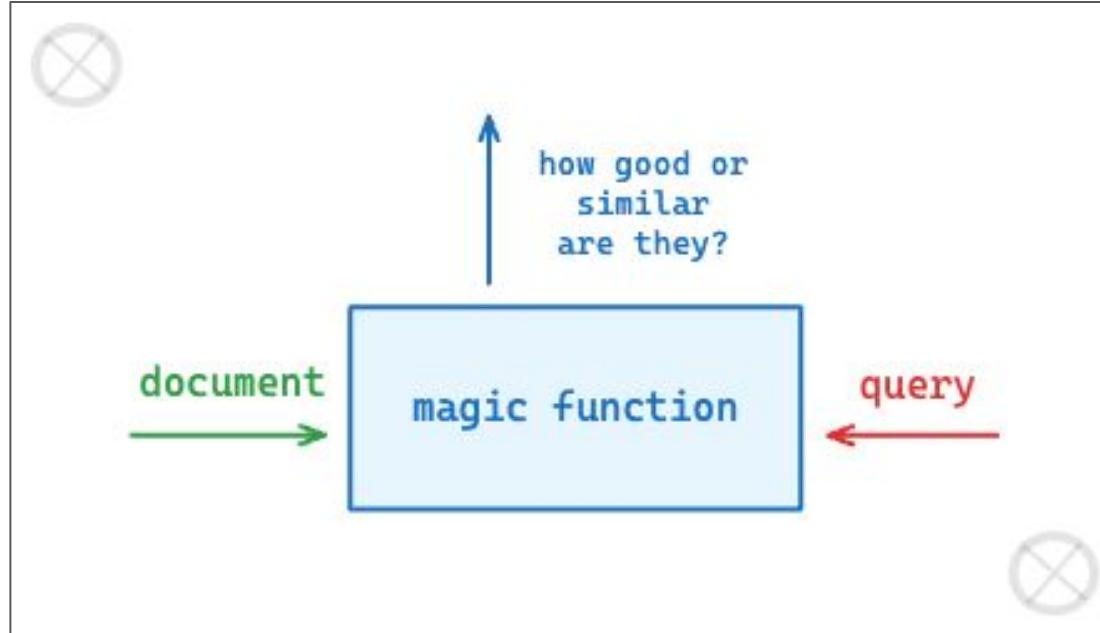


A gentle intro to information retrieval

Learn To Search



Learn To Search & Rank



TF-IDF

Term Frequency

(A)	foo	1/4	0.25
bar	1/4	0.25	
zoo			
zoo	2/4	0.50	

$$TF(t, d) = \frac{f_{t,d}}{\sum_{w \in V} f_{w,d}}$$

$f_{t,d}$ = count_d(t) simple count function :)

Inverse Document Frequency

(A)	foo	ln(3/2)	0.405
bar	ln(3/3)	0.000	
zoo	ln(3/2)	0.405	

$$IDF(t, \mathcal{D}) = \log \frac{|\mathcal{D}|}{df(t)}$$

$$df(t) = \sum_{d \in \mathcal{D}} \mathbf{1}_{\{t \in d\}}$$

loop over all documents and add "1" if it contains the current term "t"

Final Score = TF * IDF

(A)	foo	1/4 * ln(3/2)	0.1
bar	1/4 * ln(3/3)	0.0	
zoo	2/4 * ln(3/2)	0.2	

(B)	foo	1/2 * ln(3/2)	0.2
bar	1/2 * ln(3/3)	0.0	

(C)	zoo	1/2 * ln(3/2)	0.2
bar	1/2 * ln(3/3)	0.0	

Inverted Index

Final Score = TF * IDF

(A)	foo	bar	zoo
	foo	1/4 * ln(3/2)	0.1
	bar	1/4 * ln(3/3)	0.0
	zoo	2/4 * ln(3/2)	0.2

