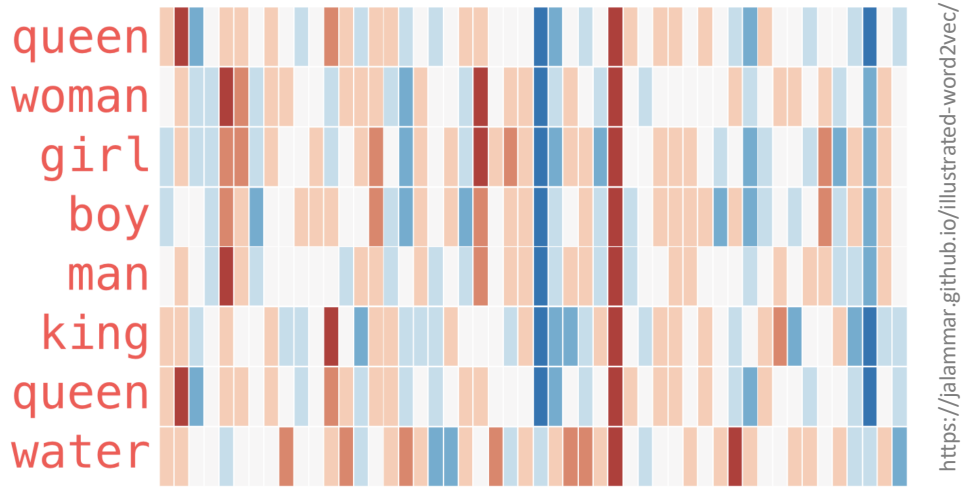
Natural Language Processing (NLP)

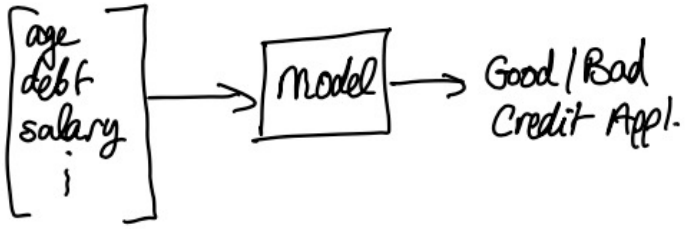


Slide credits:

- https://www.coursera.org/learn/nlp-sequence-models/home/week/2
- https://www.youtube.com/watch?v=yvMgcLKuvVg

Many good figs from this great blog:

- https://jalammar.github.io/illustrated-word2vec/



Numeric input
Fixed length

Numeric features – Fixed Length Input.

What happens when the input is text? ⟹ NATURAL LANGUAGE UNDERSTANDING/ PROCESSING (NLP)

"The movie was great" → Model → ★★★★★ (5-stars)

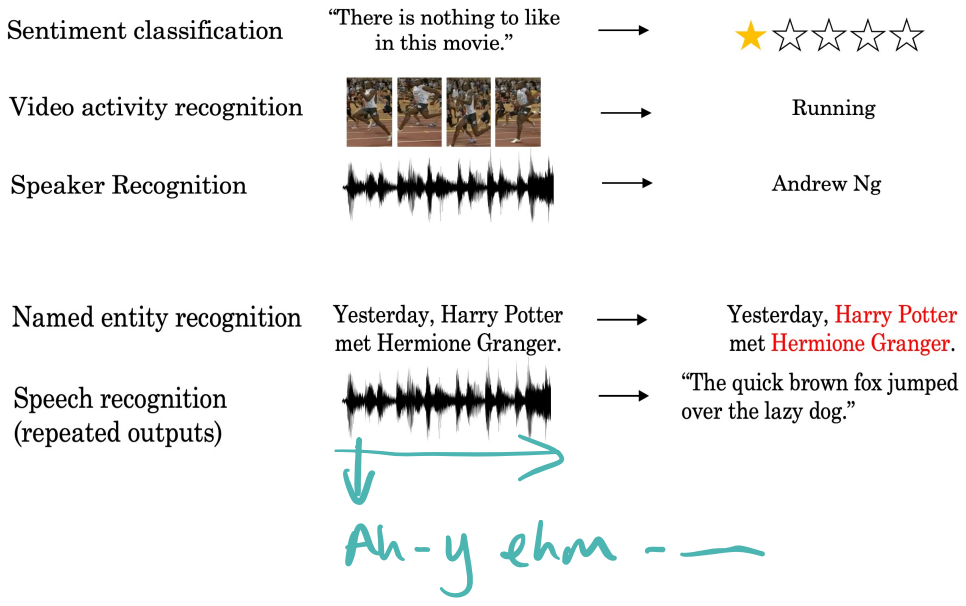
"The movie was great, but the acting was so so" → Model → ★★★

- sentiment analysis
- text summariz.
- Q & A

Text input
Variable length

Various NLP & Seq2Seq Tasks

- Sentiment Analysis (many-to-one)
- Video activity recognition (many-to-one)
- Speaker recognition
- Named Entity Recognition (many-to-many, aligned)
- Speech recognition (many-to-many, aligned)
- Machine translation (many-to-many, unaligned)



- **How do you represent words in NLP applications?**
- **How do we deal with the variable length sequence?**

Sentiment Analysis – A Simple Approach

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Sentiment Analysis – A Simple Approach

	Restaurant Reviews
R1	Great restaurant and great service !
R2	They can do better to provide better service
R3	Only two thumbs up, worst service ever

Entire Corpus

Counts of a term in a review

Set of all the words in the corpus	R1: Great Restaurant and great service !	R2: They can do better to provide better service	R3: Only two thumbs up, worst service ever
great	2	0	0
restaurant	1	0	0
and	1	0	0
service	1	1	0
they	0	1	0
can	0	1	0
do	0	1	0
better	0	2	0
to	0	1	0
provide	0	1	0
only	0	0	1
Two	0	0	1
thumbs	0	0	1
up	0	0	1
worst	0	0	1
ever	0	0	1

30,000-word lexicon for English is used for many NLP applications

Images from: https://towardsdatascience.com/text-data-representation-with-one-hot-encoding-tf-idf-count-vectors-co-occurrence-vectors-and-f1bccbd98bef

