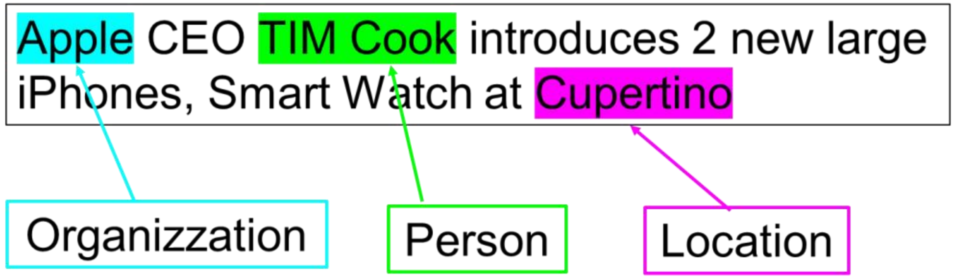
Natural Language Processing and Word Embedding

Prof. Giuseppe Serra

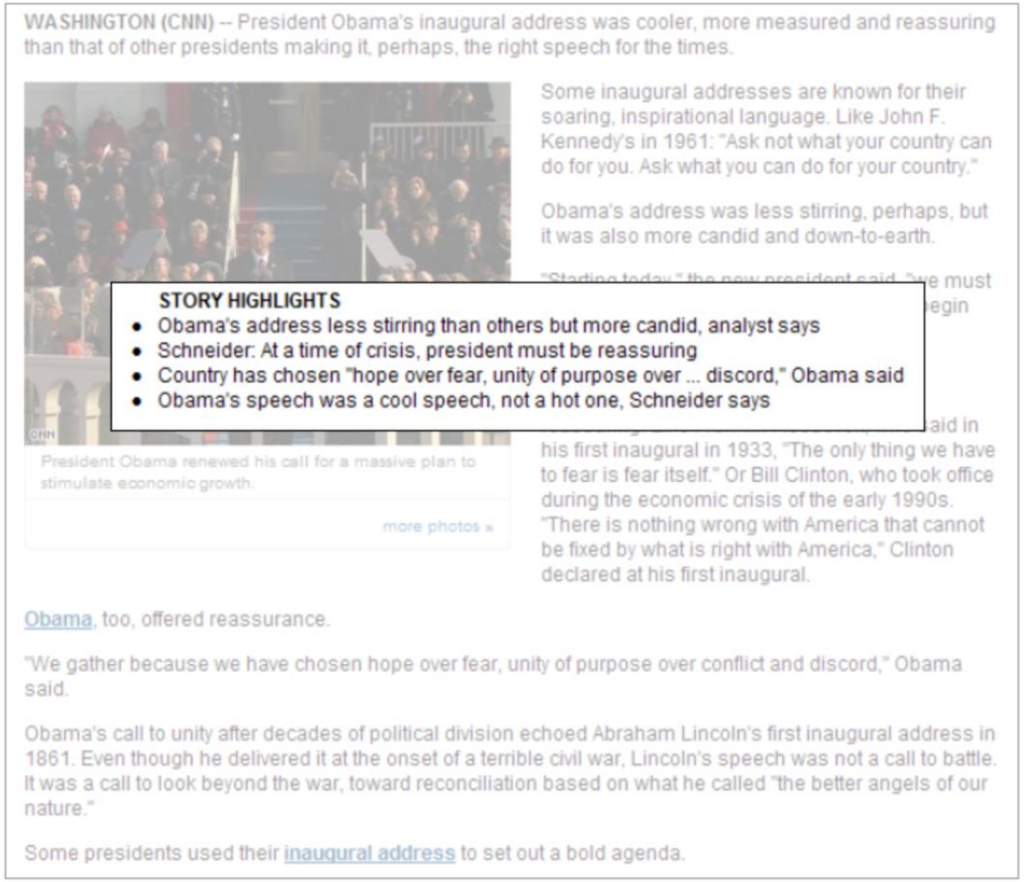
University of Udine

Document analysis

Named Entities Extraction



Summarization


WASHINGTON (CNN) -- President Obama's inaugural address was cooler, more measured and reassuring than that of other presidents making it, perhaps, the right speech for the times.

Some inaugural addresses are known for their soaring, inspirational language. Like John F. Kennedy's in 1961: "Ask not what your country can do for you. Ask what you can do for your country."

Obama's address was less stirring, perhaps, but it was also more candid and down-to-earth.

STORY HIGHLIGHTS

- Obama's address less stirring than others but more candid, analyst says
- Schneider: At a time of crisis, president must be reassuring
- Country has chosen "hope over fear, unity of purpose over ... discord," Obama said
- Obama's speech was a cool speech, not a hot one, Schneider says

President Obama renewed his call for a massive plan to stimulate economic growth.

more photos »

his first inaugural in 1933, "The only thing we have to fear is fear itself." Or Bill Clinton, who took office during the economic crisis of the early 1990s. "There is nothing wrong with America that cannot be fixed by what is right with America," Clinton declared at his first inaugural.

