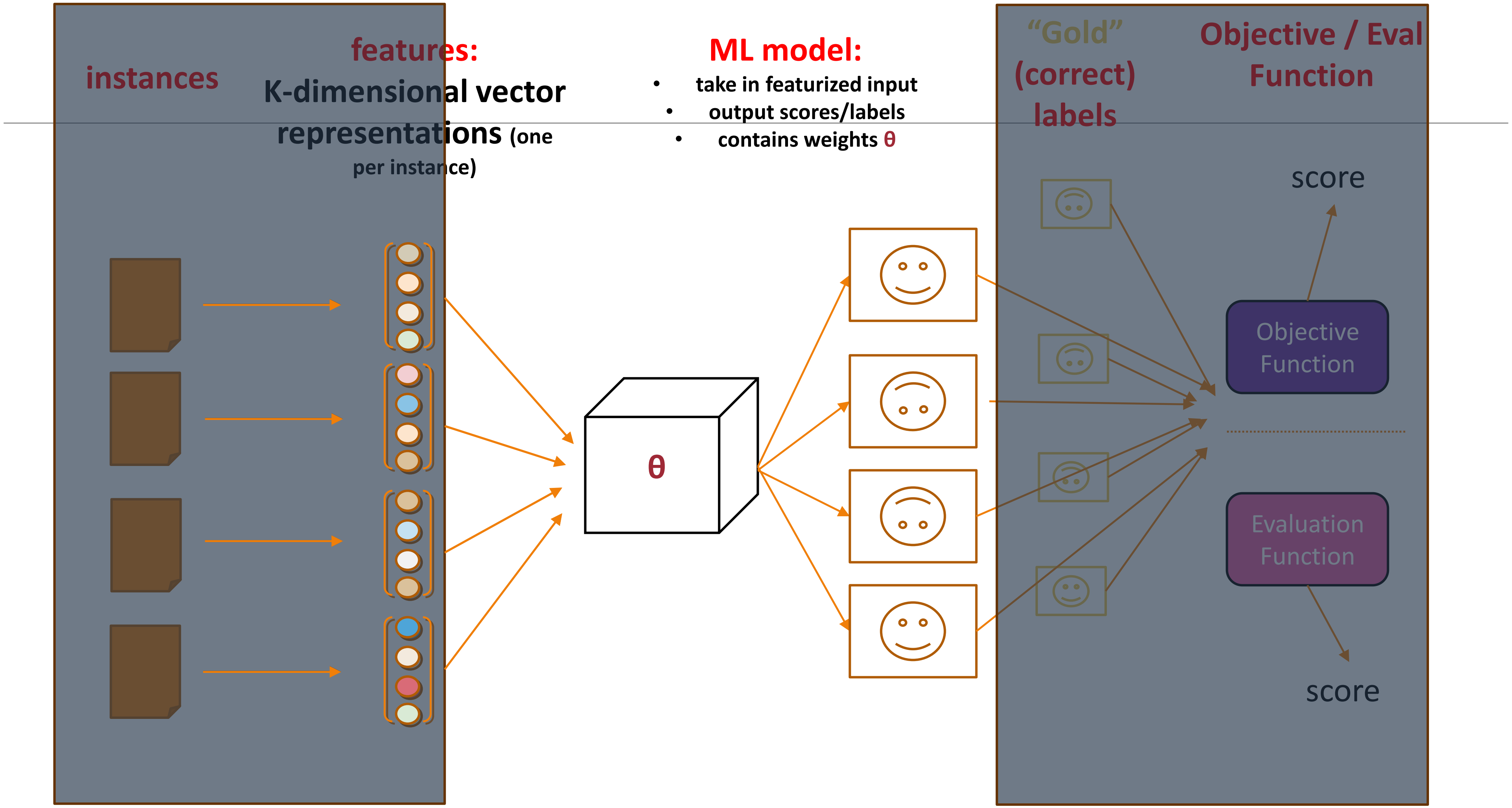
In Text

We scraped story plots from Fandom wikis[1]

<u>Footnote</u>

[1] https://www.fandom.com/

Defining the Model



Modeling

PollEv.com/laramartin527

Can a language model do classification?

Classification $$P(y \mid x)$$

Language Model (LM) $$P(w_t | w_{t-1}, w_{t-2} \cdots)$$

A language model is used to **generate** the next word(s) given a history of words.

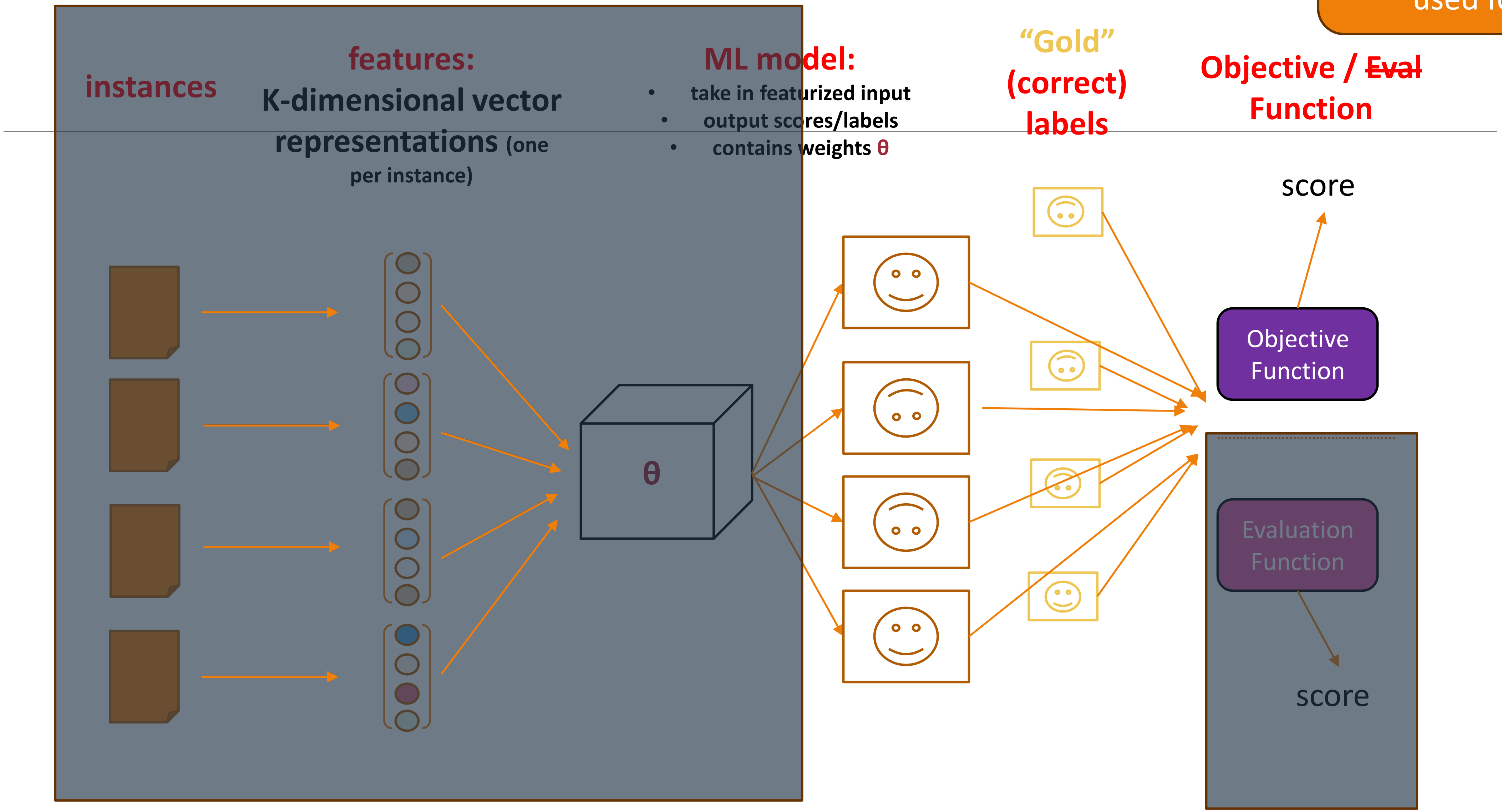
More about LMs after spring break

Can a language model do classification?

