LANGUAGE MODELING AND WORD REPRESENTATION

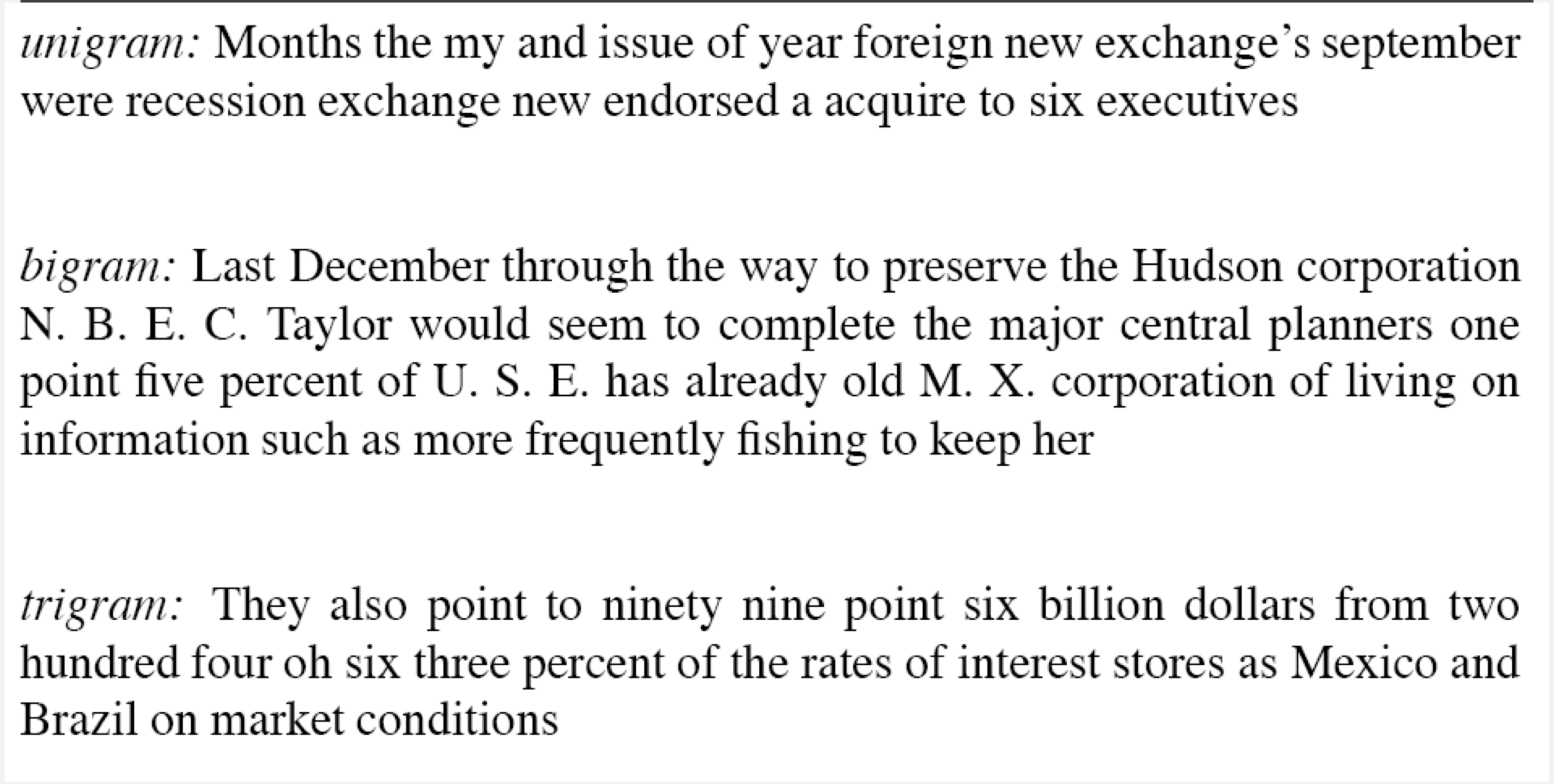
Manish Shrivastava

Lecture #3

N-GRAM TRAINING SENSITIVITY

- If we repeated the Shakespeare experiment but trained our n-grams on a Wall Street Journal corpus, what would we get?
- Note: ***This question has major implications for corpus selection or design***

WSJ IS ***NOT*** SHAKESPEARE: SENTENCES GENERATED FROM WSJ



unigram: Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

bigram: Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

trigram: They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

EVALUATION AND DATA SPARSITY QUESTIONS

- Perplexity and entropy: how do you ***estimate*** how well your language model fits a corpus once you' re done?
- Smoothing and Backoff : how do you handle unseen n-grams?

PERPLEXITY AND ENTROPY

- Information theoretic metrics
  - Useful in measuring how well a grammar or language model (LM) models a natural language or a corpus
- Entropy: With 2 LMs and a corpus, which LM is the better match for the corpus? How much information is there (in e.g. a grammar or LM) about what the next word will be? More is better!
  - For a random variable ***X*** ranging over e.g. bigrams and a probability function ***p(x),*** the entropy of X is the expected negative log probability

$$H(X) = -\sum_{x=1}^{x=n} p(x) \log_2 p(x)$$

- Entropy is the lower bound on the # of bits it takes to encode information e.g. about bigram likelihood
- Cross Entropy
  - An upper bound on entropy derived from estimating true entropy by a subset of possible strings – we don’ t know the real probability distribution
- Perplexity

$$PP(W) = 2^{H(W)}$$

  - At each choice point in a grammar
    - What are the average number of choices that can be made, weighted by their probabilities of occurrence?
    - i.e., Weighted average branching factor
  - How much probability does a grammar or language model (LM) assign to the sentences of a corpus, compared to another LM? The more information, the lower perplexity

SOME USEFUL OBSERVATIONS

- There are 884,647 tokens, with 29,066 word form types, in an approximately one million word Shakespeare corpus
  - Shakespeare produced 300,000 bigram types out of 844 million possible bigrams: so, ***99.96% of the possible bigrams were never seen (have zero entries in the table)***
- A small number of events occur with high frequency
- A large number of events occur with low frequency
- You can quickly collect statistics on the high frequency events
- You might have to wait an arbitrarily long time to get valid statistics on low frequency events
- Some zeroes in the table are really zeros But others are simply low frequency events you haven't seen yet. How to address?

George Kingsley Zipf
1902-1950

ZIPF' S LAW

- Frequency of occurrence of words is inversely proportional to the rank in this frequency of occurrence.
- When both are plotted on a log scale, the graph is a straight line.

ZIPF DISTRIBUTION

- The Important Points:
  - a few elements occur *very frequently*
  - a medium number of elements have medium frequency
  - many elements occur *very infrequently*

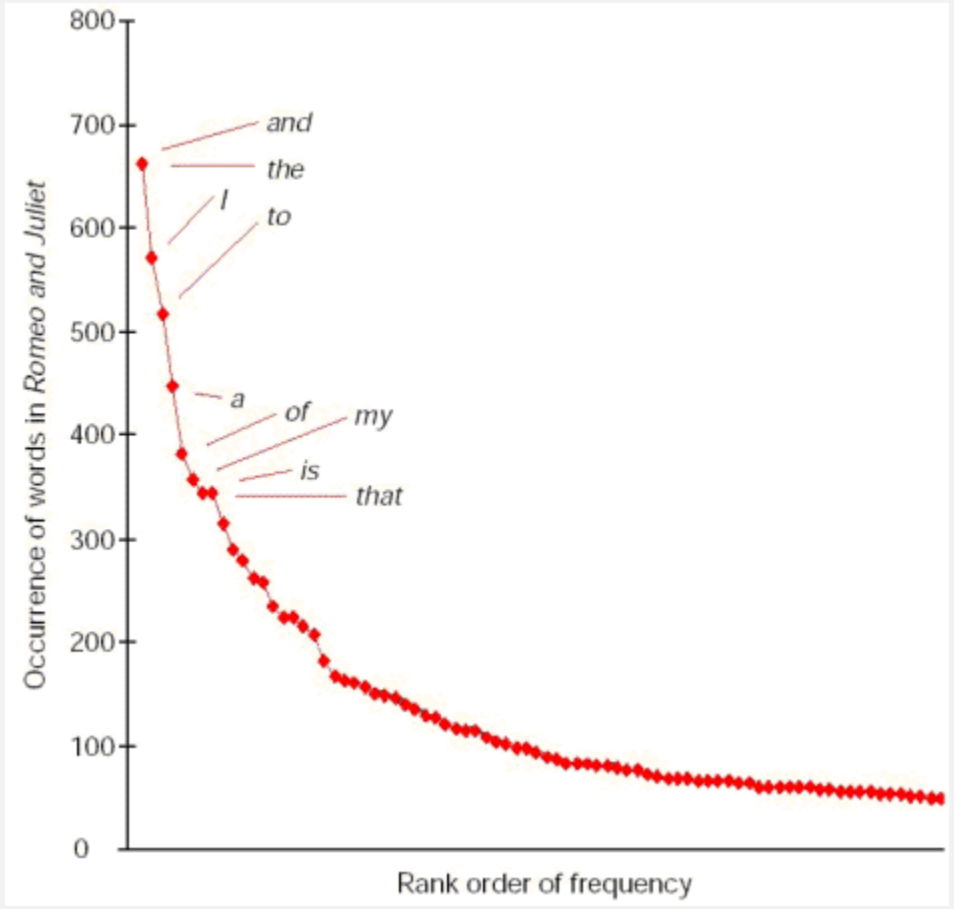
ZIPF DISTRIBUTION

The product of the frequency of words (f) and their rank (r) is approximately constant