Sentiment Analysis with tf-idf vectors

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

R1: Great Restaurant and great service !	**R2:** They can do better to provide better service	**R3:** Only two thumbs up, worst service ever
1	0	0
1	0	0
1	0	0
1	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	1	0
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1

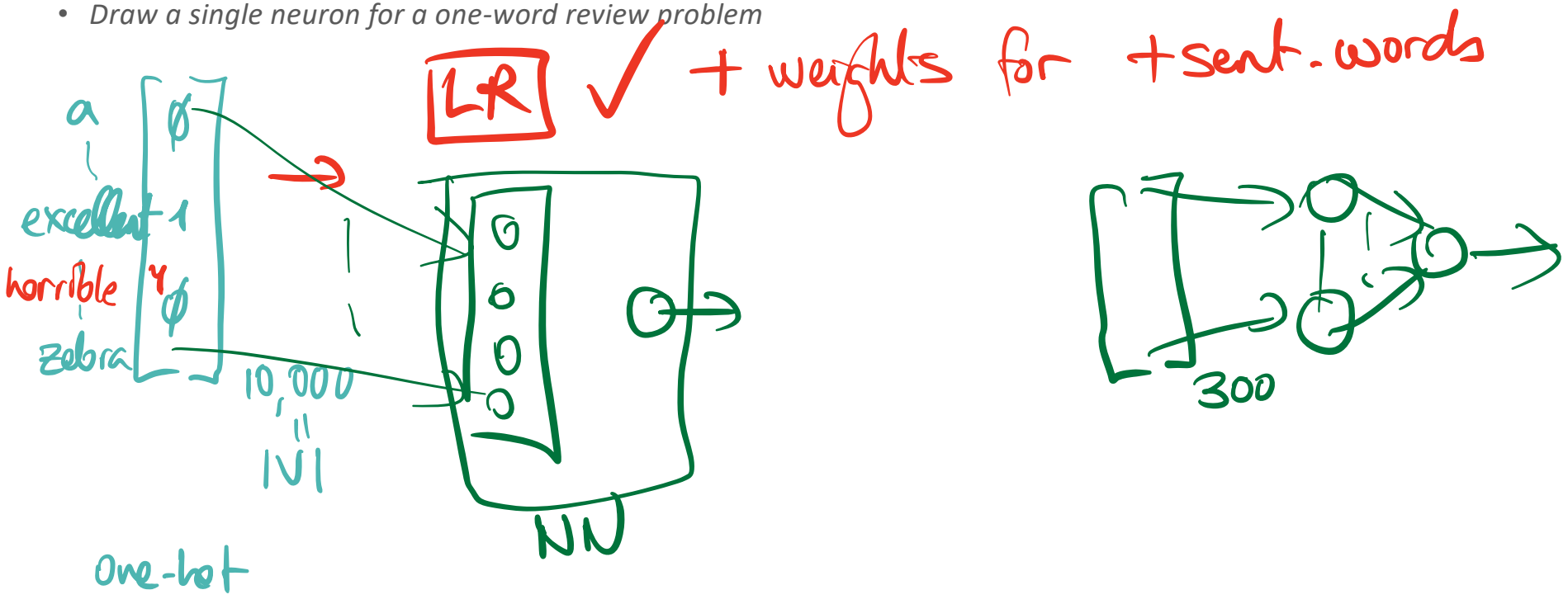
TF term (TF(t,d)) indicates the frequency of term t in document d

IDF indicates the inverse document frequency of the term t in the whole collection D

df(t) = number of documents term t appears / |D|

Term frequencies as features

- Train a model with this feature representation
- *Draw a single neuron for a one-word review problem*



One-Hot Encoding

"Hotel was great, a bit small but very comfortable"

a …	great	… hotel …. motel	zone

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

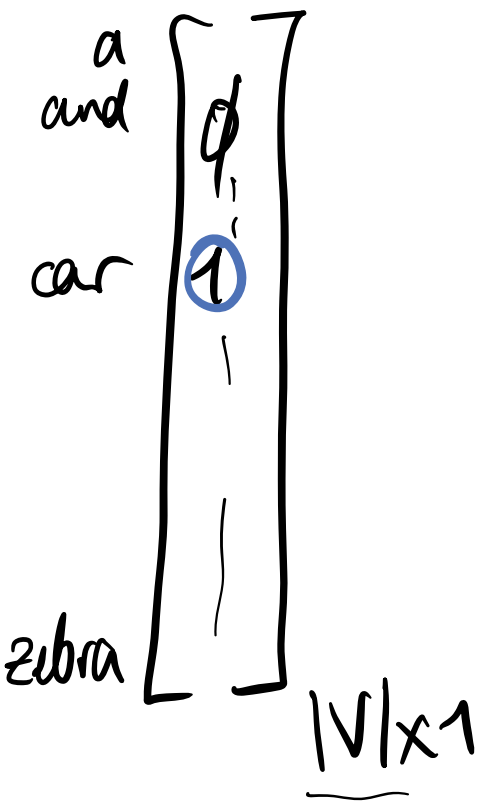
great =

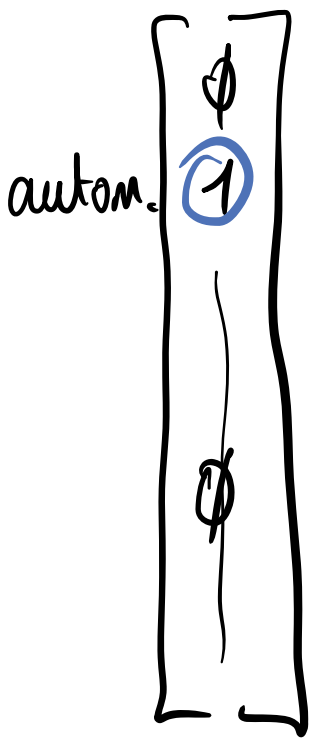
Major Shortcomings:

- Any two vectors are orthogonal
- No notion of similarity between similar words

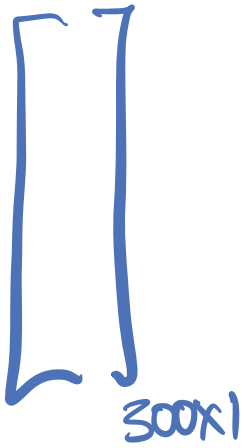
One-Hot Encoding

$V = \{a, \_\_, \_\_, zebra\}$





Distributed



Challenges

Variable-length: A text has variable length which makes it infeasible to use ML models that expect a fixed-length input

- Bag-of-words is a fixed-length vector

Large lexicon: Turkish has agglutinative morphology

- 1-million word lexicon
- Preprocessing: stemming etc.

gelebiliyor
gelebiliyorum
gelebiliyorsun
...