Either answer could be correct!

Defining the Objective

What is the objective function used for?



$$p_\theta(y \mid x)$$ probabilistic model

$$\Downarrow$$

$$F(\theta; x, y)$$ **objective**

Review: *Maximize* Log-Likelihood

$$\log \prod_i p_\theta(y_i|x_i) = \sum_i \log p_\theta(y_i|x_i)$$

$$= \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$

$$= F(\theta)$$

Review: *Minimize* Cross Entropy Loss

Cross entropy:
How much $\hat{y}$ differs from the true y

Classifier output

True probability (i.e., correct output)

$$L^{\text{xent}}(\hat{y}, y) =$$

index of "1" indicates correct value

$$\begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{pmatrix}$$

one-hot vector

objective is convex
(when f(x) is not learned)

Review: Classification Log-likelihood (max) ≅ Cross Entropy Loss (min)

$$F(\theta) = \sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i)$$

CROSSENTROPYLOSS

CLASS `torch.nn.CrossEntropyLoss(weight=None, size_average=None, ignore_index=-100, reduce=None, reduction='mean')` [SOURCE]

This criterion combines `LogSoftmax` and `NLLLoss` in one single class.

It is useful when training a classification problem with C classes. If provided, the optional argument `weight` should be a 1D *Tensor* assigning weight to each of the classes. This is particularly useful when you have an unbalanced training set.

The *input* is expected to contain raw, unnormalized scores for each class.

input has to be a Tensor of size either $(minibatch, C)$ or $(minibatch, C, d_1, d_2, ..., d_K)$ with $K \geq 1$ for the *K*-dimensional case (described later).

This criterion expects a class index in the range $[0, C-1]$ as the *target* for each value of a 1D tensor of size *minibatch*; if *ignore_index* is specified, this criterion also accepts this class index (this index may not necessarily be in the class range).

The loss can be described as:

$$\text{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

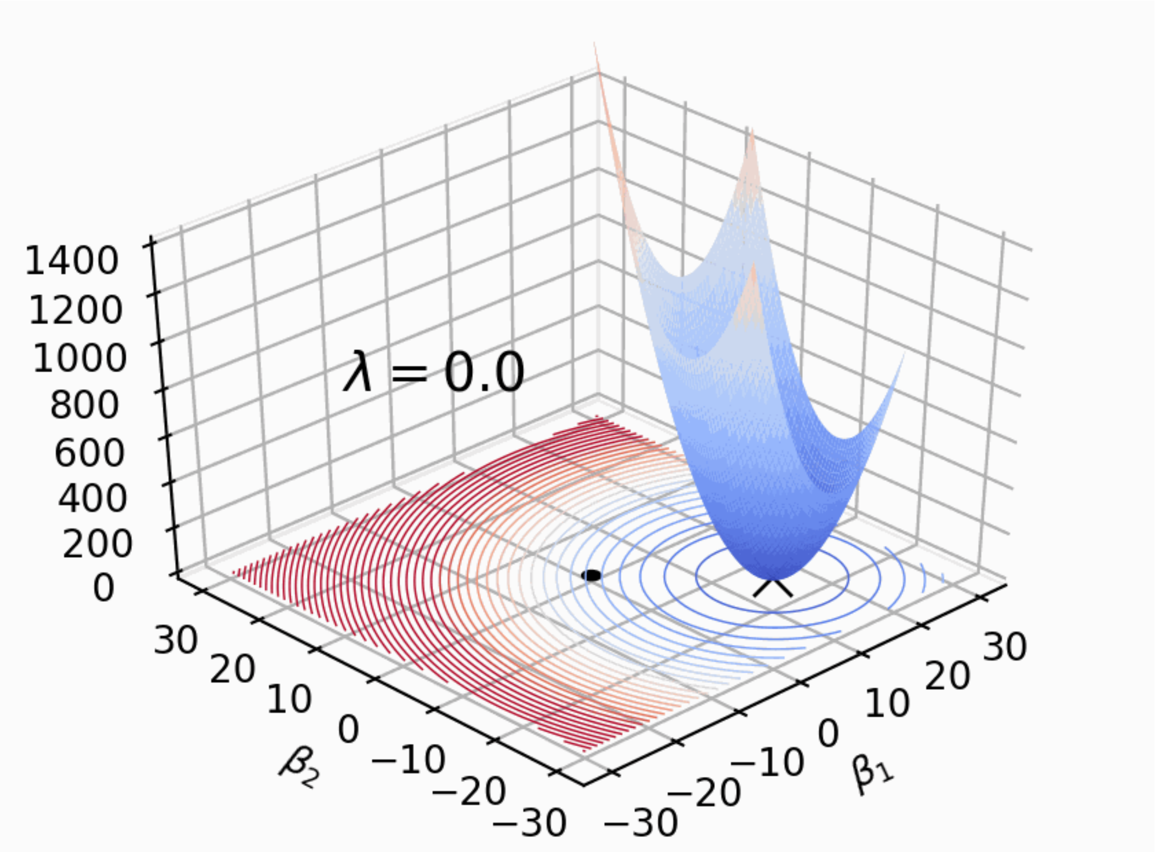
Review:
Regularization: Preventing Extreme Values

$$F(\theta) = \left(\sum_i \theta_{y_i}^T f(x_i) - \log Z(x_i) \right) - R(\theta)$$

With fixed/predefined features, the values of θ determine how "good" or "bad" our objective learning is