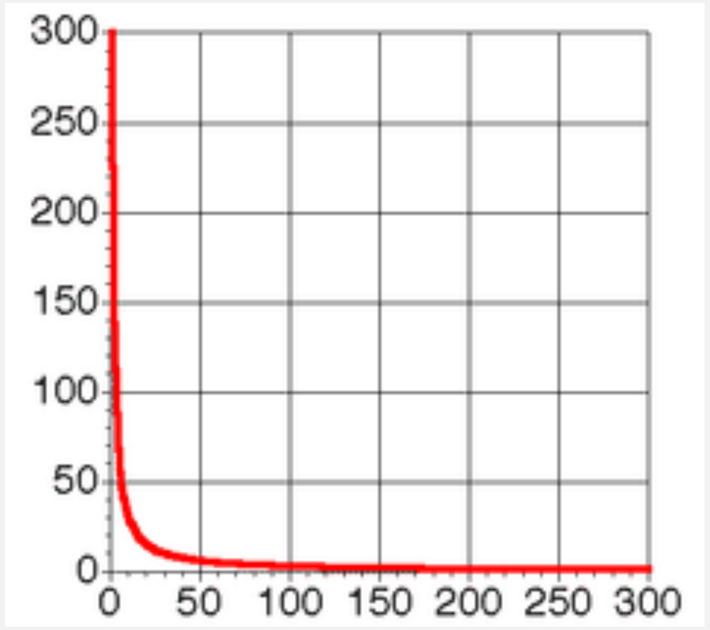
Rank = order of words' frequency of occurrence

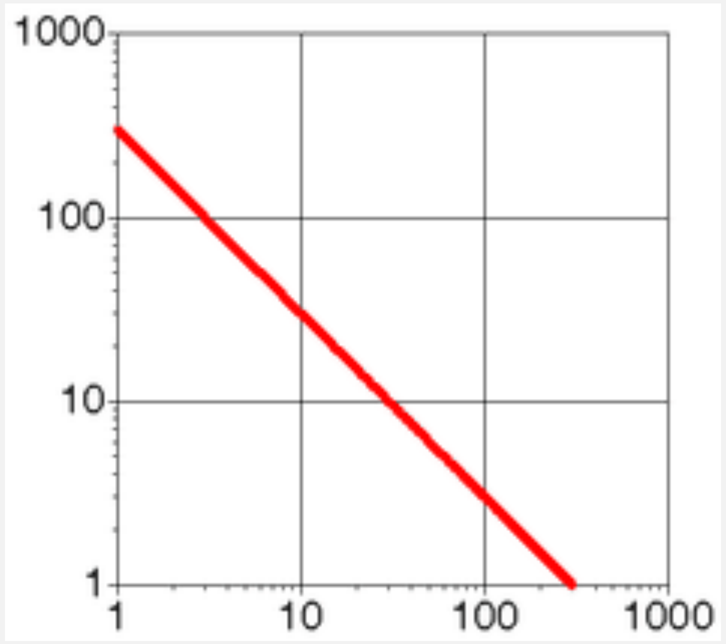
$$f = C * 1/r$$

$$C \cong N/10$$



ZIPF DISTRIBUTION (SAME CURVE ON LINEAR AND LOG SCALE)

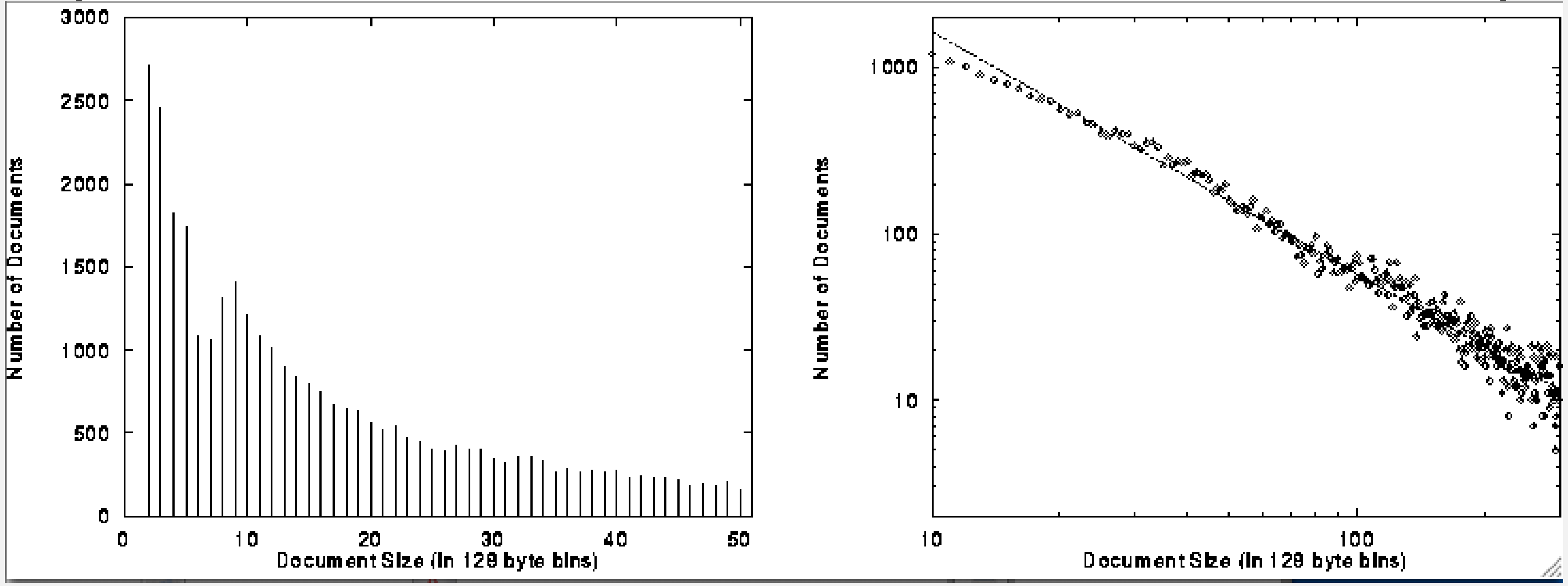




WHAT KINDS OF DATA EXHIBIT A ZIPF DISTRIBUTION?

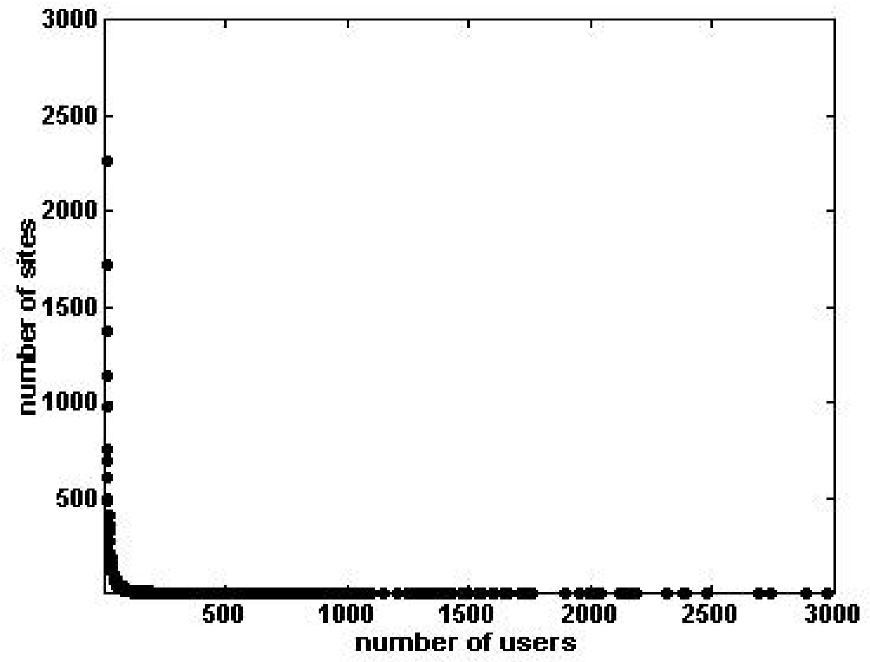
- Words in a text collection
  - Virtually any language usage
- Library book checkout patterns
- Incoming Web Page Requests (Nielsen)
- Outgoing Web Page Requests (Cunha & Crovella)
- Document Size on Web (Cunha & Crovella)