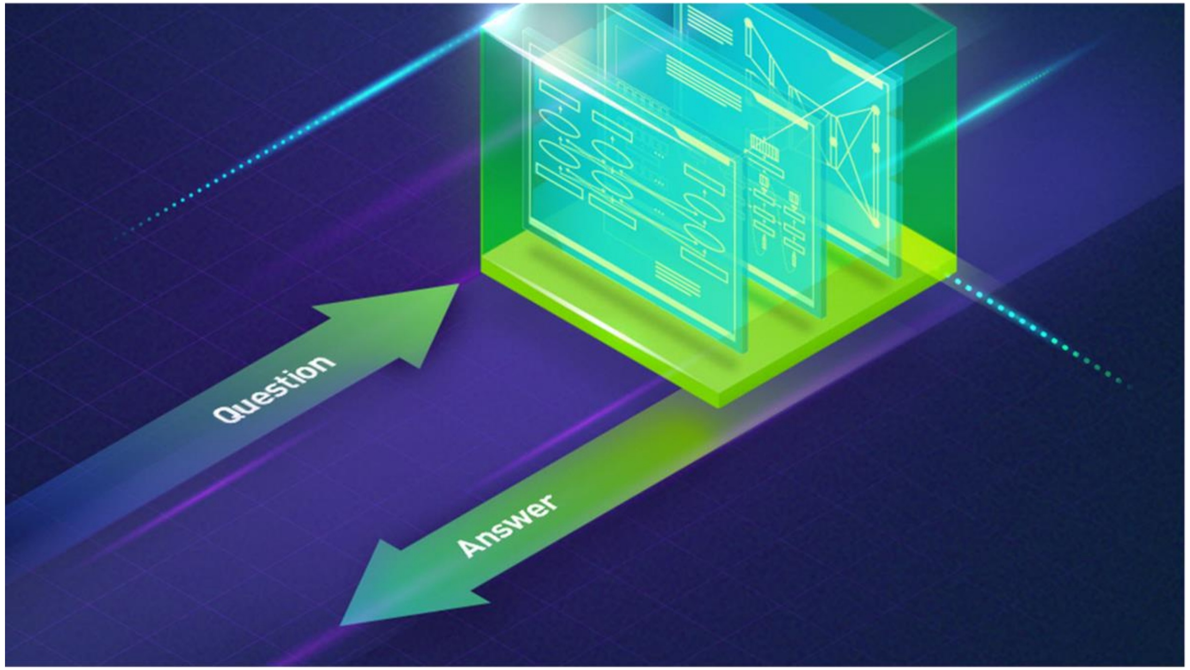
Obama, too, offered reassurance.

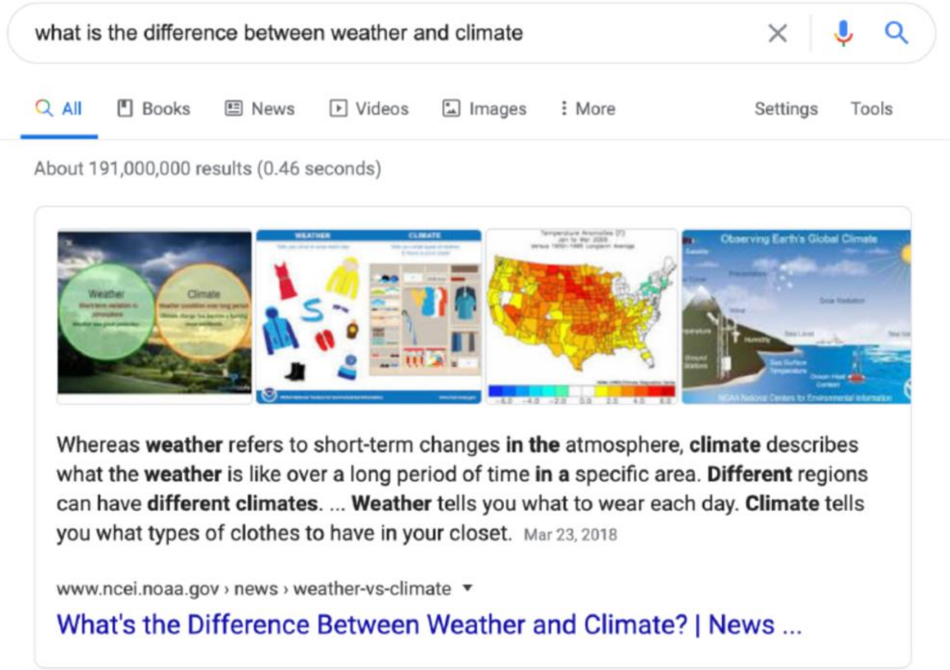
"We gather because we have chosen hope over fear, unity of purpose over conflict and discord," Obama said.