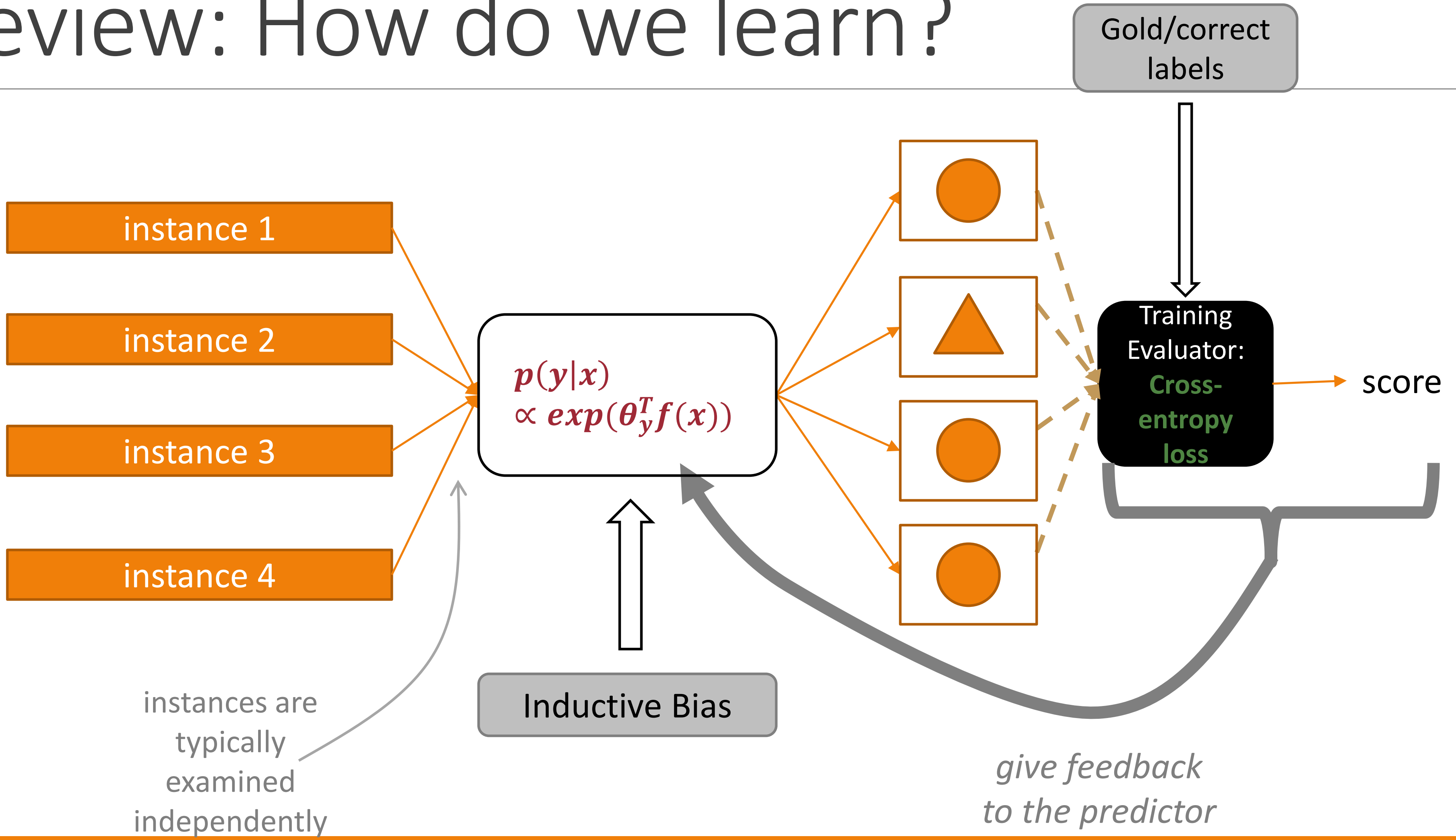
- Augment the objective with a **regularizer**
  - This regularizer places an inductive bias (or, prior) on the general "shape" and values of θ

Review: (Squared) L2 Regularization

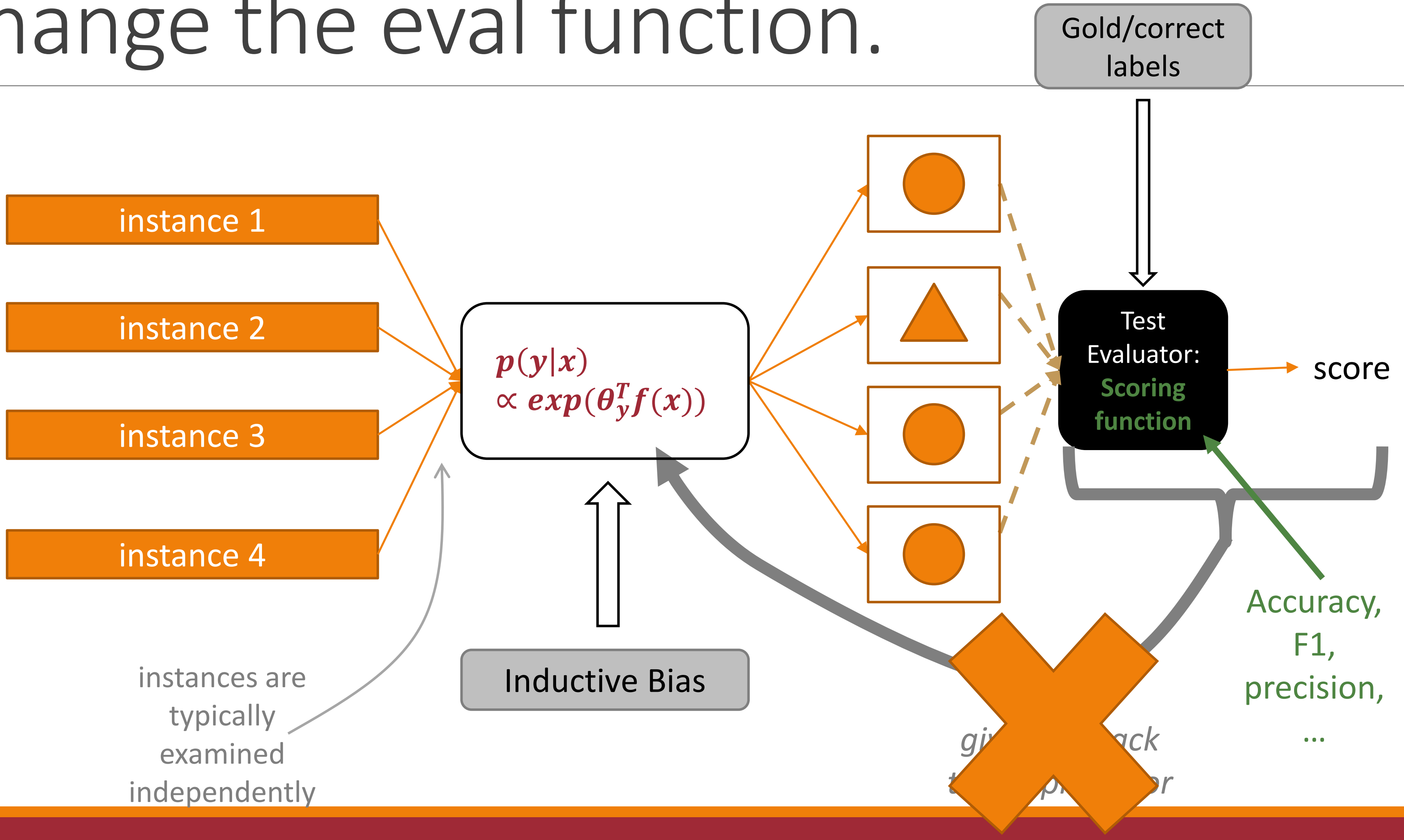


$$R(\theta) = \|\theta\|_2^2 = \sum_k \theta_k^2$$

Review: How do we learn?



Review: How do we evaluate (or use)? Change the eval function.