foo	1/4 * ln(3/2)	0.1
bar	1/4 * ln(3/3)	0.0
zoo	2/4 * ln(3/2)	0.2

(B)	foo	bar
	foo	1/2 * ln(3/2)
	bar	1/2 * ln(3/3)

foo	1/2 * ln(3/2)	0.2
bar	1/2 * ln(3/3)	0.0

(C)	zoo	bar
	zoo	1/2 * ln(3/2)
	bar	1/2 * ln(3/3)

zoo	1/2 * ln(3/2)	0.2
bar	1/2 * ln(3/3)	0.0

Inverted Index

foo	(B, 0.2), (A, 0.1)
bar	(A, 0.0), (B, 0.0), (C, 0.0)
zoo	(A, 0.2), (C, 0.2)

Is it good enough?

On the web, this strategy often returns very short documents that are the query plus a few words. For example, we have seen a major search engine return a page containing only "Bill Clinton Sucks" and picture from a "Bill Clinton" query. Some argue that on the web, users should specify more accurately what they want and add more words to their query. We disagree vehemently with this position. If a user issues a query like "Bill Clinton" they should get reasonable results since there is a enormous amount of high quality information available on this topic. Given examples like these, we believe that the standard information retrieval work needs to be extended to deal effectively with the web.



FOUNDERS AND CODERS

1998



The anatomy of a large-scale hypertextual Web search engine¹

Sergey Brin², Lawrence Page^{*,2}

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at <http://google.stanford.edu>

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of Web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the Web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and Web proliferation, creating a Web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale Web search engine — the first such detailed public description we know of to date.

Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want. © 1998 Published by Elsevier Science B.V. All rights reserved.

Keywords: World Wide Web; Search engines; Information retrieval; PageRank; Google



WELCOME TO THE GATEWAY PLACE

Computer Networks and ISDN Systems 30 (1998) 107-117

COMPUTER NETWORKS
ISDN SYSTEMS

The anatomy of a large-scale hypertextual Web search engine¹

Sergey Brin², Lawrence Page^{*,2}

Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at <http://google.stanford.edu>

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of Web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the Web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and Web proliferation, creating a Web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale Web search engine — the first such detailed public description we know of to date.

Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want. © 1998 Published by Elsevier Science B.V. All rights reserved.

Keywords: World Wide Web; Search engines; Information retrieval; PageRank; Google

1. Introduction

The Web creates new challenges for information retrieval. The amount of information on the Web is growing rapidly, as well as the number of new users inexperienced in the art of Web research. People are

¹ Corresponding author.
² There are two versions of this paper — a longer full version and a shorter printed version. The full version is available on the Web and the conference CD-ROM.

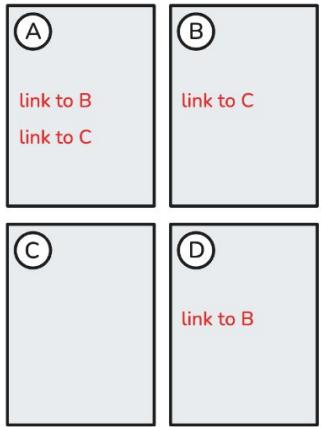
E-mail: sergey.page@cs.stanford.edu

0169-7552/98/\$19.00 © 1998 Published by Elsevier Science B.V. All rights reserved.
PII S0169-7552(98)00110-X

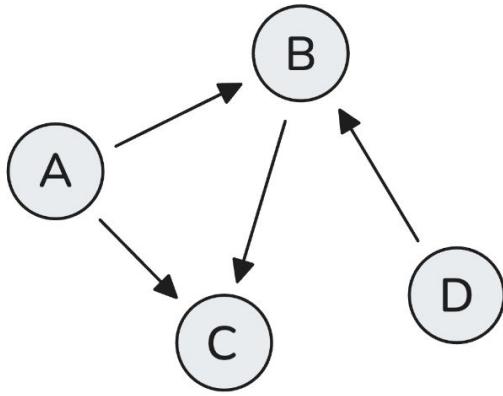
likely to surf the Web using its link graph, often starting with high quality human maintained indices such as [Yahoo!](http://www.yahoo.com)³ or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead