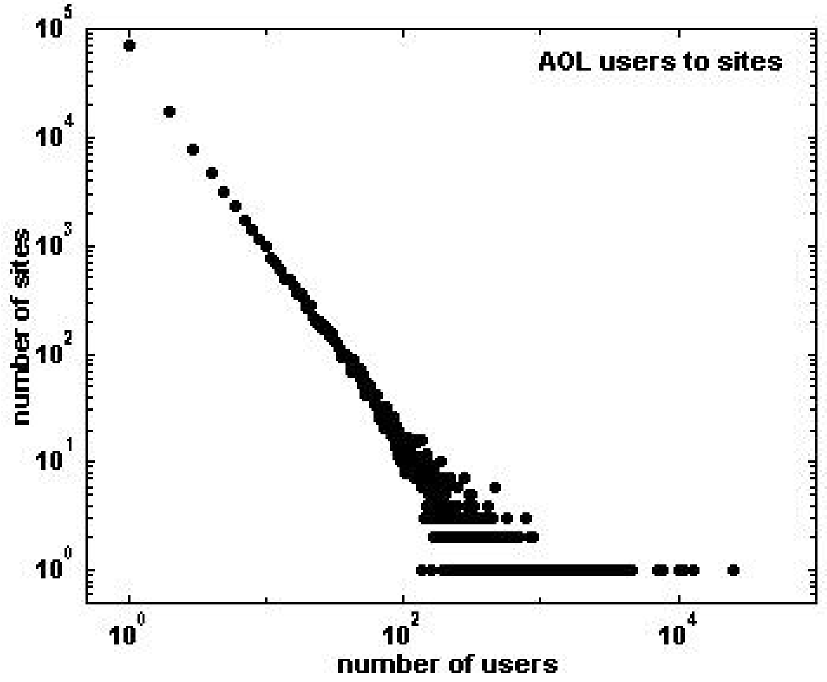
CHARACTERISTICS OF WWW CLIENT-BASED TRACES



Zipf's Law Applied To WWW Documents

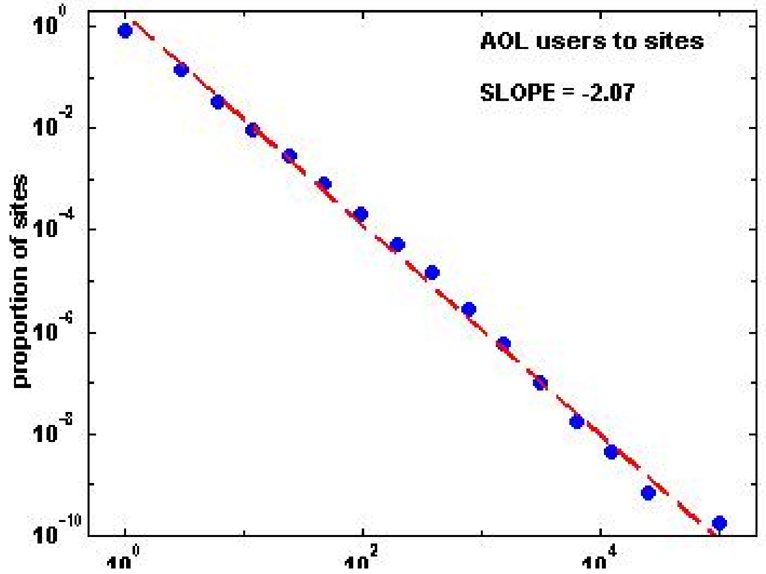
DISTRIBUTION OF USERS AMONG WEB SITES



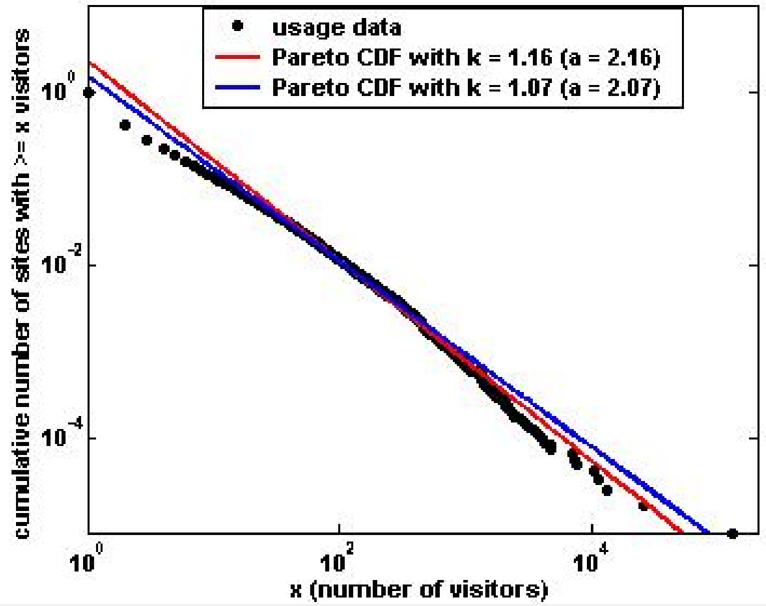


Binned distribution of users to sites

Exponentially increasing bins

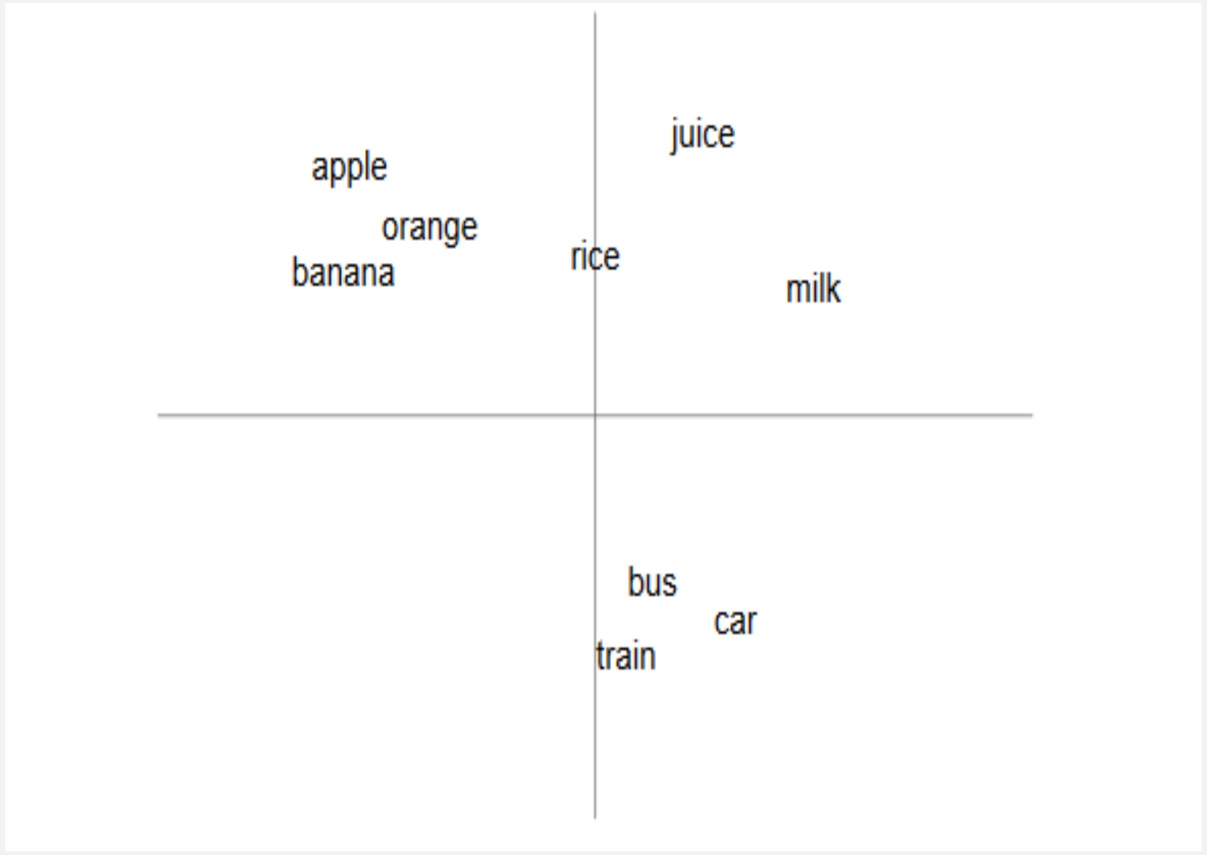


Cumulative distribution of users to sites

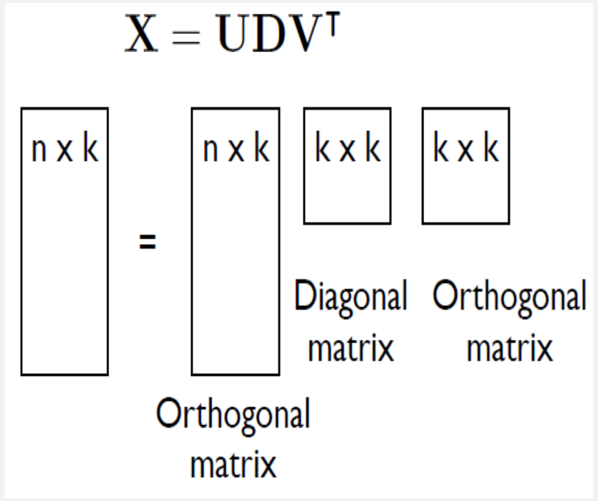


“You shall know a word by the company it keeps!”

Firth (1957)