Review: What if you can't find the roots? Follow the **gradient**

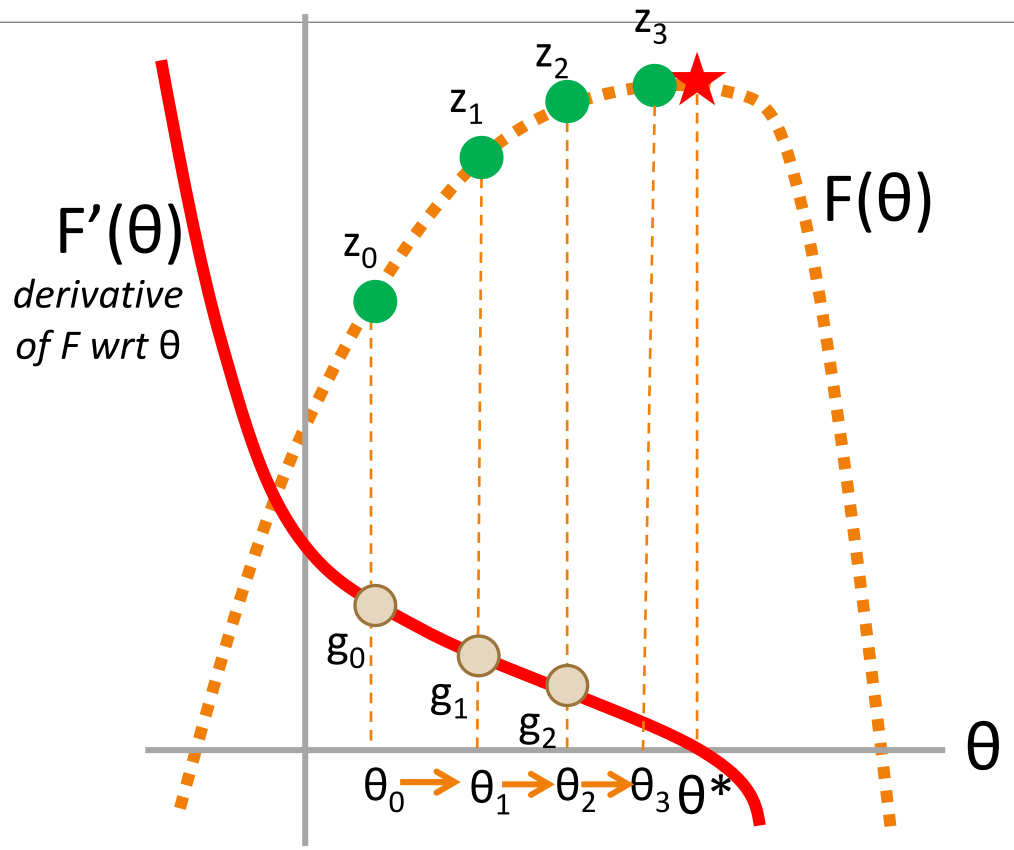
Set t = 0

Pick a starting value θ_t

Until converged:

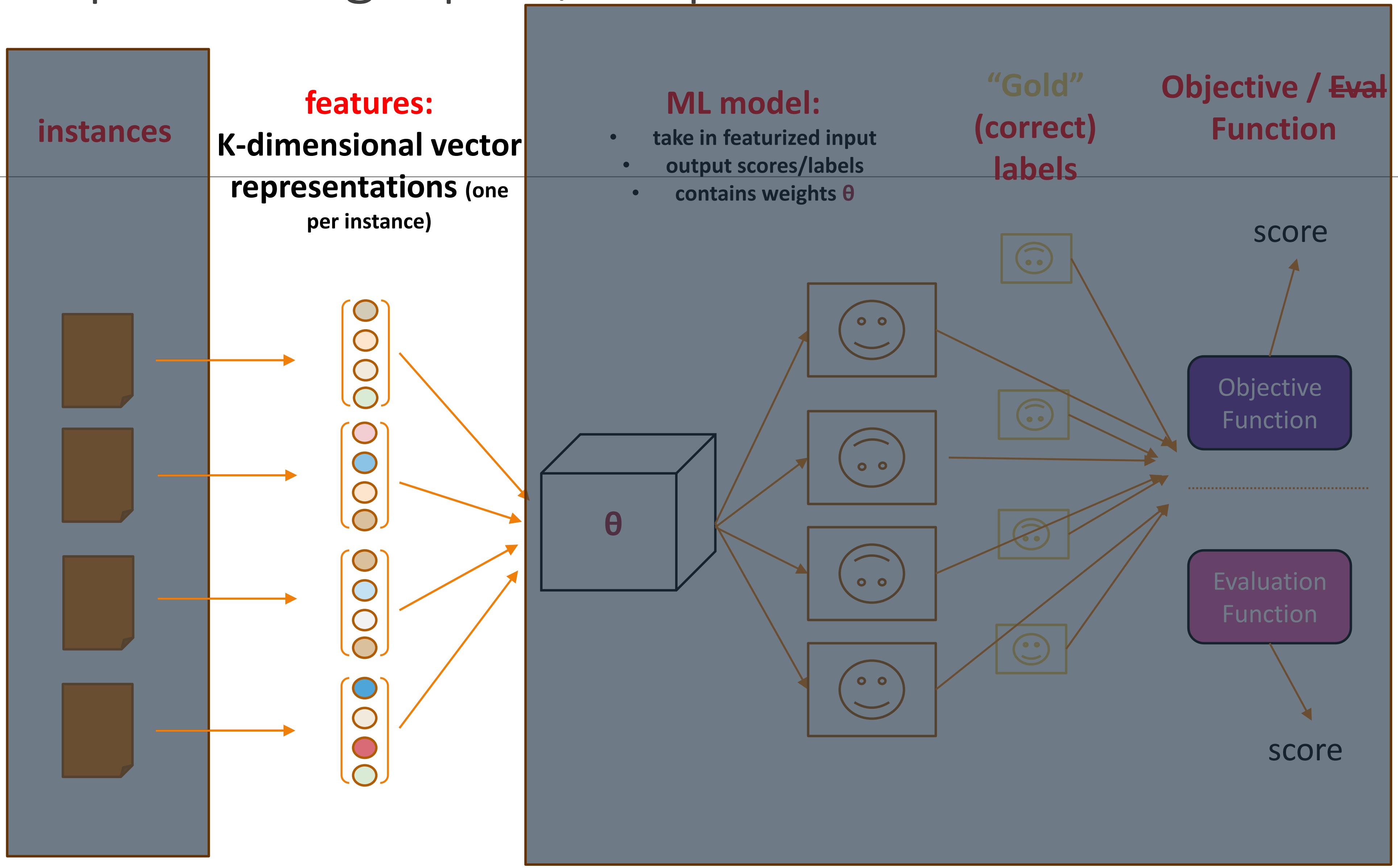
1. Get value $z_t = F(\theta_t)$
2. Get **gradient** $g_t = F'(\theta_t)$
3. Get scaling factor ρ_t
4. Set $\theta_{t+1} = \theta_t + \rho_t * g_t$
5. Set t += 1