³ <http://www.yahoo.com>

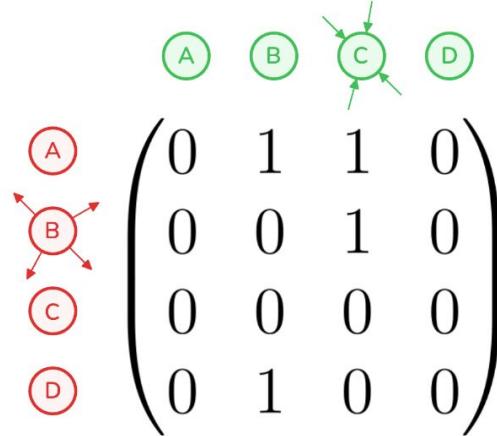
Basic Graph Theory



Documents



Graph



Adjacency Matrix

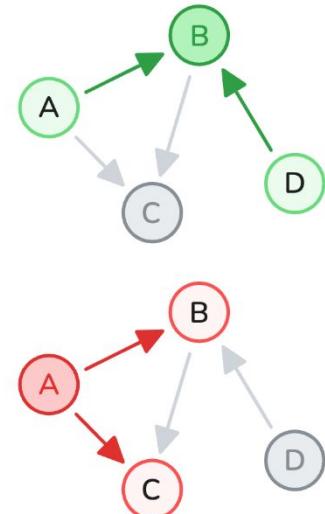
Point of views

outward

starting from this node
what is the probability
of ending in the next node

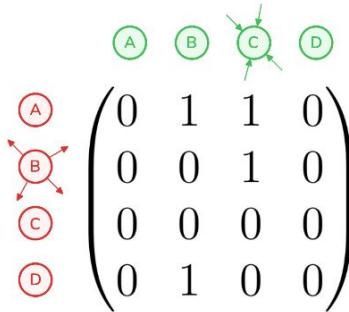
inward just arrived here, what is the probability I came from a specific node

→	$p_{a \rightarrow a}$	$p_{a \rightarrow b}$	$p_{a \rightarrow c}$	$p_{a \rightarrow d}$
→	$p_{b \rightarrow a}$	$p_{b \rightarrow b}$	$p_{b \rightarrow c}$	$p_{b \rightarrow d}$
→	$p_{c \rightarrow a}$	$p_{c \rightarrow b}$	$p_{c \rightarrow c}$	$p_{c \rightarrow d}$
→	$p_{d \rightarrow a}$	$p_{d \rightarrow b}$	$p_{d \rightarrow c}$	$p_{d \rightarrow d}$



From Graph To Markov Chain

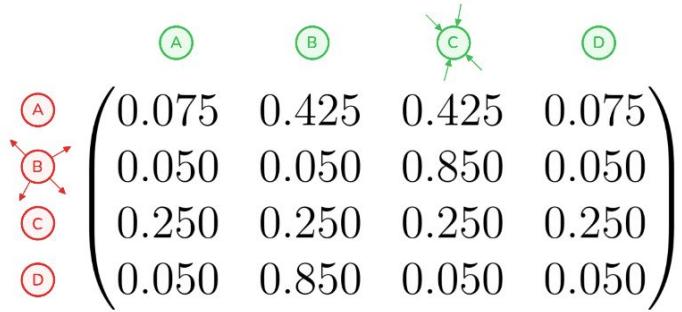
Adjacency Matrix



Teleportation alpha=0.15

$$p_{ij} = \begin{cases} \frac{1 - \alpha}{k_i}, & \text{if } a_{ij} = 1 \text{ and } k_i > 0 \\ \frac{\alpha}{n - k_i}, & \text{if } a_{ij} = 0 \text{ and } k_i > 0 \\ \frac{1}{n}, & \text{if } k_i = 0 \end{cases}$$