Word similarities are not captured

- "Great" is close to "Excellent"
- "Motel" is close to "Hotel"
- ....

Distributed Representation

- Can we map the meaning of a **word** or even a **sentence** to a vector?
- *Show example with Excellent, Great, So so, Horrible or King, Queen, Man, Woman…*

- Find vector representations that represent the meaning in text or images.
- Then we can take these meaning representations (embeddings) and use them as features in any ML model.

Distributed Representation

- Can we map the meaning in a word or even a sentence to a vector?

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

$$banking = \begin{bmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{bmatrix}$$

Note: word vectors are also called word embeddings or (neural) word representations
They are a distributed representation

From Manning, Stanford.

Word meaning as a neural word vector – visualization

$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

From R. Socher, Stanford.

Distributed Representations

But…

- The dimensions are not interpretable
- A given aspect of meaning may be distributed over a combination of many dimensions
- A given dimension may contribute to capturing several aspects of meaning

One-hot vs Distributional Representations

- Local Representation (One-hot Representation)
  - Entities as discrete symbols
  - Interaction between entities as discrete relations

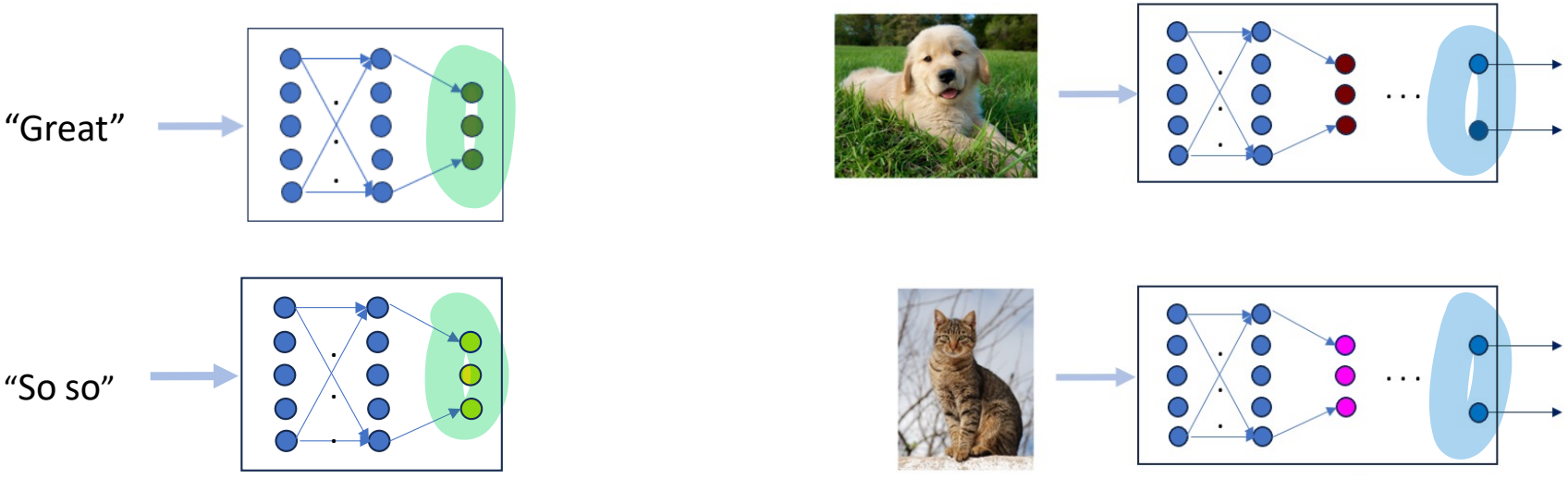
- Distributed Representations [Hinton et al., 1987]
  - Entity as a **vector of values**
  - **Meaning and relation to other entities are captured** by these vectors
  - Words mapped to **shared low dimensional space**

Embeddings

Find vector representations that represent the meaning in text or images.

Can we map the meaning in a word or even a sentence to a vector?

- We will use deep learning approaches



One-hot vs Distributional Representations

- Local Representation (One-hot Representation)
  - Entities as discrete symbols
  - Interaction between entities as discrete relations

- Distributed Representations [Hinton et al., 1987]
  - Entity as a **vector of values**
  - **Meaning and relation to other entities are captured** by these vectors
  - Words mapped to **shared low dimensional space**
  - The dimensions are not interpretable
    - A given aspect of meaning may be distributed over a combination of many dimensions
    - A given dimension may contribute to capturing several aspects of meaning

➢Word2Vec … and now transformer based word embeddings

Examples of Learned Relationships

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

What can you do with word embeddings?

- *Show the toy NN with word embeddings*

How to learn Word Embeddings

Distributional semantics

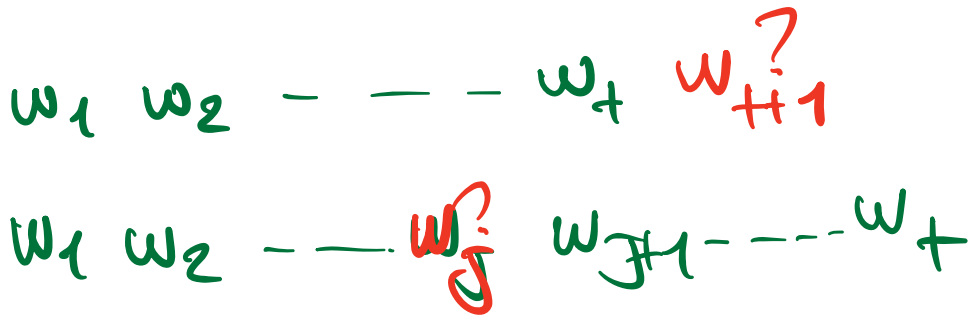
First observation:

- Distributional semantics: **A word's meaning is given by words that appear close by.**

"A red sports …… was speeding through the intersection. "

Corollary (implication):

➢Words in similar contexts should have similar embeddings (e.g. car and automobile)



Language Modelling !
- unigrams
- bigrams

$P(w_{t+1} \mid q_t =$)

P(to | "I am going")

Language Model

- A language model is a probability distribution over sequences of words.

$$P(w_1, w_2, \ldots, w_T)$$

- Given any sequence of words of length T, a language model assigns a probability P to the whole sequence.