K-dimensional vectors

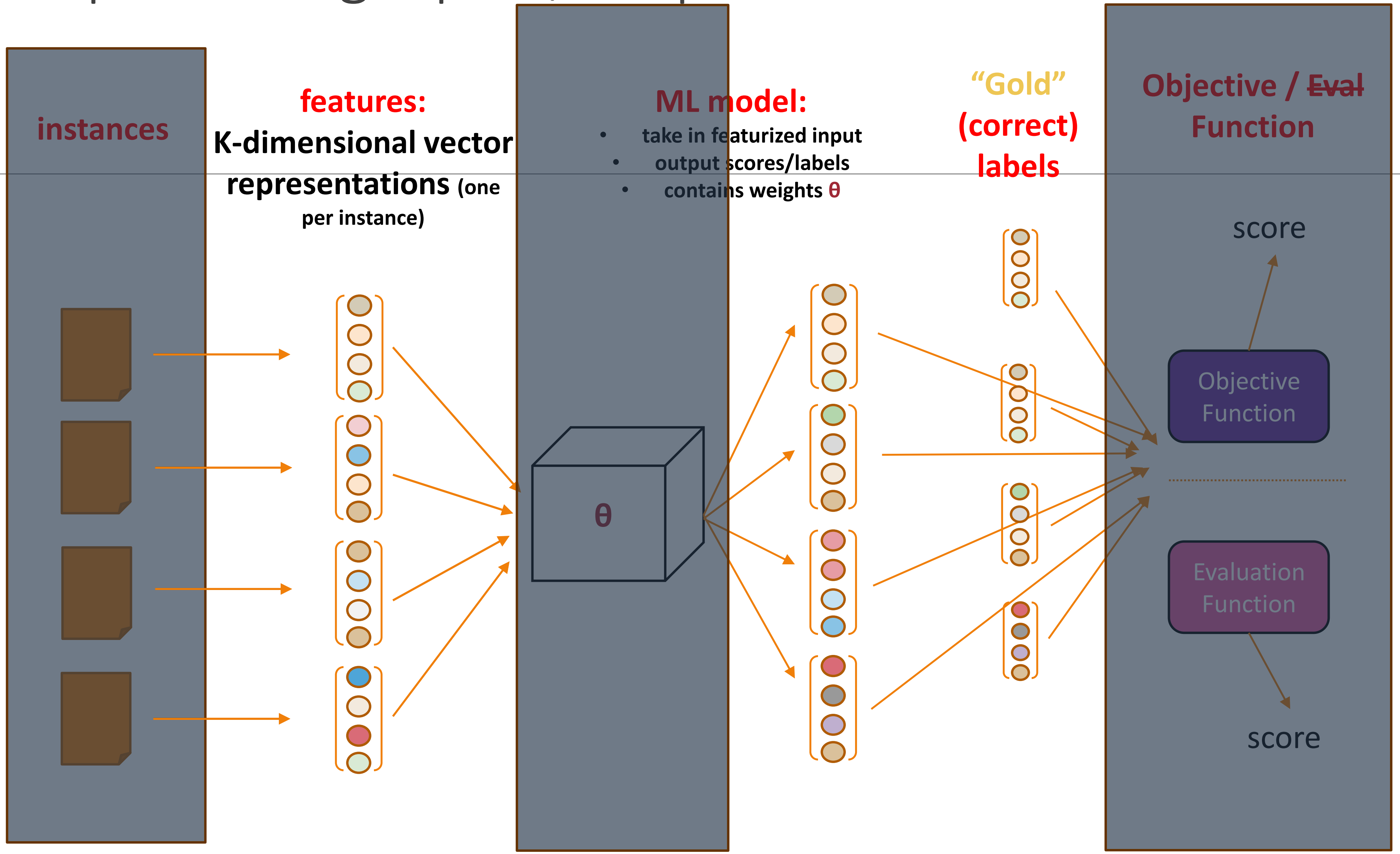


Embeddings

Representing Inputs/Outputs



Representing Inputs/Outputs



How have we represented words?

Each word is a distinct item

- Bijection between the strings and unique integer ids:
- "cat" --> 3, "kitten" --> 792 "dog" --> 17394

- Are "cat" and "kitten" similar?

Equivalently: "One-hot" encoding

- Represent each word type w with a vector the size of the vocabulary
- This vector has V-1 zero entries, and 1 non-zero (one) entry

One-Hot Encoding Example

Let our vocab be {a, cat, saw, mouse, happy}

V = # types = 5

Assign:

a	4
cat	2
saw	3
mouse	0
happy	1

How do we represent "cat?"

$$e_{\text{cat}} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

How do we represent "happy?"

$$e_{\text{happy}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The Fragility of One-Hot Encodings
Case Study: Maxent Plagiarism Detector

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

What is/are the:

Method/steps for predicting?

General formulation?

Features?

There's no way you'll catch me!

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Maxent Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any}-\text{common}-\text{word},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<word v>},\text{Plag.}}(x_1, x_2) = ???$$

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Maxent Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any-common-word,Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<word v>,Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<ngram Z>,Plag.}}(x_1, x_2) = ???$$

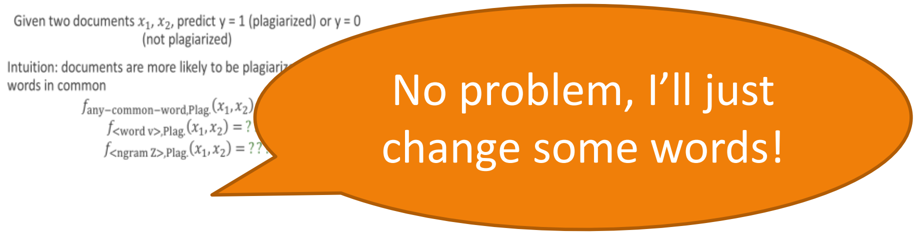


Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Maxent Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any}-\text{common}-\text{word},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<word v>},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<ngram Z>},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{synonym}-\text{of}-\text{<word v>},\text{Plag.}}(x_1, x_2) = ???$$

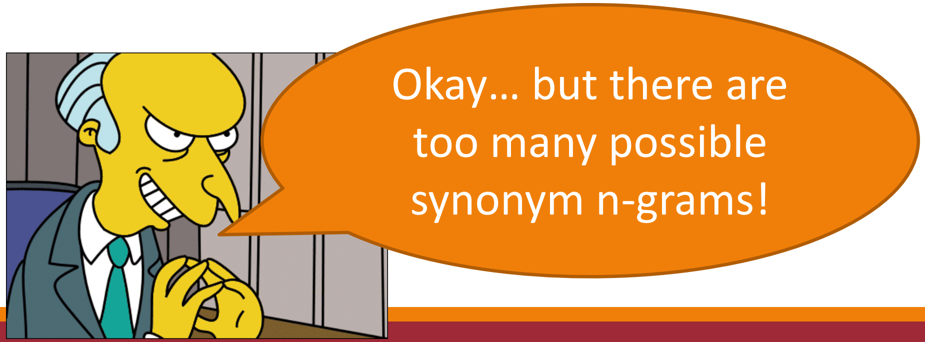


Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Case Study: Maxent Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

$$f_{\text{any}-\text{common}-\text{word},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<word v>},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{<ngram Z>},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{synonym}-\text{of}-\text{<word v>},\text{Plag.}}(x_1, x_2) = ???$$

$$f_{\text{synonym}-\text{of}-\text{<ngram Z>},\text{Plag.}}(x_1, x_2) = ???$$

Hah, I win!

Image: http://3.bp.blogspot.com/_MWJDzvrnu7Y/TTcXpCWZerI/AAAAAAAAAWY/zxu1GwrIBz0/s1600/excellent-mr-burns.gif

Plagiarism Detection: Word Similarity?