Transition Matrix



The transition matrix is derived from the adjacency matrix and a teleportation factor $\alpha = 0.15$. The matrix is:

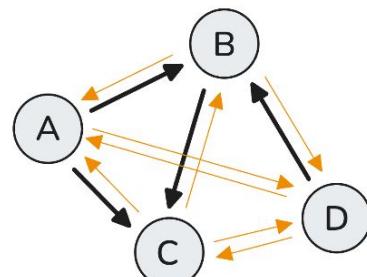
$$\begin{pmatrix} 0.075 & 0.425 & 0.425 & 0.075 \\ 0.050 & 0.050 & 0.850 & 0.050 \\ 0.250 & 0.250 & 0.250 & 0.250 \\ 0.050 & 0.850 & 0.050 & 0.050 \end{pmatrix}$$

The Minions Walk

Transition Matrix

$$\begin{array}{cccc}
 & \textcircled{A} & \textcircled{B} & \textcircled{C} & \textcircled{D} \\
 \textcircled{A} & 0.075 & 0.425 & 0.425 & 0.075 \\
 \textcircled{B} & 0.050 & 0.050 & 0.850 & 0.050 \\
 \textcircled{C} & 0.250 & 0.250 & 0.250 & 0.250 \\
 \textcircled{D} & 0.050 & 0.850 & 0.050 & 0.050
 \end{array}$$

Fully Connected Graph



The Stationary Equation

$$\pi = \pi T$$

$$\begin{aligned}
 \pi &= [\pi_A, \pi_B, \pi_C, \pi_D] \\
 \pi^{(0)} &= [0.25_A, 0.25_B, 0.25_C, 0.25_D]
 \end{aligned}$$

Random walks & Markov chains

- Andrey Markov introduced them in 1906 (~120 years ago)
- Random walks are a core model in stochastic processes
- Model everything from: weather, games & stock markets
- PageRank didn't invent anything new mathematically
- Teleportation makes PageRank's random walk ergodic

Can we do better than TF-IDF?

Distribution hypothesis

What is the meaning of **bardiwac**?

- He poured her a glass of **bardiwac**.
- Beef tastes better when you drink **bardiwac** with it.
- Nigel felt dizzy after too much **bardiwac**.
- The **bardiwac** grapes grow well in warm sunshine.
- I ate bread and cheese with this good **bardiwac**.
- They served dark red **bardiwac** and a light, sweet white wine.

word2vec - recipe

1

Treat the entire corpus as one long string. Slide a window of five tokens across it; for each window, mask the middle token and use the remaining four to predict the "?"

For dinner we served dark red bardiwac with steak and mushrooms.

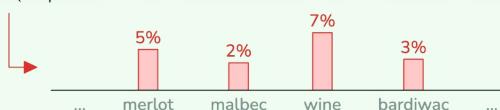
For dinner we served dark red bardiwac with steak and mushrooms.

$$\begin{array}{ccc} \text{dark red} & ? & \text{with steak} \\ \hline ? = f(\text{dark, red, with, steak}) \end{array}$$

2

Because any given context can sensibly be completed by several different words, we don't force the model to output one "correct" token. Instead, we teach it to return a probability distribution over the whole vocabulary

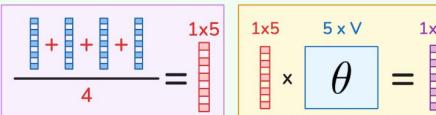
$$p_\theta(?) | \text{"dark", "red", "with", "steak"})$$



3

CBOW
Continuous Bag Of Words

dark	→	1	dark	[0.4, 0.1, 2.2, 0.9, 1.4]
red	→	2	steak	[0.5, 1.1, 0.5, 1.0, 2.0]
with	→
steak	→	23,451	with	[0.5, 0.1, 1.2, 0.7, 0.1]
		23,452	red	[0.1, 1.1, 0.2, 0.5, 0.2]



Efficient Estimation of Word Representations in Vector Space

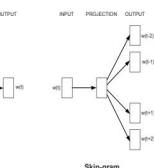


Figure 1: New model architecture. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

R words from the center of the current word as correct labels. This will require us to do $R \times 2$ word classifications, with the current word as input, and each of the $R + R$ words as output. In the following experiments, we use $C = 10$.