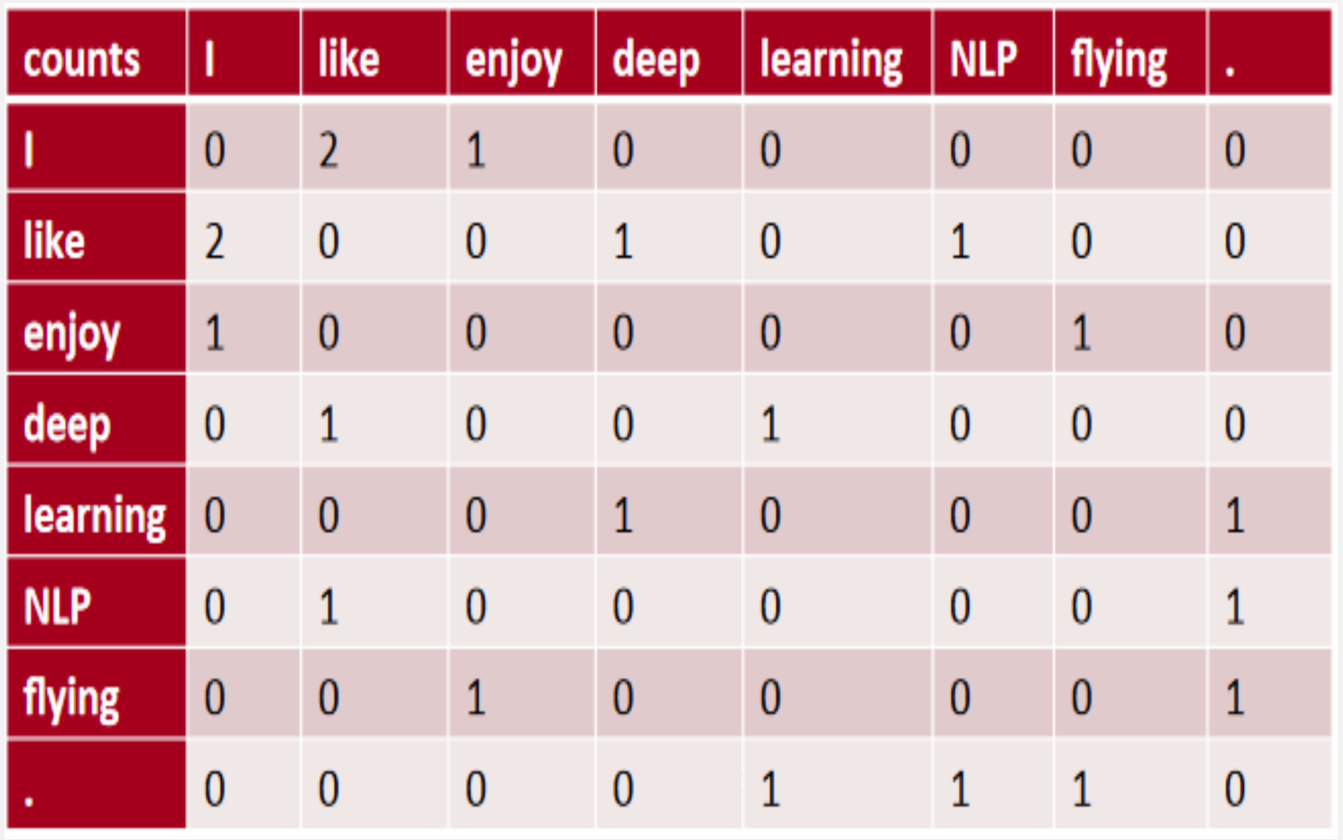
CO-OCCURRENCE MATRIX WITH SINGULAR VALUE DECOMPOSITION



BUILDING A CO-OCCURRENCE MATRIX

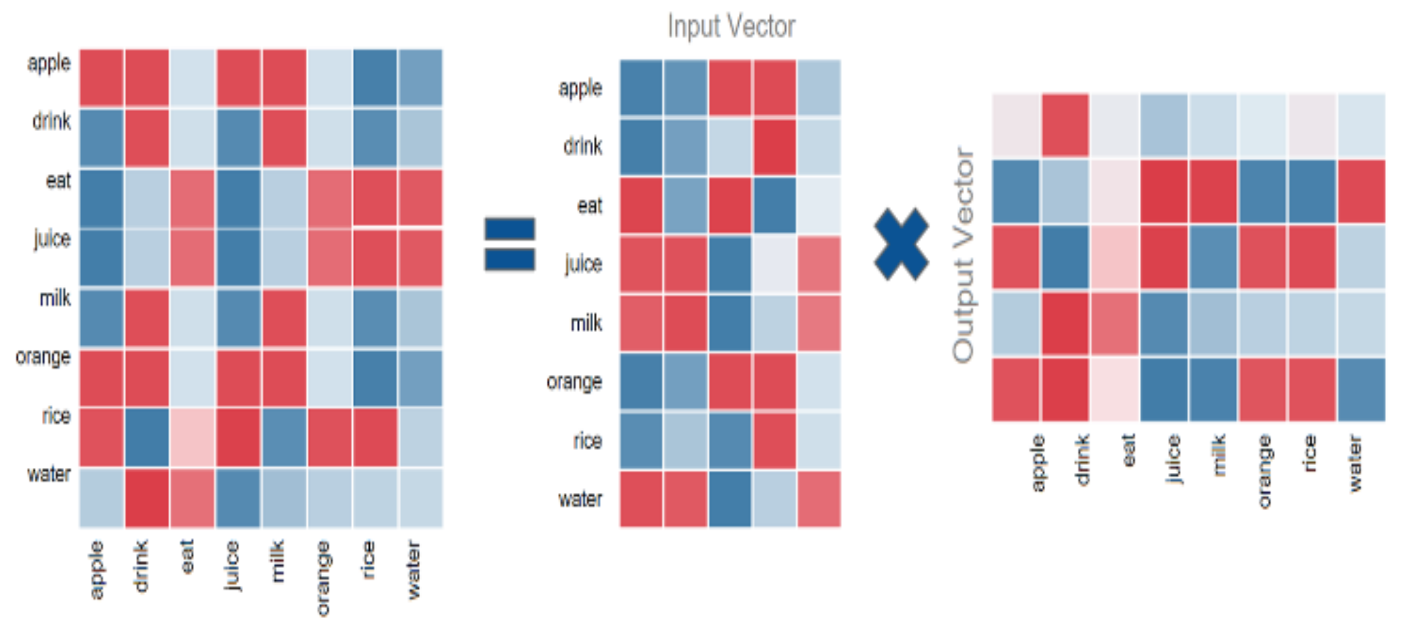
Corpus = {"I like deep learning"
"I like NLP"
"I enjoy flying"}

Context = previous word and next word

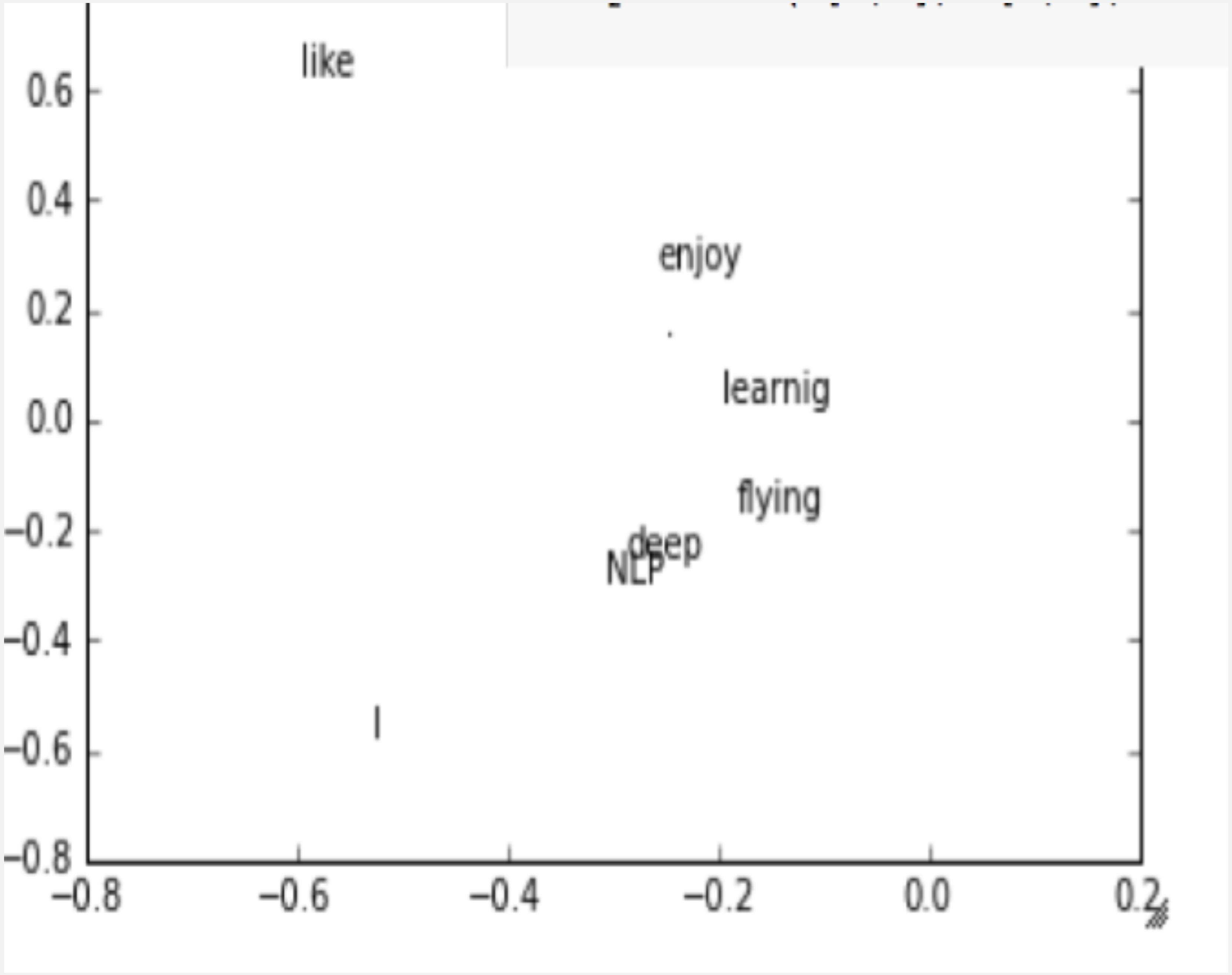


counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

SINGULAR VALUE DECOMPOSITION



The problem with this method, is that we may end up with matrices having billions of rows and columns, which makes SVD computationally restrictive.

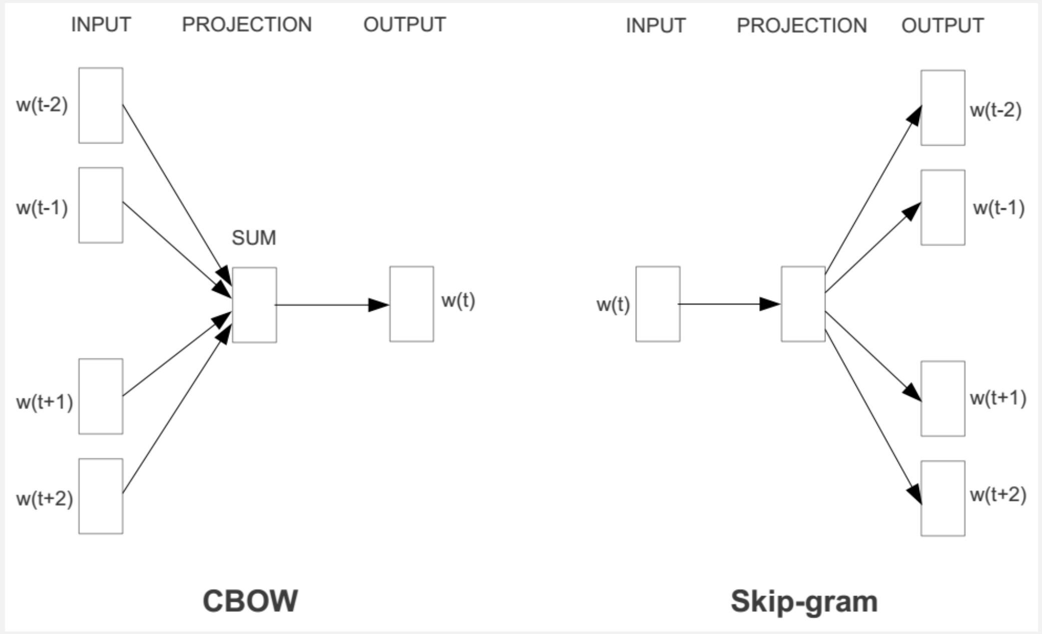

like
enjoy
learnig
flying
deep
NLP
I
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.8
−0.6
−0.4
−0.2
0.0
0.2

WORD REPRESENTATIONS

Traditional Method - Bag of Words Model	Word Embeddings
• Uses one hot encoding • Each word in the vocabulary is represented by neighbouring words • For example, if we have a vocabulary of 10000 words, and "Hello" is the 4th word in the dictionary, it would be represented by: 12 3 1 0 0 4 1 50 … 0 • Each number (in this instance) is the frequency of surrounding words • Dimensions are words.	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300) • Unsupervised, built just by reading huge corpus • For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02] • Dimensions are basically projections along different axes, more of a mathematical concept.