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

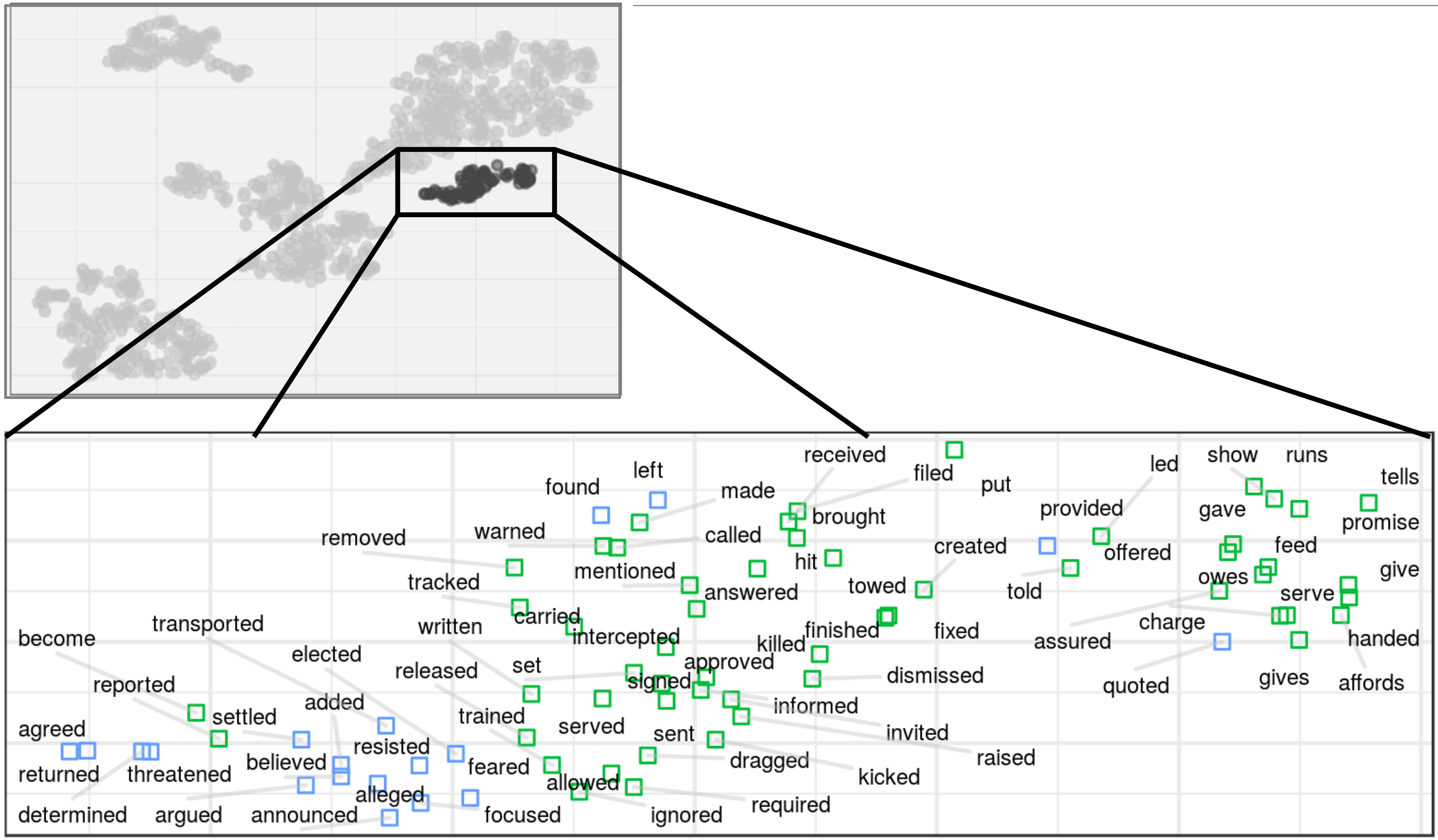
Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc.

A Dense Representation (E=2)



Distributional Representations

A dense, “low”-dimensional vector representation

Many values are not 0 (or at least less sparse than one-hot)

Up till ~2013: E could be any size
2013-present: E << vocab

An E-dimensional vector, often (but not always) real-valued

These are also called

- **embeddings**
- **Continuous representations**
- **(word/sentence/…) vectors**
- **Vector-space models**

Distributional models of meaning
= vector-space models of meaning
= vector semantics

Zellig Harris (1954):

- “oculist and eye-doctor … occur in almost the same environments”
- “If A and B have almost identical environments we say that they are synonyms.”

Firth (1957):

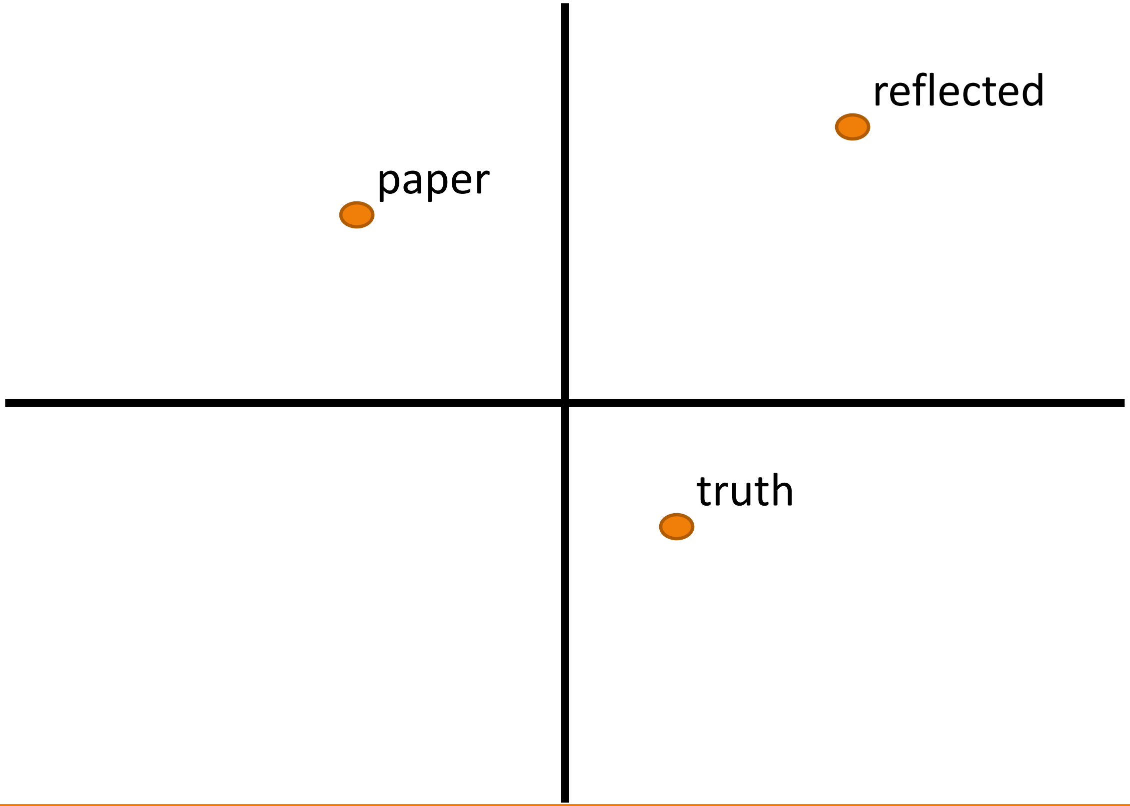
- “You shall know a word by the company it keeps!”