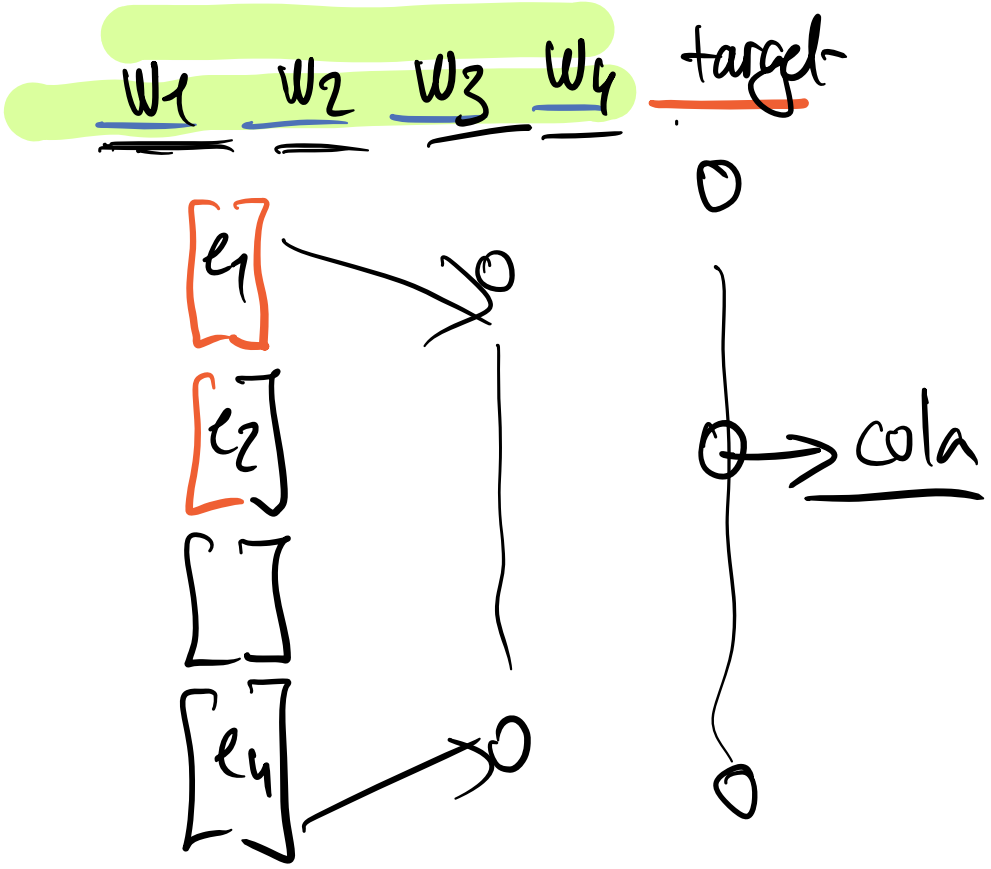
- Equivalently, find $P(w_T | w_1, \ldots, w_{T-1})$ – What is the next word?

Thou shalt not ……

- ***kill / steal / commit ...***

Bengio et al. 2003 - A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003.

- Learn word embeddings while learning a language model
- First algorithm to learn word embeddings



$P(w_t | w_{t-1}, w_{t-2}, \ldots)$

- unigram
- bigram
- trigram

Mikolov et al. 2013

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

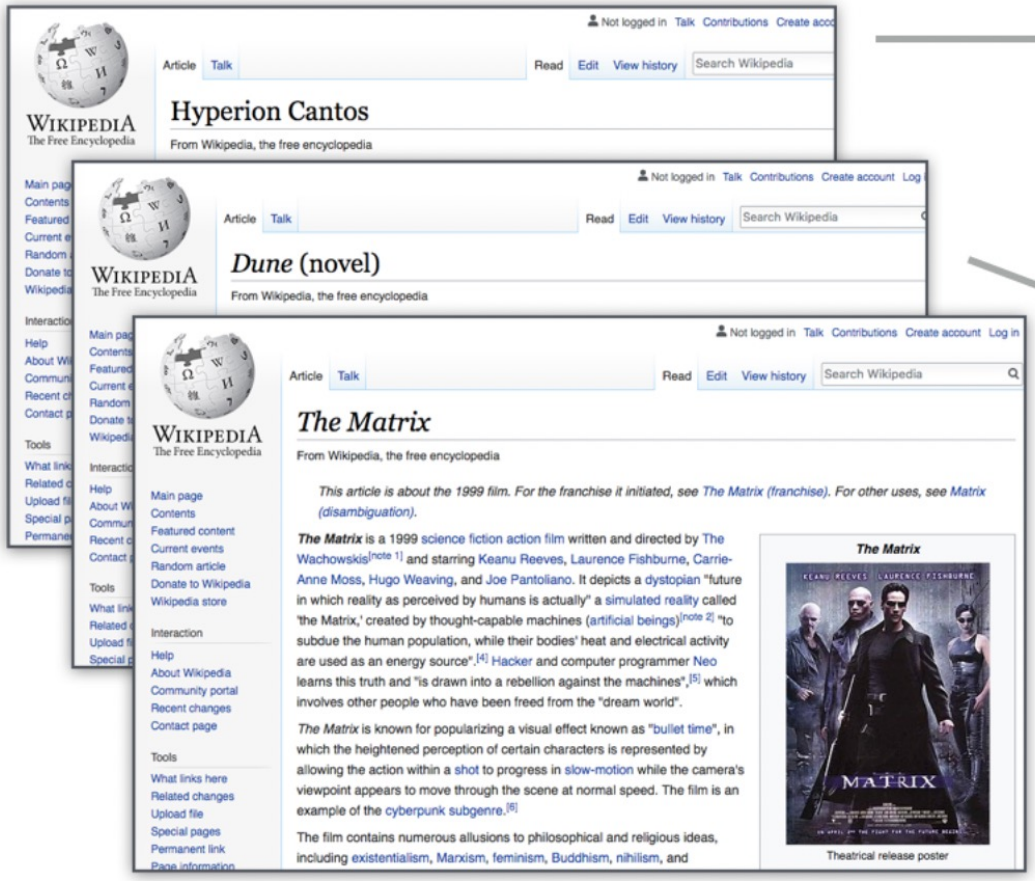
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