4 Results

To compare the quality of different versions of word vectors, previous papers typically use a table showing example words and their most similar words, respectively. Although it is a good way to show the quality of the learned vectors, perhaps some other comparison is much more challenging when subjecting those vectors to a more complex similarity task, as follows. We follow the approach of Mikolov et al. [20] to measure the quality of learned vectors. For example, word *kg* is similar to *kgg* in the same sense that *small* is similar to *small*. Example of another type of relationship can be word pairs (e.g., *figer* and *small* - *smaller*) [20]. We further denote the word pairs as *kg*-*kgg* and *small*-*smaller*. The question is: what is the word that is similar to *kg* in the same sense as *figer* is similar to *kgg*?

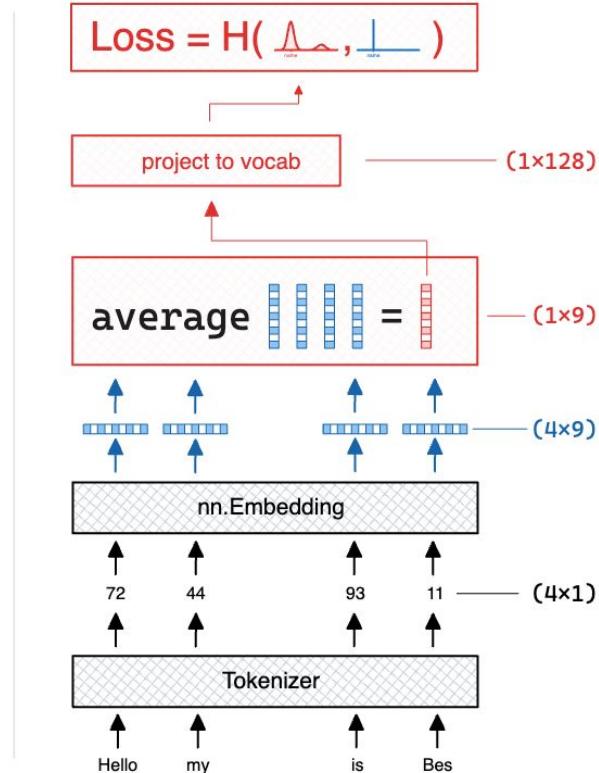
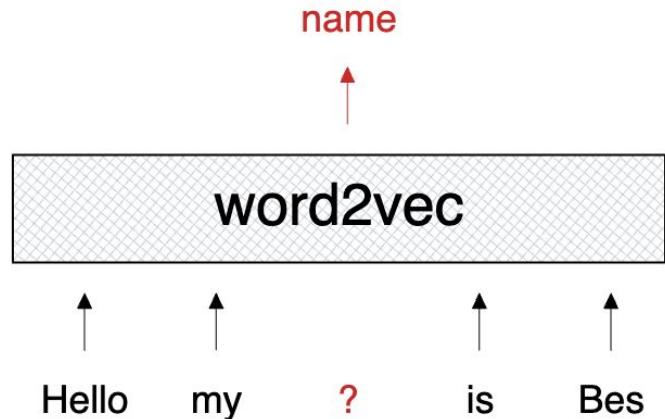
Somewhat surprisingly, these questions can be answered by performing simple algebraic operation with the learned word vectors. For example, if we want to know what word is in the same sense as *figer* is similar to *kgg*, we can simply compute vector $X = \text{vector}(\text{"figer"}) - \text{vector}(\text{"kg"}) + \text{vector}(\text{"kgg"})$ and use it as the answer to the question (we discuss the input question words during this search). When the word vectors are well trained, it is possible to find the correct answer (word *smaller*).