Continuous Meaning

The paper reflected the truth.

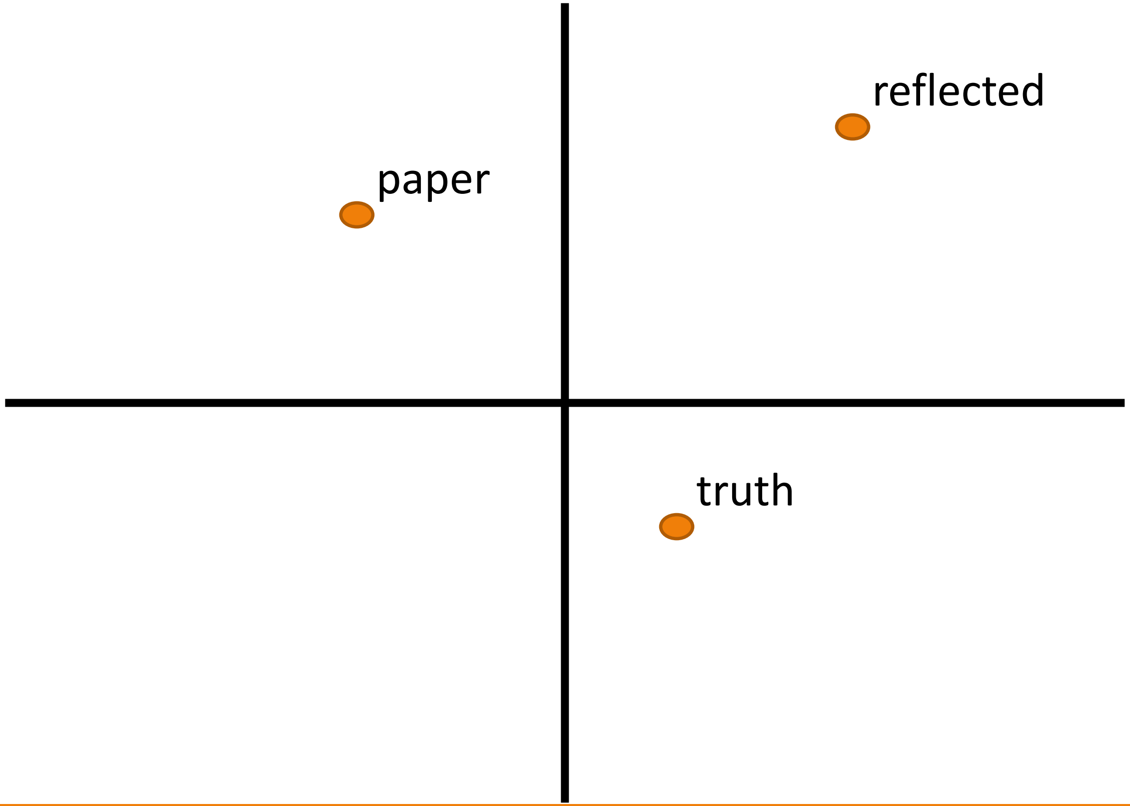
Continuous Meaning

The paper reflected the truth.



Continuous Meaning

The paper reflected the truth.



glean

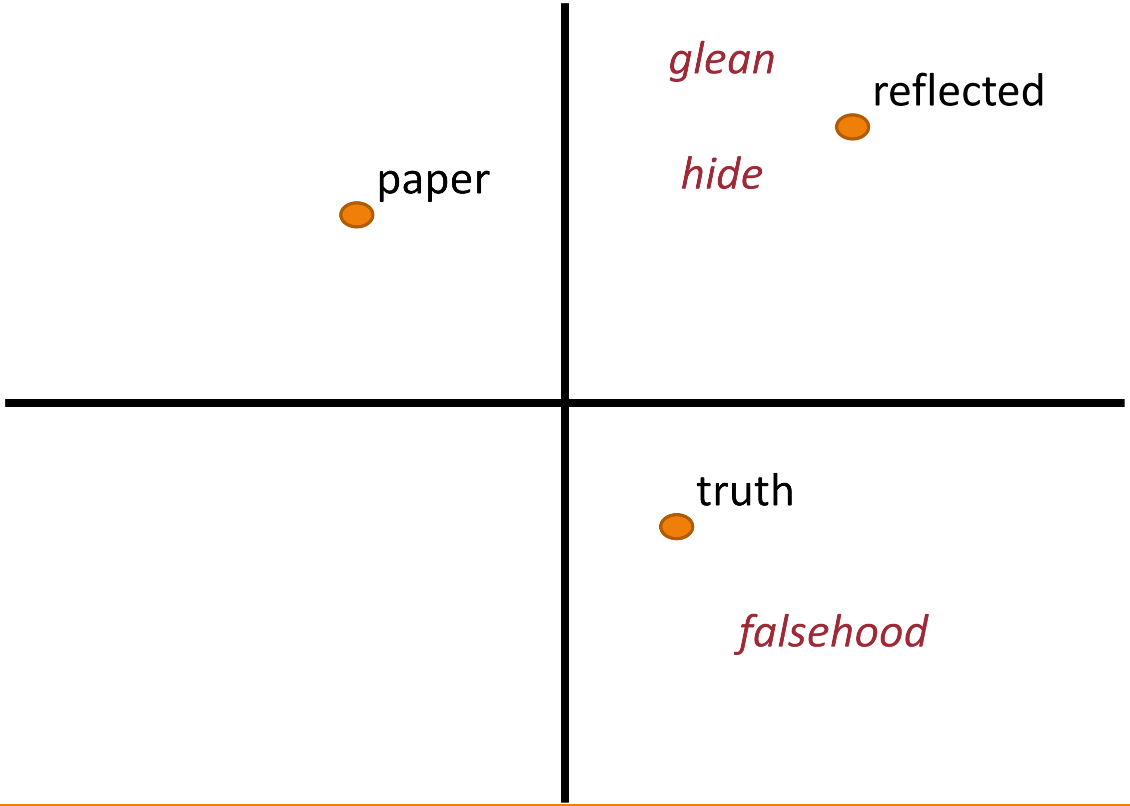
hide

falsehood

where might these go in this space?