7 Sep 2013

- Word2Vec -

Large corpus



The **Hyperion Cantos** is a series of science fiction novels by Dan Simmons. The title ... in the series, *Hyperion* and *The Fall of Hyperion*,[1][2] and later came to refer to the overall storyline, including *Endymion*, *The Rise of Endymion*, and a number of short stories.[3][4] More narrowly, inside the fictional storyline, after the first volume, the Hyperion Cantos is an epic poem written by the character Martin Silenus covering in verse form the events of the first book.[5]

Of the four novels, *Hyperion* received the Hugo and Locus Awards in 1990;[6] *The Fall of Hyperion* won the Locus and British Science Fiction Association Awards in 1991;[7] and *The Rise of Endymion* received the Locus Award in 1998.[8] All four novels were also nominated for various science fiction awards.

An event series is being developed by Bradley Cooper, Graham King, and Todd Phillips for Syfy based on the first novel *Hyperion*.[9]

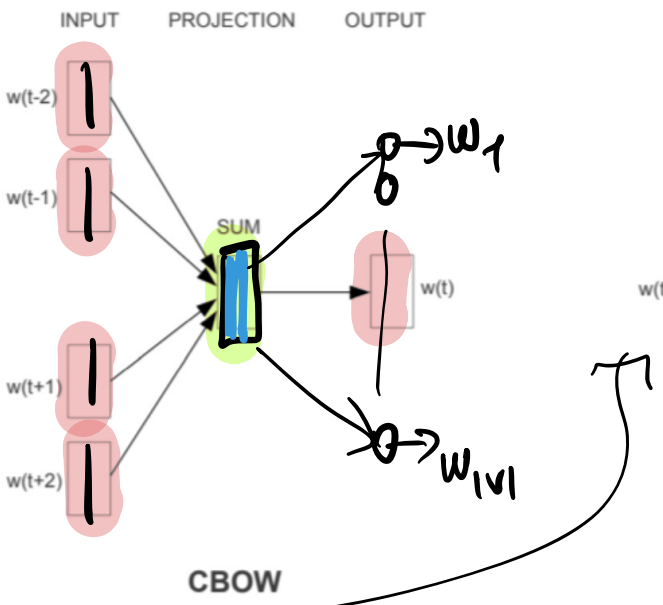
Dune is a 1965 science fiction novel by American author Frank Herbert, originally published as two separate serials in *Analog* magazine. It tied with Roger Zelazny's *This Immortal* for the Hugo Award in 1966,[3] and it won the inaugural Nebula Award for Best Novel.[4] It is the first installment of the *Dune* saga, and in 2003 was cited as the world's best-selling science fiction novel.[5][6]

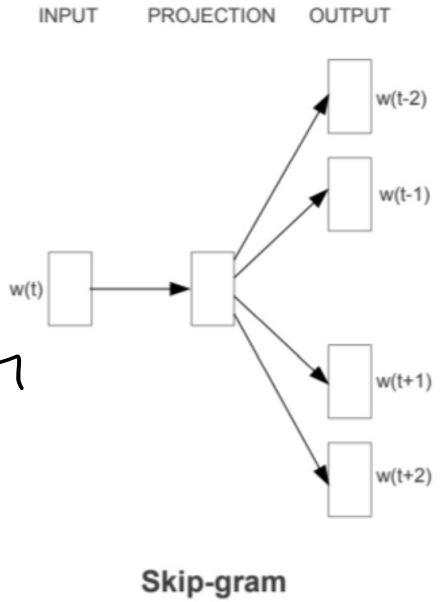
Set in the distant future amidst a feudal interstellar society in which noble houses, in control of individual planets, owe allegiance to the Padishah Emperor, *Dune* tells the story of young Paul Atreides, whose noble family accepts the stewardship of the planet Arrakis. It is an inhospitable and sparsely populated desert wasteland, but is also the only source of melange, also known as "spice", a drug that enhances mental abilities. As melange is the most important and valuable substance in the universe, control of Arrakis is a coveted—and dangerous—undertaking. The story explores the multi-layered interactions of politics, religion, ecology, technology, and human emotion, as the factions of the empire confront each other in a struggle for the control of Arrakis

The Matrix is a 1999 science fiction action film written and directed by The Wachowskis[note 1] and starring Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving, and Joe Pantoliano. It depicts a dystopian "future in which reality as perceived by humans is actually" a simulated reality called 'the Matrix,' created by thought-capable machines (artificial beings)[note 2] "to subdue the human population, while their bodies' heat and electrical activity

CBOW vs Skip-Gram

- The distributed representations (word embeddings) of context words are combined to **predict the word in the middle** (target word).
- The distributed representation of the input word is used to **predict the context**.