Finally, we found that when we train high dimensional word vectors on a large amount of data, the resulting vectors can be used to answer very subtle semantic relationships between words, such as a set of words that are all related to the same concept. This kind of semantic reasoning combined with such semantic relationships could be used to improve many existing NLP applications, such as machine translation, information retrieval and question answering systems, and maybe offer future applications yet to be invented.

5

Mikolov et al. 2013

word2vec - cbow



```

1  #
2  #
3  #
4  import torch
5  import torch.nn.functional as F
6  #
7  #
8  #
9  #
10 vocab = {
11     "Hello": 72,
12     "my": 44,
13     "name": 21,
14     "is": 93,
15     "Bes": 11
16 }
17 #
18 #
19 #
20 sentence = ["Hello", "my", "is", "Bes"]
21 #
22 #
23 class CBOW(torch.nn.Module):
24     def __init__(self):
25         super(CBOW, self).__init__()
26         self.emb = torch.nn.Embedding(128, 9)
27         self.linear = torch.nn.Linear(9, 128)
28     def forward(self, inputs):
29         embs = self.emb(inputs)
30         embs = embs.mean(dim=1)
31         out = self.linear(embs)
32         out = F.log_softmax(out, dim=1)
33         return out
34
35
36
37
38

```

Notice something...



This is an encoder model

Given any sentence, the model
will produce a vector representation.

Search Systems

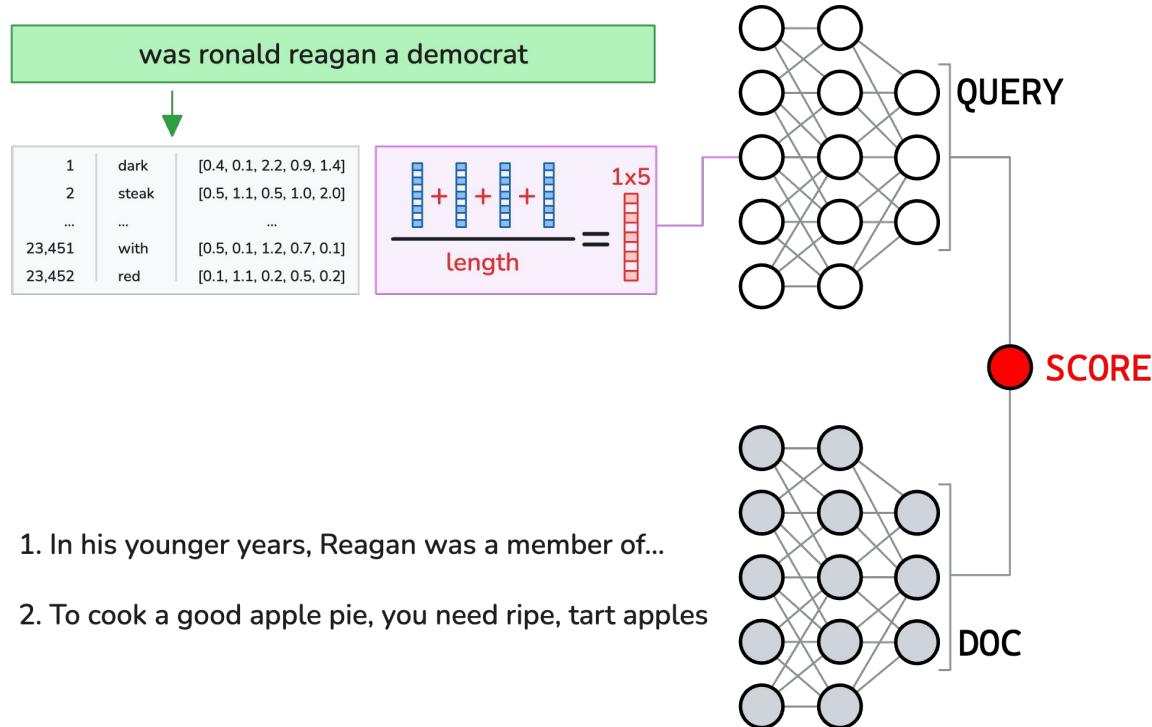
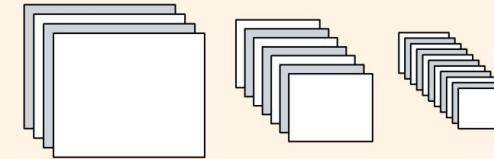