REPRESENT THE MEANING OF WORD **WORD2VEC**

- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.



WORD2VEC – CONTINUOUS BAG OF WORD

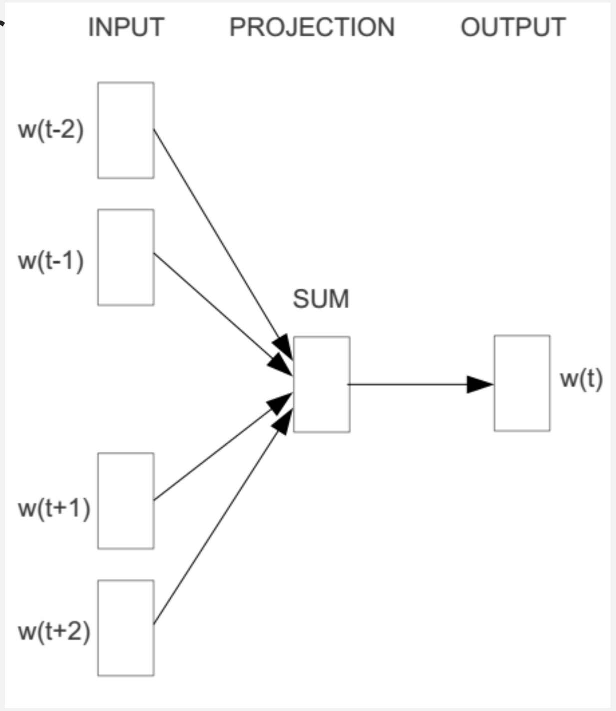
- E.g."The cat sat on floor
  - Window size = 2