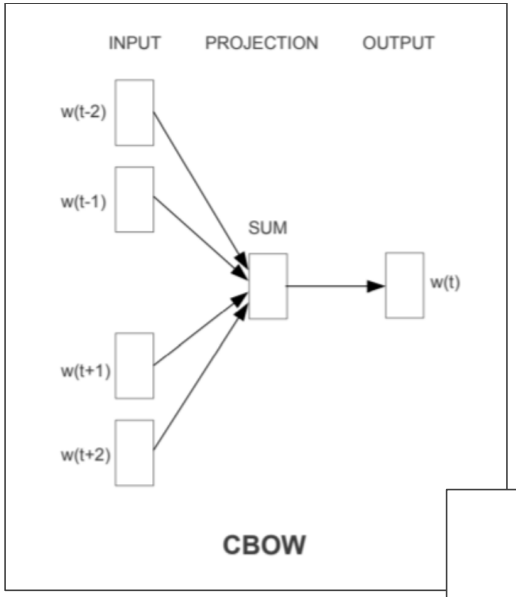
context to middle

middle to context

CBOW vs Skip-Gram

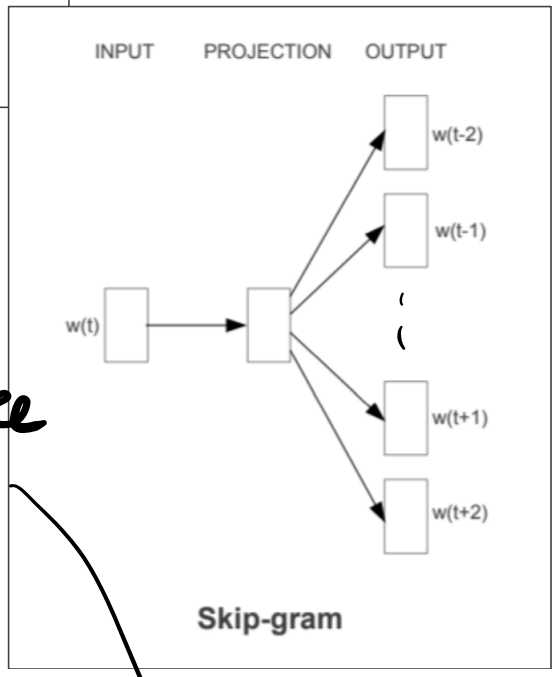
- CBOW: The distributed representations of context words are given to **predict the target word** (coffee).

  *"I like to have a cup of **coffee** at breakfast"*



- Skip-Gram: The distributed representation of the input word (cup) is used to **predict the context words**.

  *"I like to have a cup of **coffee** at breakfast"*



Skipgram Training set

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	…

input

context

input word	target word
not	thou
not	shalt
not	make
not	a

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	…

thou	shalt	not	make	a	machine	in	the	…

input word	target word
not	thou
not	shalt
not	make
not	a

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

Cbow – Continuous Bag of Words Training set

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	…

Dataset

input 1	input 2	output
thou	shalt	not

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	…
thou	shalt	not	make	a	machine	in	the	

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make
not	make	a
make	a	machine
a	machine	in

SkipGram - Learning Task

- If two different words *(e.g. car and automobile)* have similar "contexts",
  then our model needs to **output similar results** (posterior probabilities) **for these two words**.

- One way for the network to output similar predictions for these two words is **if the word vectors are similar.**

- We will predict the posterior probabilities of context words (called outside words in Manning slides) based on the **similarity (dot product) of its word embedding with that of the center word.**

Skip-gram

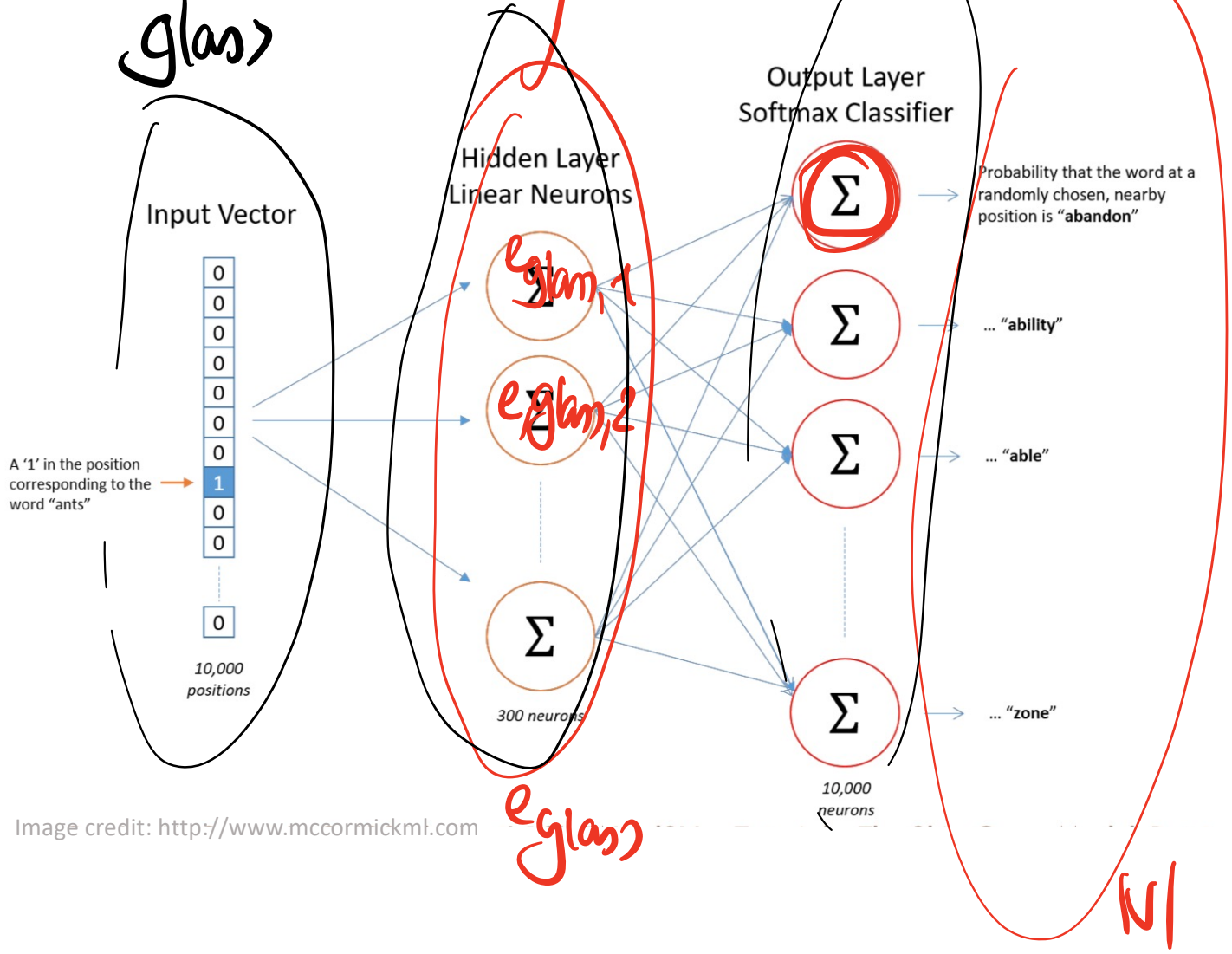


Image credit: http://www.mccormickml.com

e_{glass}

$e_{glass} \cdot \theta_{abandon}$

$e_{glass} \cdot \theta_{juice}$

$e_{glass} \cdot \theta_{apple}$

$e_{glass} \cdot \theta_{zebra}$

↓

Apply softmax!