Image representation

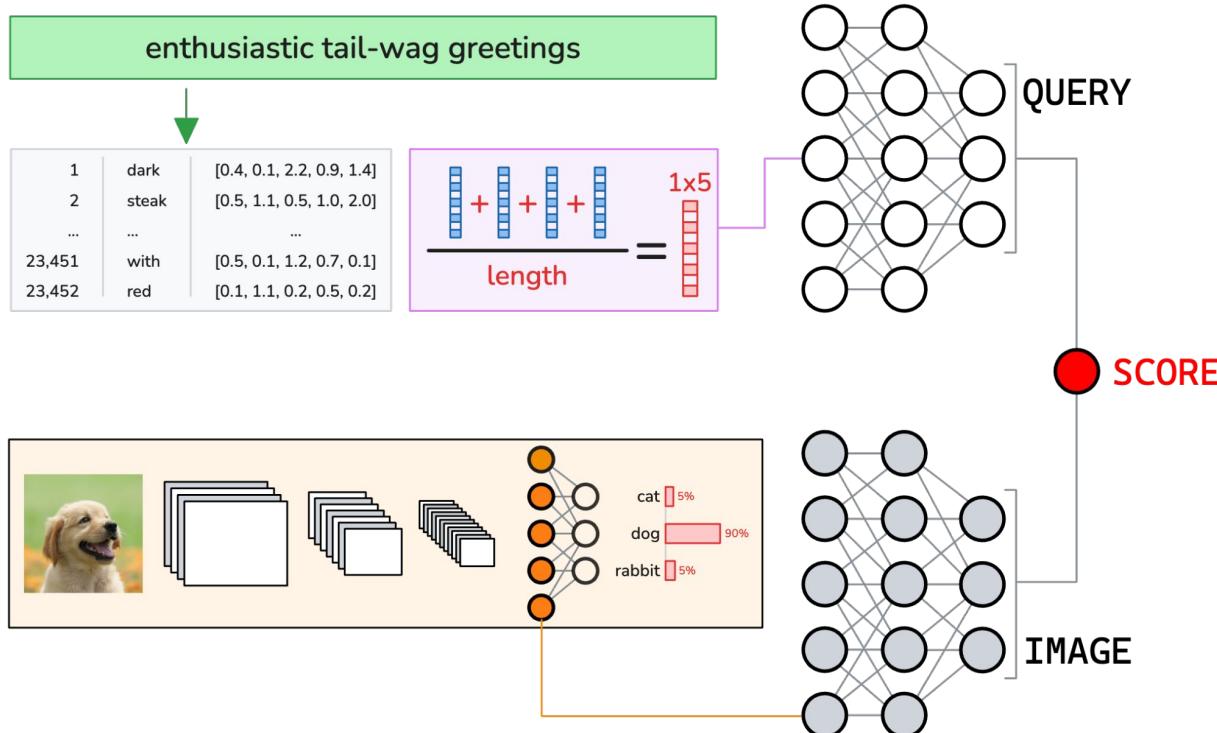
Convolutional Neural Network (CNN)



This is the image representation
in vector space, we can called it "embedding"



Multimodal Search Systems



Recap

We covered tons!

- classic TF-IDF
- PageRank algorithm and the birth of Google
- Graphs and Markov Chains close connection
- multimodal search systems

Thank you!