the

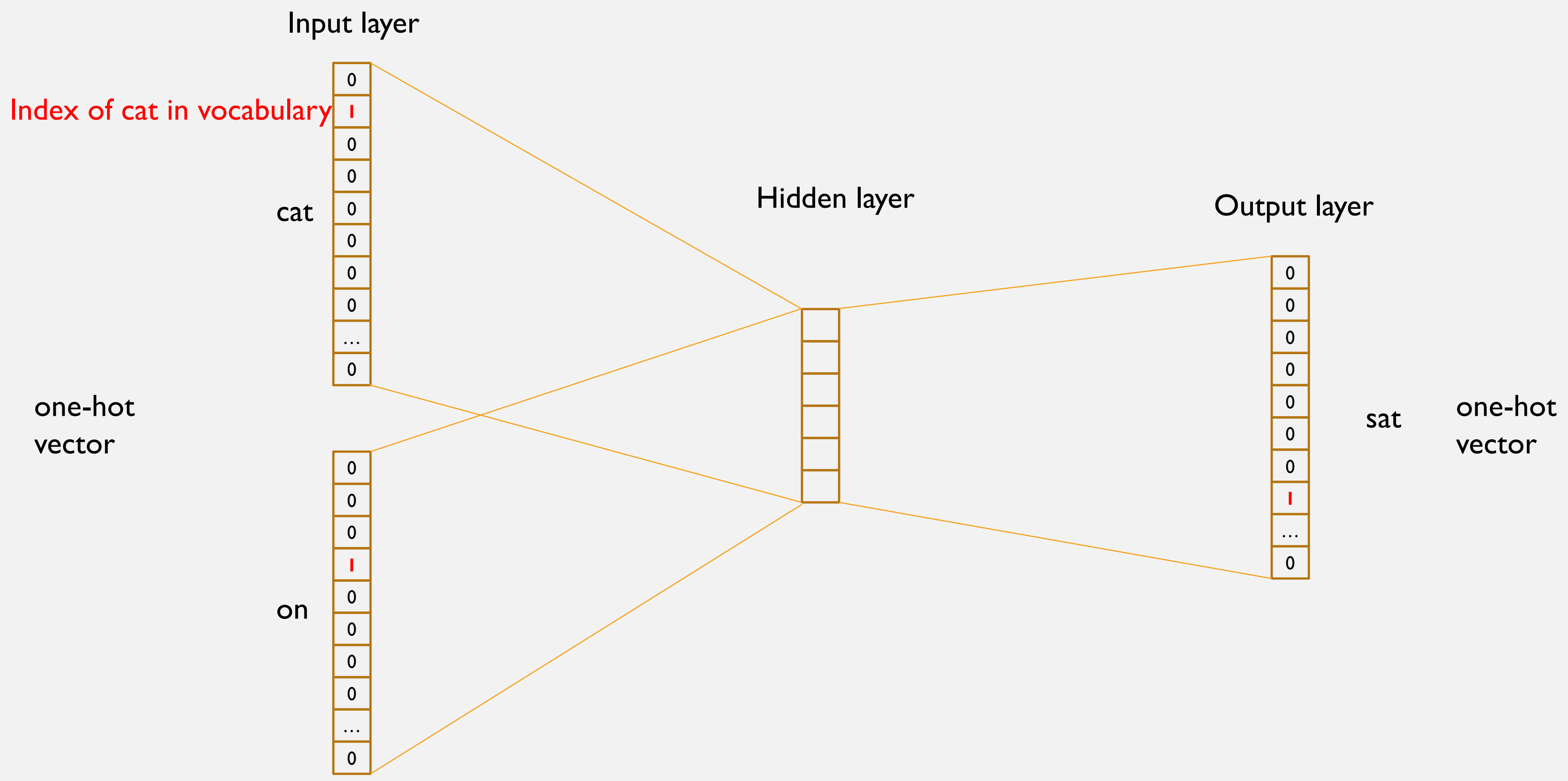
cat

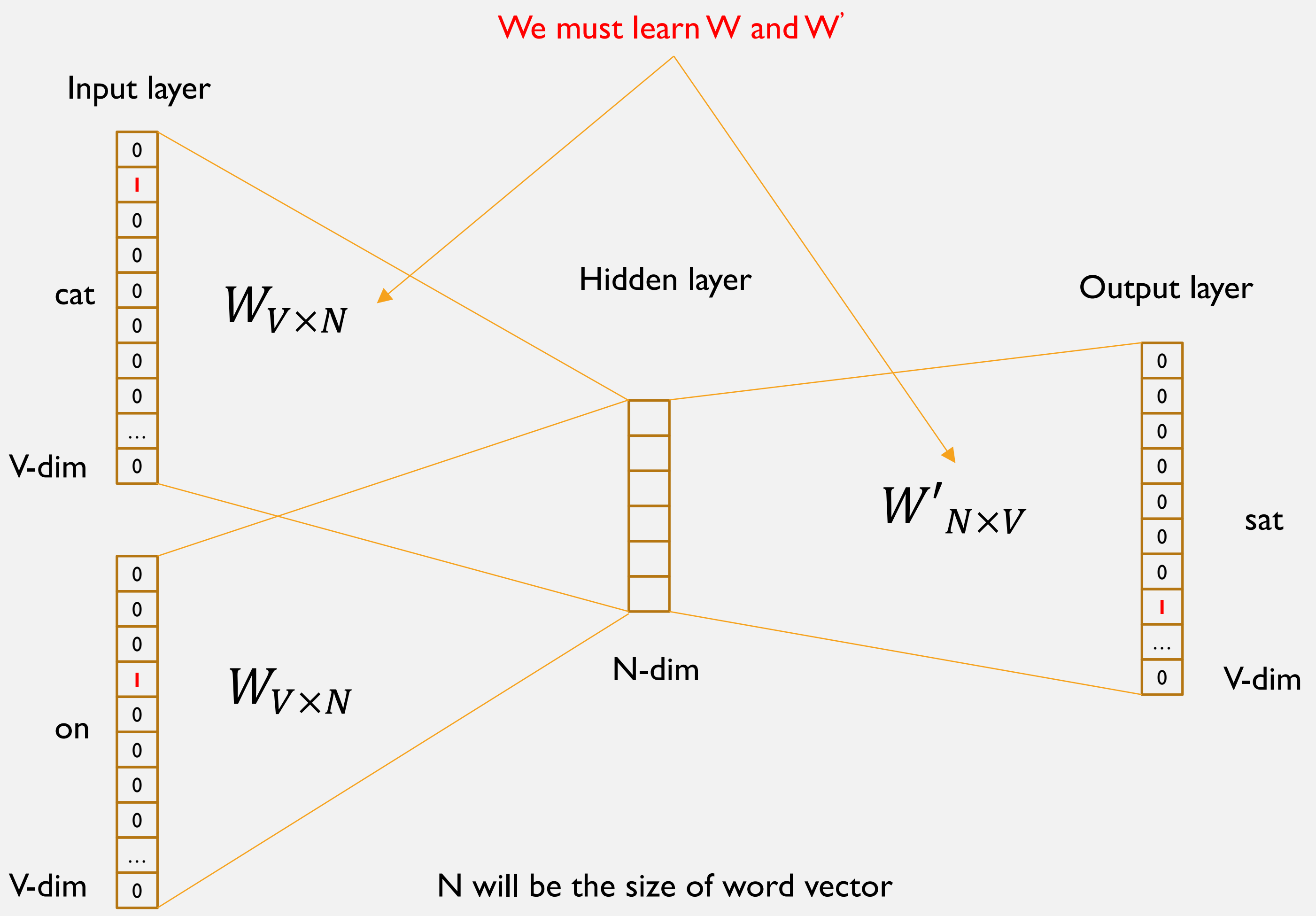
on

floor

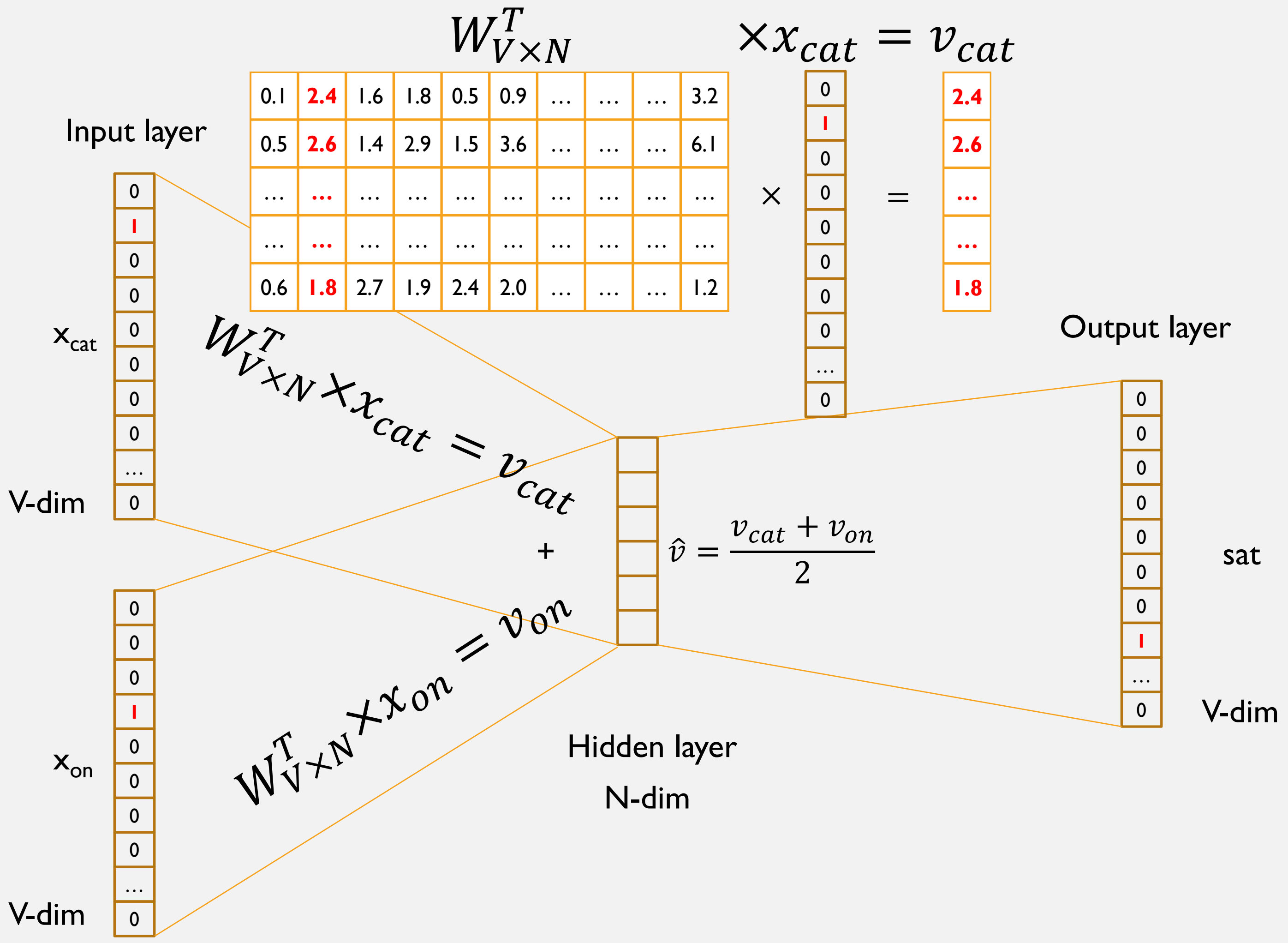


sat

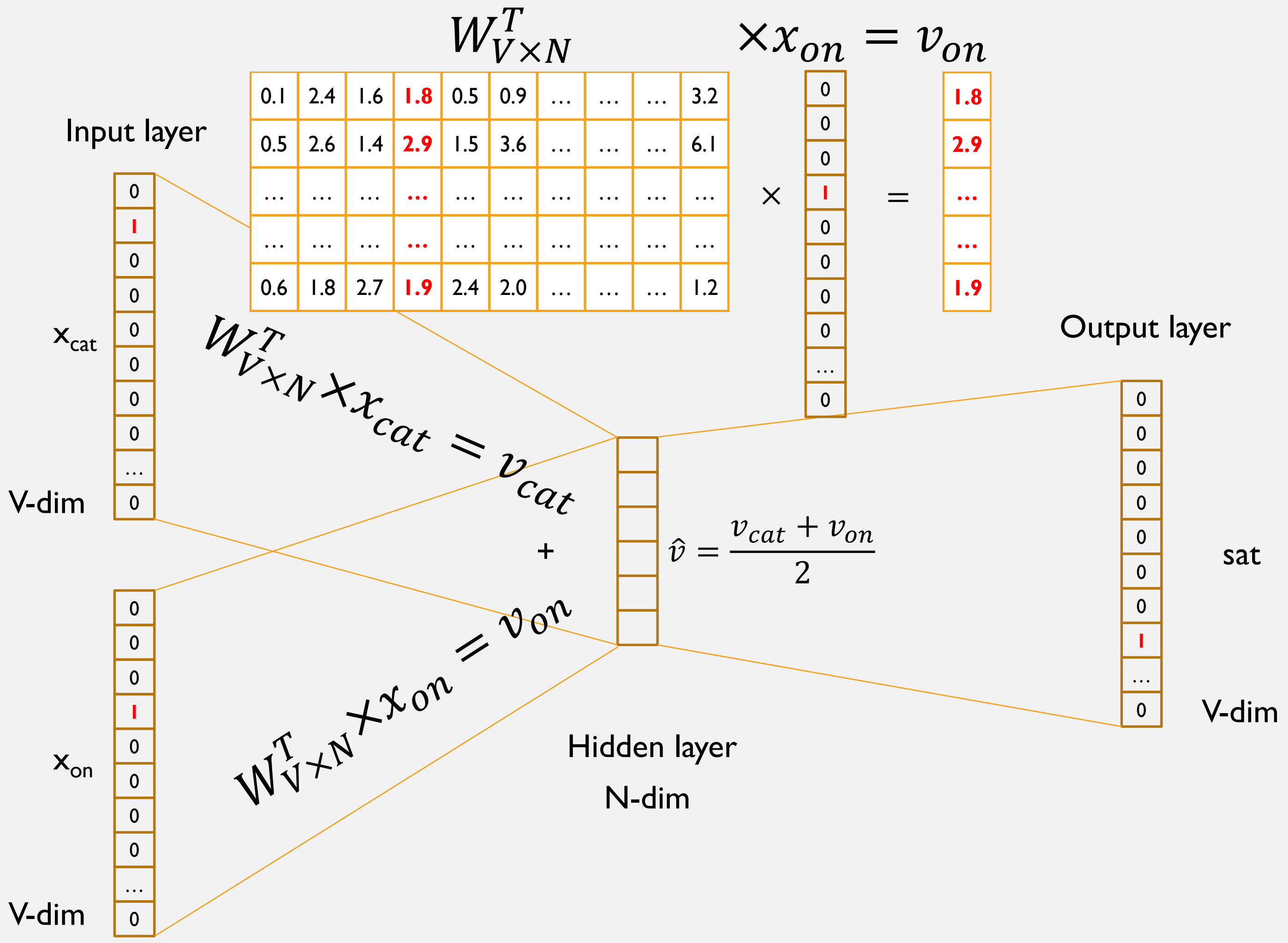

Input layer
Index of cat in vocabulary
cat
on
one-hot vector
Hidden layer
Output layer
sat
one-hot vector
0
1
...