$e_1 = \phi \cdot w_1 + \phi \cdot w_2 + \cdots + 1 \cdot w_{glass} + \phi \cdots$

Corpus: I want to drink ---

I want to drink a glass of orange juice .

o_{glass} e_{glass}

$W = E_{in}$ $\theta = E_{out}$

θ_{juice} : weights / embedding for word 'juice'

θ_{juice}

$p(a \mid glass)$

$p(aaron \mid glass)$

$p(juice \mid glass)$

$p(zulu \mid glass)$

$|V| = 19{,}000$ one-hot vector

$d = 300$ $|V| \times 300$ weights

w/ 3-dim.

[0.8 -0.2 0.6]

[0.7 -0.01 0.5]

Softmax activation

one hot

$\exp(\theta_{juice}^T \times e_{glass})$

$\sum_{w \in V} \exp(\theta_w^T \times e_{glass})$

e_{glass}

E

300 $|V|$ e

$300 \times |V| \times |V| \times 1 \Rightarrow 300$ dim. emb-

Chosen input = glass

= context = juice word

first dim. of e

o_{glass}

θ_{juice} juice

$|V|$

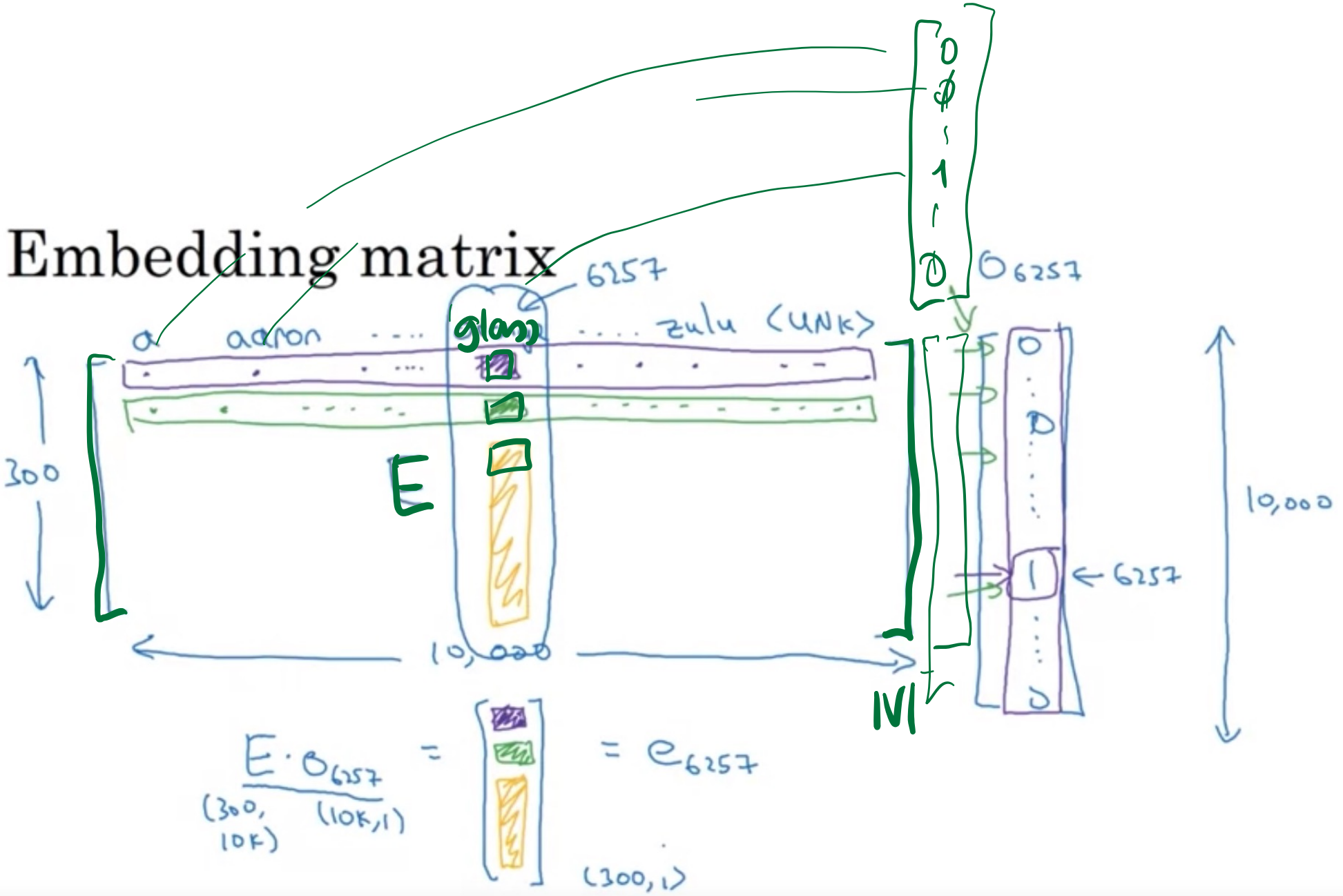
300

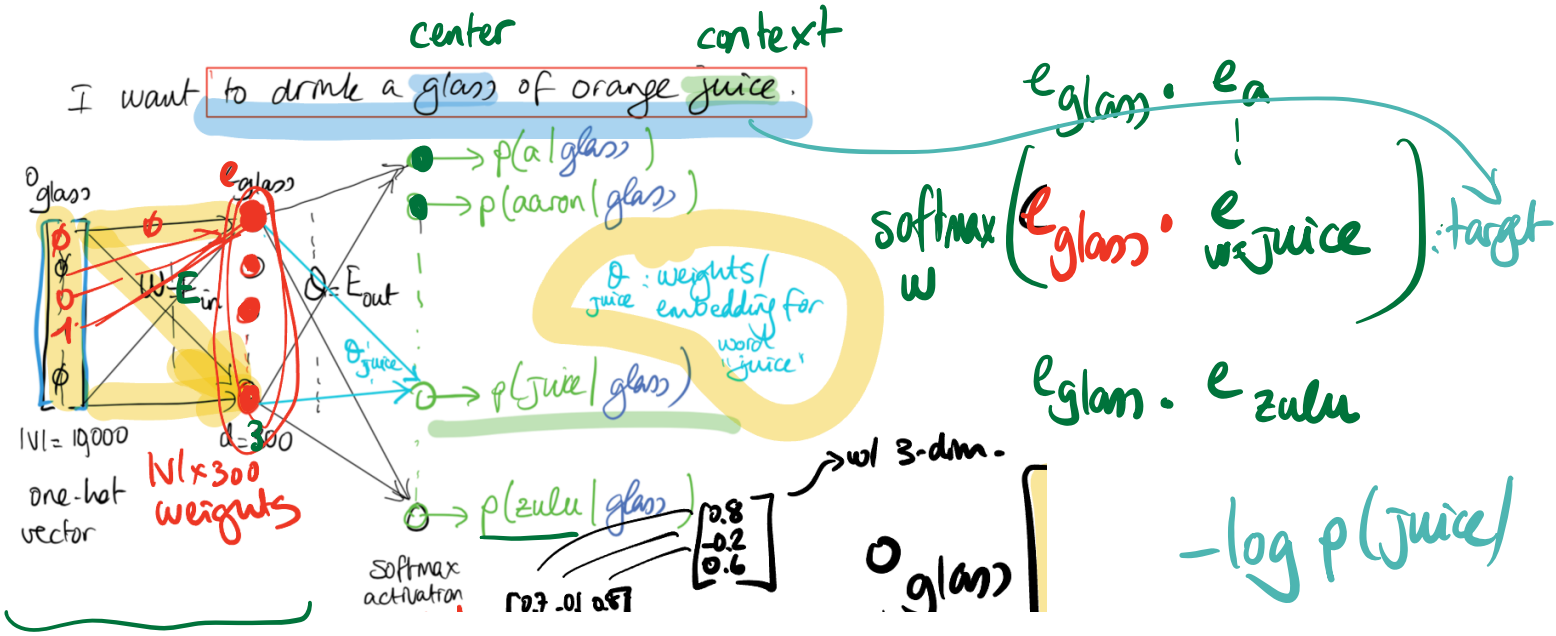
$|V|$

o_{glass}

e_{glass}

Embedding matrix





$o_{glass} = [0, \ldots, 1, \ldots, 0]$

e_{glass} extracted by $E \cdot o_{glass}$ → hidden activations

E: weight tensor of the hidden layer.

Word2vec: prediction function

② Exponentiation makes anything positive

① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

③ Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to (0,1)^n$ ← Open region

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