Continuous Meaning

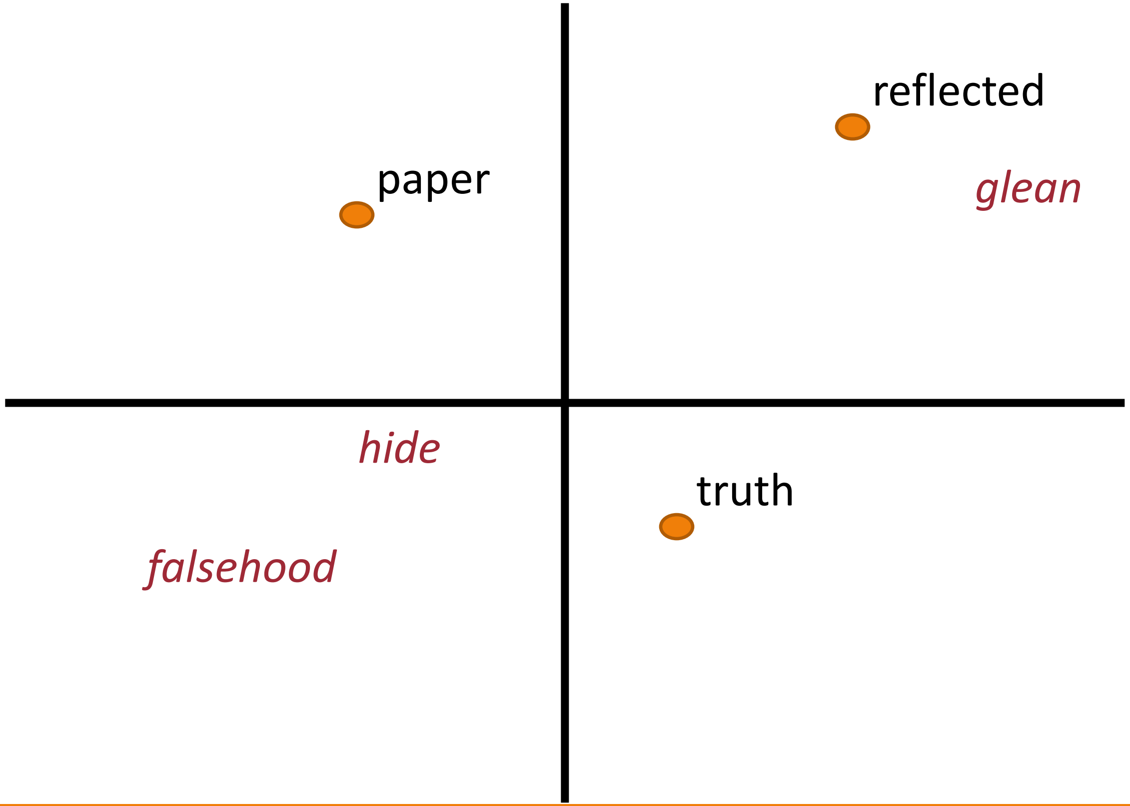
The paper reflected the truth.



One option

Continuous Meaning

The paper reflected the truth.



Another option

(Some) Properties of Embeddings

https://media3.giphy.com/media/3orif0M8U1E7NfpFzq/200_s.gif

Capture "like" (similar) words

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Mikolov et al. (2013)

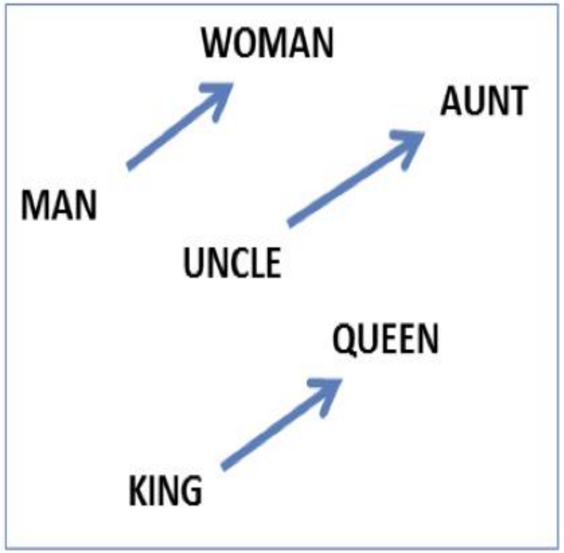
(Some) Properties of Embeddings

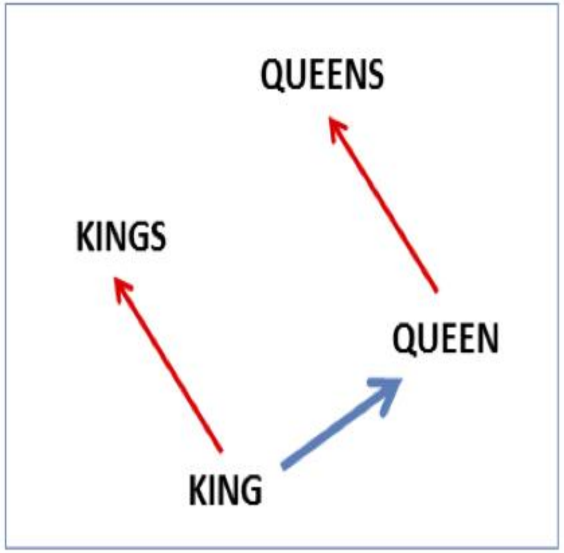
https://media3.giphy.com/media/3orif0M8U1E7NfpFzq/200_s.gif

Capture "like" (similar) words

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Capture relationships





vector(*'king'*) – vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')

Mikolov et al. (2013)

Case Study: Maxent Plagiarism Detector (Feature Example)

Given two documents x_1, x_2, predict y = 1 (plagiarized) or y = 0 (not plagiarized)

Intuition: documents are more likely to be plagiarized if they have words in common

☹

$\ldots\text{ommon}-\text{word,Plag.}(x_1, x_2) = ???$

$\ldots\text{rd v>,Plag.}(x_1, x_2) = ???$

$\ldots\text{gram Z>,Plag.}(x_1, x_2) = ???$

$f_{\text{synonym}-\text{of}-\text{<word v>,Plag.}}(x_1, x_2) = ???$

$f_{\text{synonym}-\text{of}-\text{<ngram Z>,Plag.}}(x_1, x_2) =$

get_similarity_with_embeddings()

Creating Vector Representations

“Embeddings” Did Not Begin In This Century…

Hinton (1986): “Learning Distributed Representations of Concepts”

Deerwester et al. (1990): “Indexing by Latent Semantic Analysis”

Brown et al. (1992): “Class-based n-gram models of natural language”

Key Ideas

1. Acquire basic contextual statistics (often counts) for each word type v

Key Ideas

1. Acquire basic contextual statistics (often counts) for each word type v
2. Extract a real-valued vector e_v for each word v from those statistics

Key Ideas

1. Acquire basic contextual statistics (often counts) for each word type v
2. Extract a real-valued vector e_v for each word v from those statistics
3. Use the vectors to represent each word in later tasks

Key Ideas:
Generalizing to linguistic “blobs”

1. Acquire basic contextual statistics (often counts) for each blob type v

2. Extract a real-valued vector e_v for each blob v from those statistics

3. Use the vectors to represent each blob in later tasks

Evaluating Vector Embeddings

Evaluating Similarity

Extrinsic (task-based, end-to-end) Evaluation:

- Question Answering
- Spell Checking
- Essay grading

Evaluating Similarity

Extrinsic (task-based, end-to-end) Evaluation:

- Question Answering
- Spell Checking
- Essay grading

Intrinsic Evaluation:

- Correlation between algorithm and human word similarity ratings
- Taking TOEFL multiple-choice vocabulary tests

Common Evaluation: Correlation between similarity ratings

Input: list of N word pairs $\{(x_1, y_1), \dots, (x_N, y_N)\}$

- Each word pair (x_i, y_i) has a human-provided similarity score h_i

Use your embeddings to compute an embedding similarity score $s_i = \mathrm{sim}(x_i, y_i)$

Compute the correlation between human and computed similarities

$$\rho = \mathrm{Corr}((h_1, \dots, h_N), (s_1, \dots, s_N))$$

Wordsim353: 353 noun pairs rated 0-10

Cosine: Measuring Similarity

Given 2 target words v and w how similar are their vectors?

Dot product or inner product from linear algebra

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- High when two vectors have large values in same dimensions, low for orthogonal vectors with zeros in complementary distribution

Correct for high magnitude vectors $\dfrac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$