N will be the size of word vector

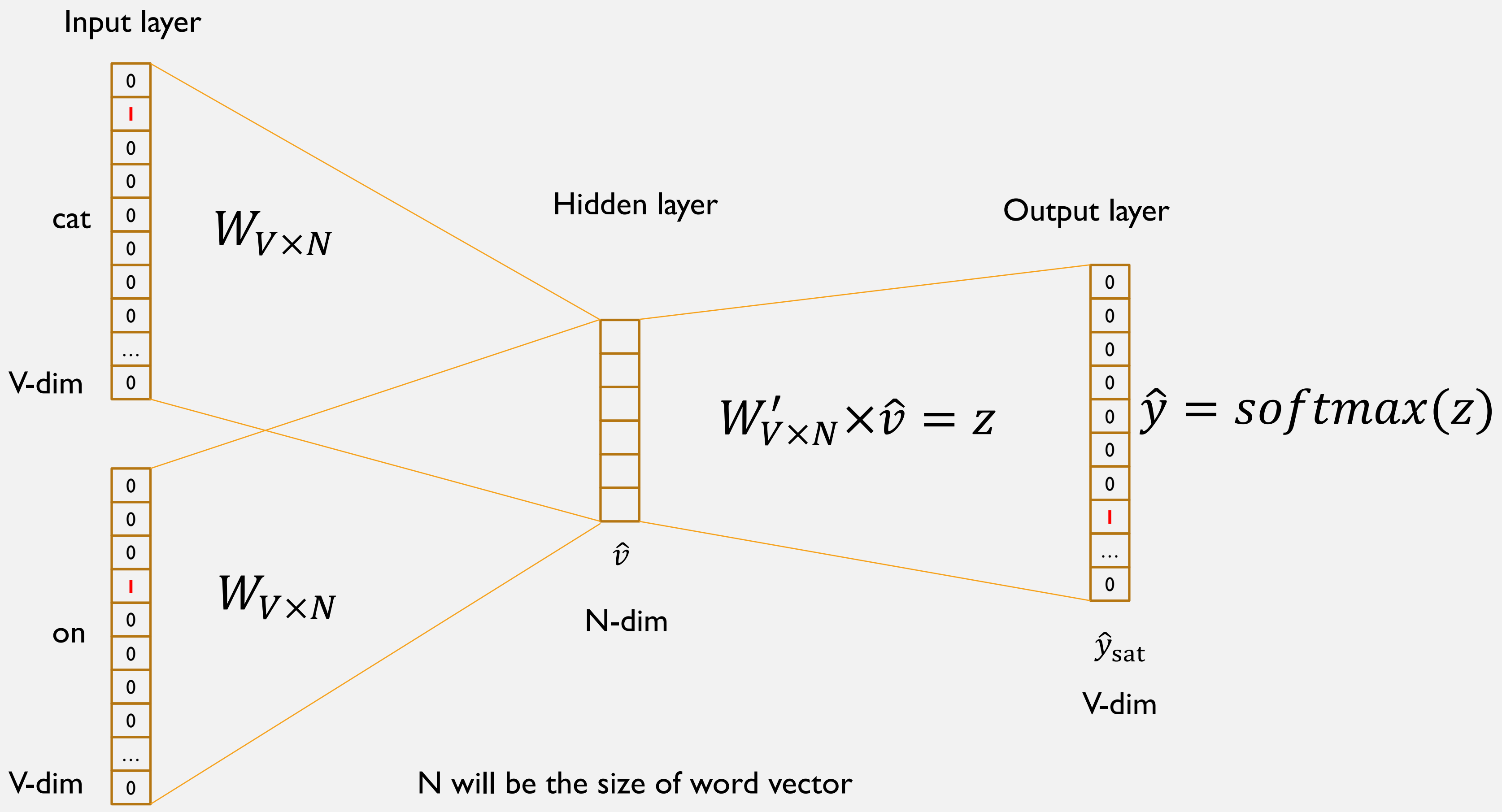


$W^T_{V\times N}$
$\times x_{cat} = v_{cat}$
Input layer
0.1 2.4 1.6 1.8 0.5 0.9 … … … 3.2
0.5 2.6 1.4 2.9 1.5 3.6 … … … 6.1
0.6 1.8 2.7 1.9 2.4 2.0 … … … 1.2
×
=
2.4
2.6
1.8
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
$W^T_{V\times N} \times x_{on} = v_{on}$
x_{on}
V-dim
Hidden layer
N-dim
Output layer
sat
V-dim

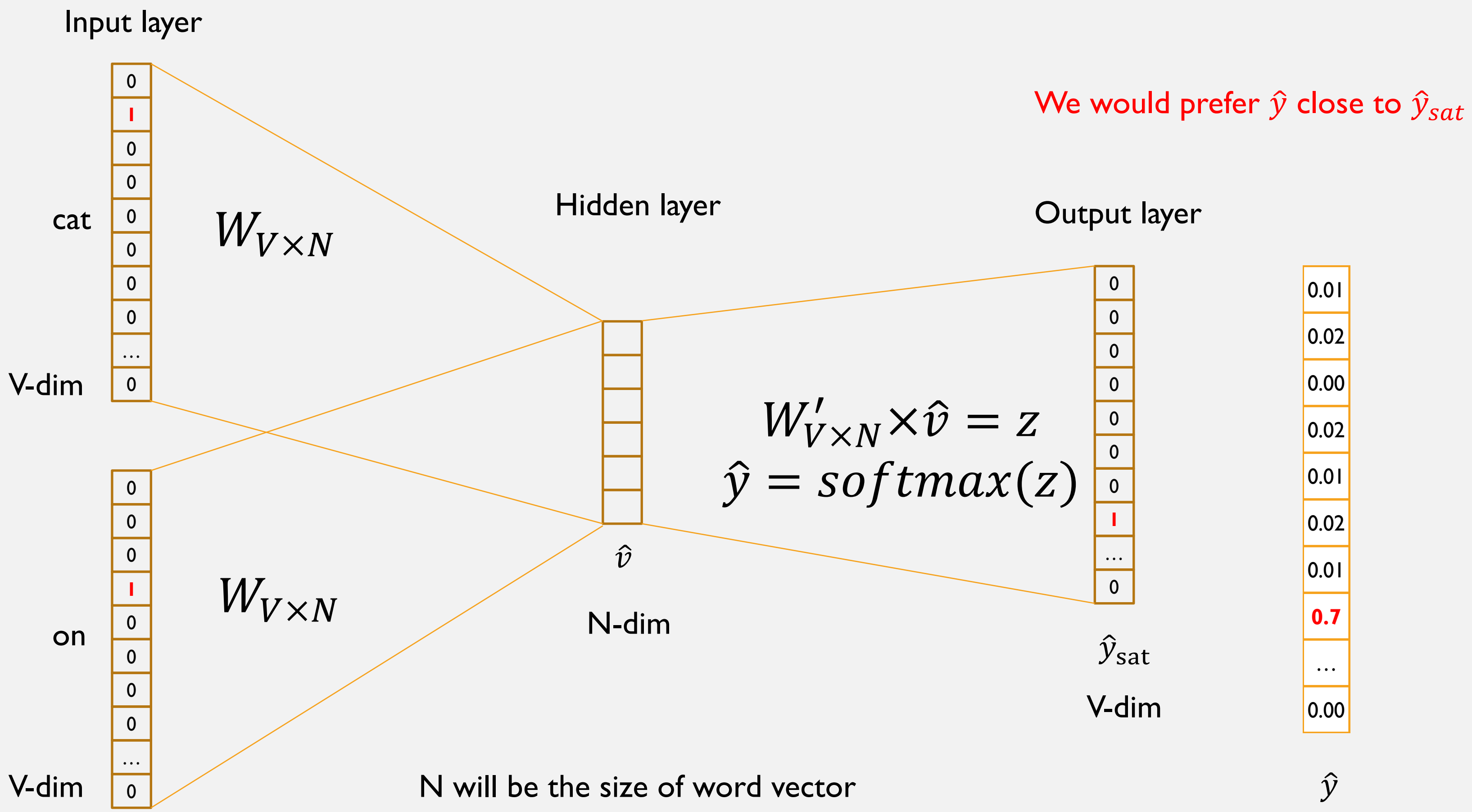

$W^T_{V\times N}$
$\times x_{on} = v_{on}$
Input layer
x_{cat}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
x_{on}
V-dim
$\hat{v} = \frac{v_{cat}+v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim

0.1	2.4	1.6	1.8	0.5	0.9	…	…	…	3.2
0.5	2.6	1.4	2.9	1.5	3.6	…	…	…	6.1
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
0.6	1.8	2.7	1.9	2.4	2.0	…	…	…	1.2

$$\times \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\\vdots\\0\end{bmatrix} = \begin{bmatrix}1.8\\2.9\\\dots\\\dots\\1.9\end{bmatrix}$$

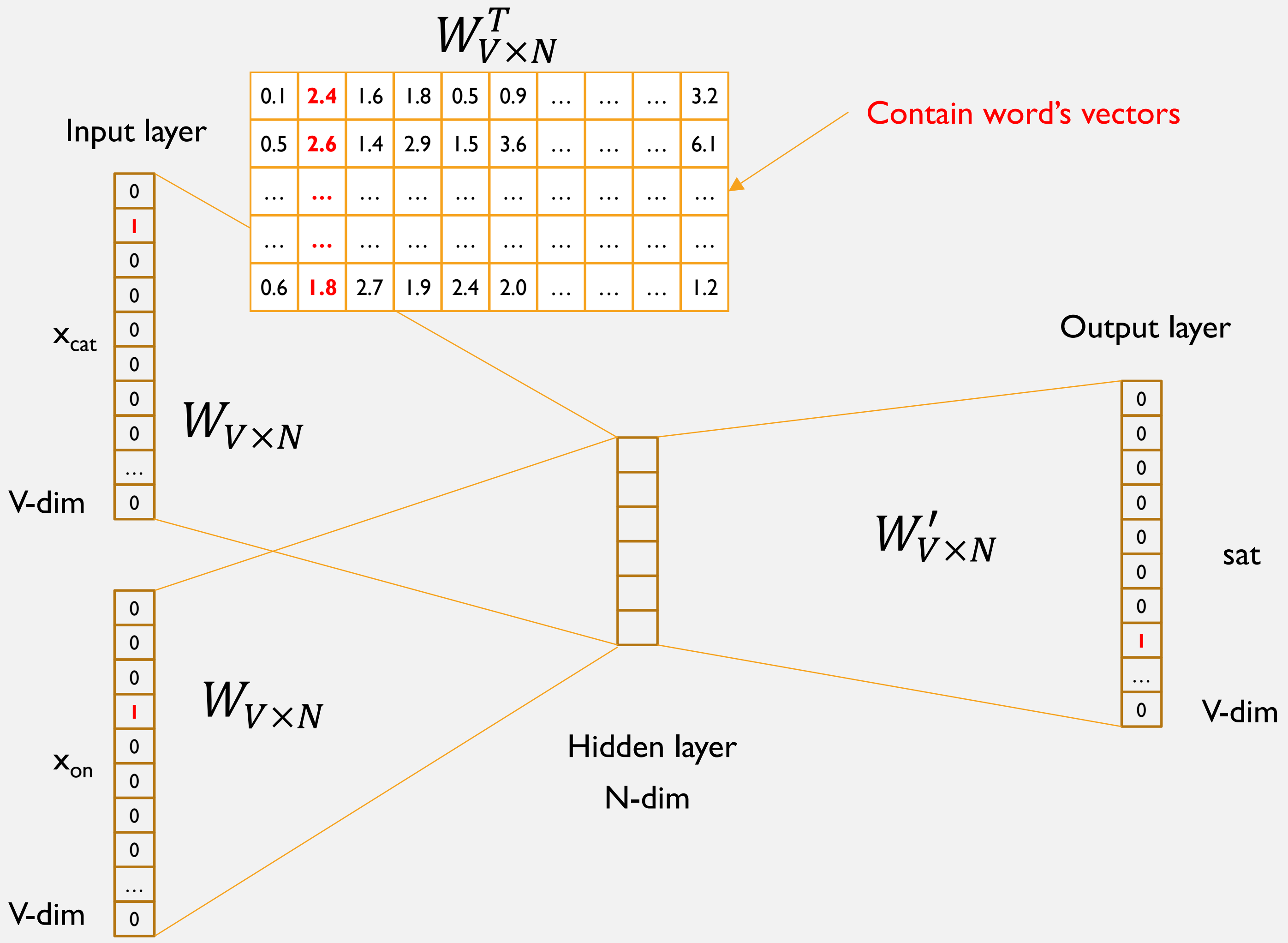


N will be the size of word vector



We would prefer $\hat{y}$ close to $\hat{y}_{sat}$

N will be the size of word vector



We can consider either W or W' as the word's representation. Or even take the average.