But sort of a weird name because it returns a distribution!

Shortcomings

- Huge number of weights in the embedding matrix and the output layer weights
  - |V|xD where |V| is the vocabulary size and D is the embedding dimensions.
  - 3M for |V|=10,000 D=3000

- Training with a large corpus, this is prohibitive.
  - ➢ Proposed suggestions in the next paper:
    - ➢ Hierarchical softmax
    - ➢ Negative sampling

➢ Fixed embeddings for each word – that does not depend on the context

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Negative Sampling

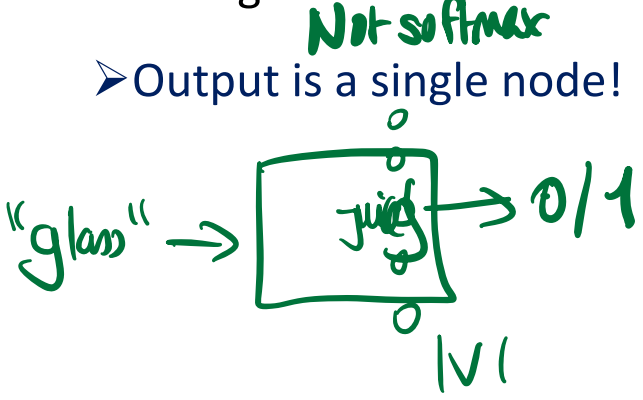
given
Task 1 : input word → predict context word
"glass" "juice"
out of all $w \in V$

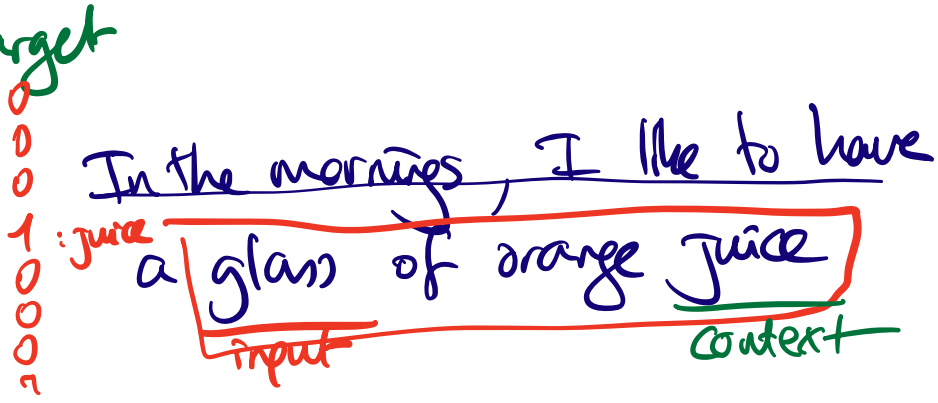
Negative Sampling Idea

Task 2 :

Instead of a |V|-way softmax which is costly, change the task to predicting "whether two words are in context of one another"

- So, for the context words around the center word, target will be 1
- Then pick a negative sample randomly from the dictionary (V) and set its target to 0 as it is not in context (at least not now).
- But instead of using all words in the dictionary (except for context) as negative samples, it randomly selects a handful ([2-20])of words depending on the training size, as negative sample.
- Output is a single node!





dataset
glass juice 1
glass car ∅
glass sleep ∅
glass orange 1