SOME INTERESTING RESULTS

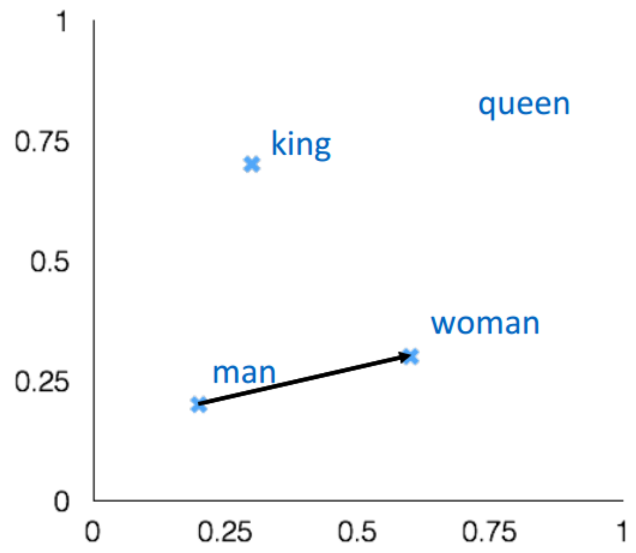
Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

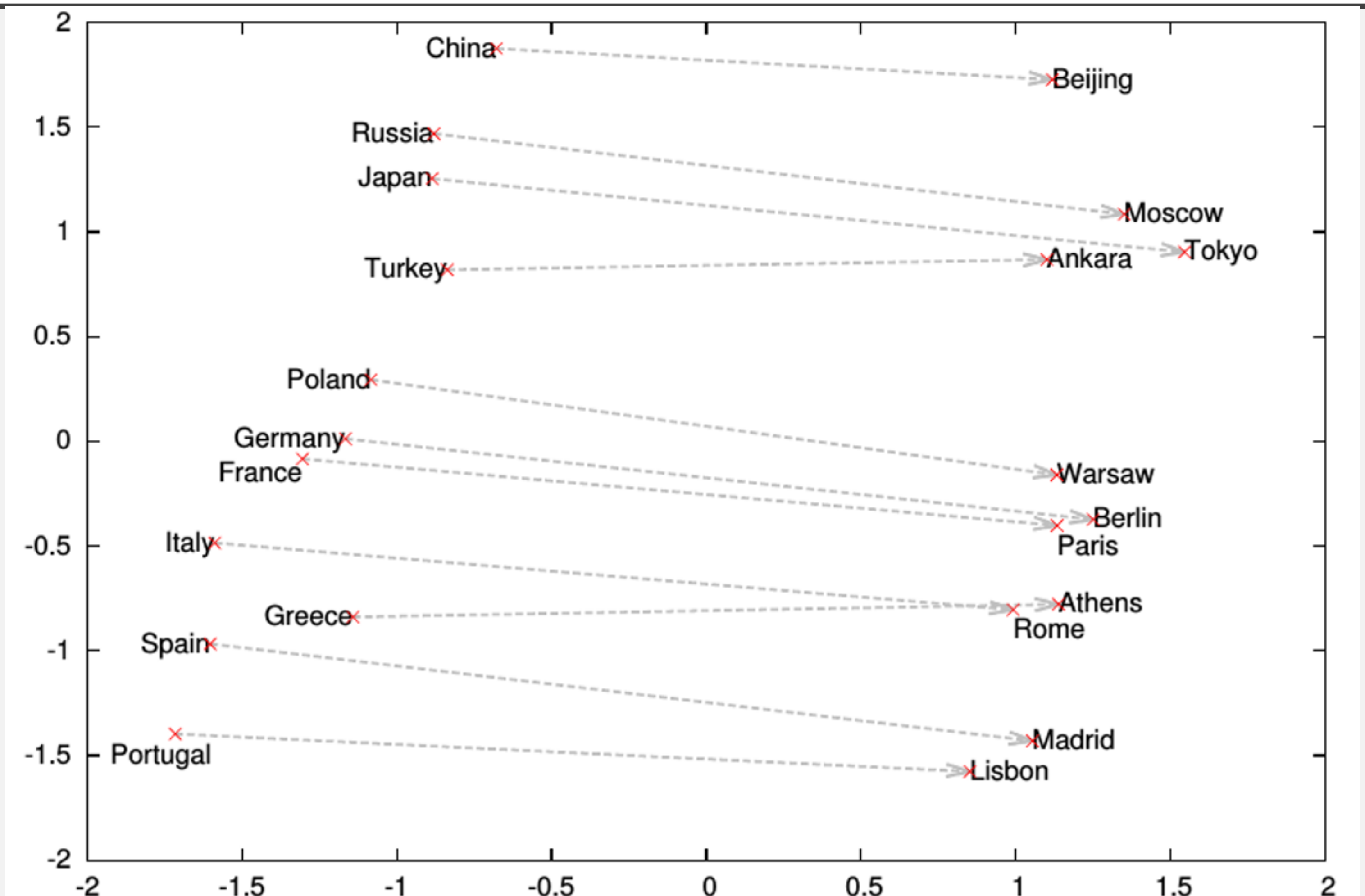
$\boxed{\text{a:b :: c:?}} \longrightarrow \boxed{d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}}$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



WORD ANALOGIES



NOW WHAT?

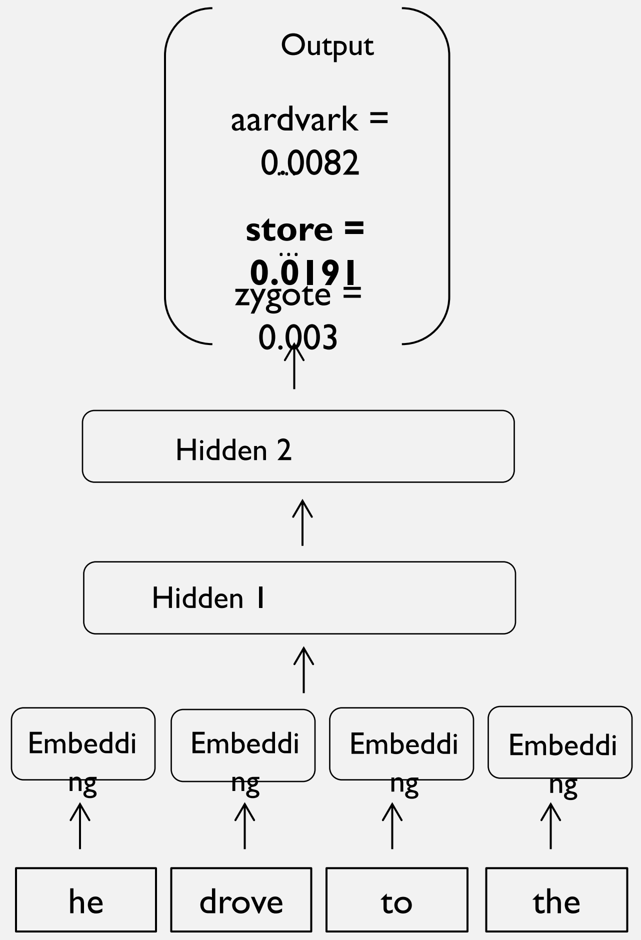
- Word2Vec == Deep Learning
  - No!
  - The first step
- Embeddings are dense vectors
  - Can encapsulate context information
  - Can serve as information points similar to pixels
  - Serve as input to Deep learning models
  - CNNs, RNNs and many more

DEEP LEARNING AND NLP

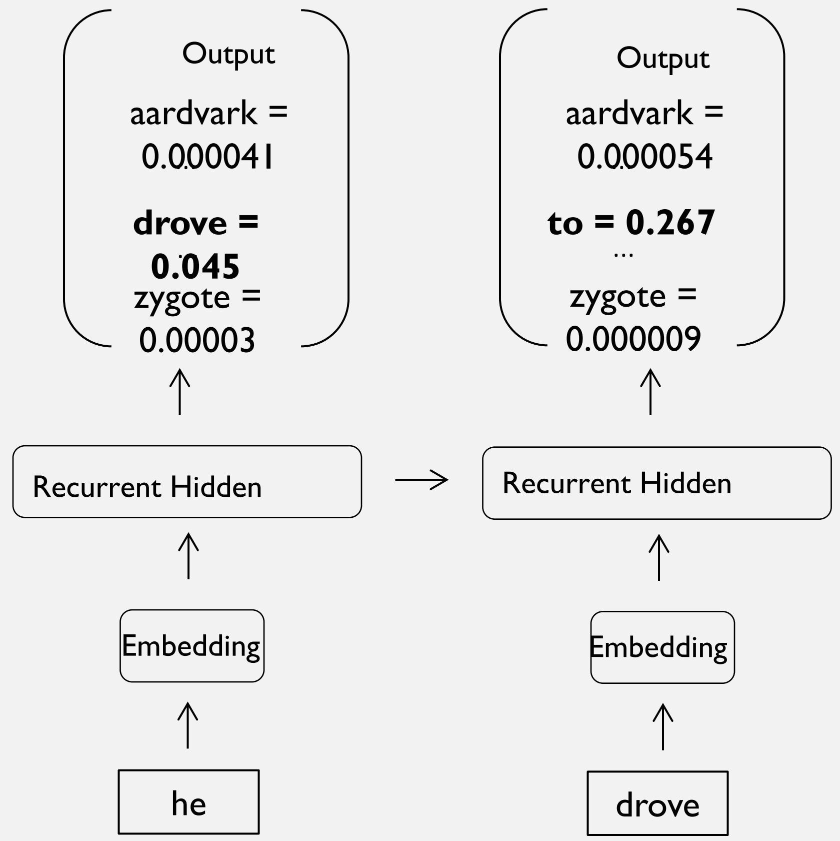
- Tasks
  - Language Modeling
  - Semantic encoding of larger sentential units
  - Semantic similarity at larger granularity for various tasks

NEURAL NETWORK LANGUAGE MODELS (NNLMS)

Feed-forward NNLM



Recurrent NNLM



Output
aardvark = 0.000041
drove = 0.045
...
zygote = 0.00003
Output
aardvark = 0.000054
to = 0.267
...
zygote = 0.000009
Recurrent Hidden
Recurrent Hidden
Embedding
Embedding
he
drove