Obama's call to unity after decades of political division echoed Abraham Lincoln's first inaugural address in 1861. Even though he delivered it at the onset of a terrible civil war, Lincoln's speech was not a call to battle. It was a call to look beyond the war, toward reconciliation based on what he called "the better angels of our nature."

Some presidents used their inaugural address to set out a bold agenda.

Question Answering





Social Media Intelligence

forbes.com

Chiara Ferragni, Instagram's It Girl

Since then, Ferragni and The Blonde Salad have gone from strength to strength: she's launched her own line of apparel and accessories, and employs 20 staffers for a fashion site that competes with traditional media platforms.

The 30-year-old lands at #1 on Forbes' Top Influencers list in the Fashion category, joining other impressive

chiaraferragni
Paris, France
Segui

chiaraferragni So proud for being nominated #1 fashion influencer in the world by @forbes 🥇 Read the full article on my story 🔥 #TheBlondeSaladNeverStops #NeverEver

Carica altri commenti

thobstore Nice post 👍 check my page out

laboccadichiara Ua

emsgrimes123 Congrats ❤

phuongtrangnd Proud of the girl i've followed more than 5years ❤️

beatricinaa Vorrei capire cosa spinge le persone a seguirti

leiasemper Auguri! Sei in donna in gamba! Sembri simpaticissima 😃💐

reneednoble Very impressive... congrats! 💗

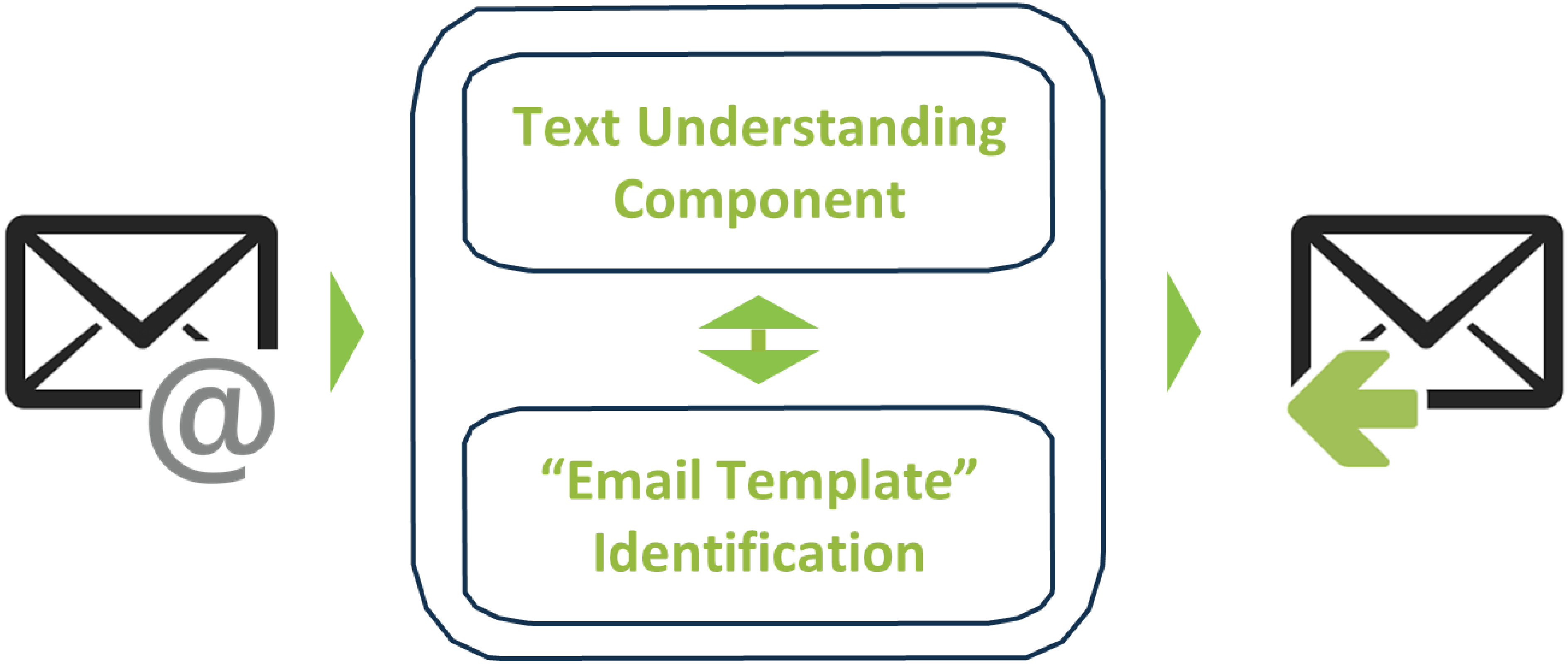
Piace a 117.639 persone

1 GIORNO FA

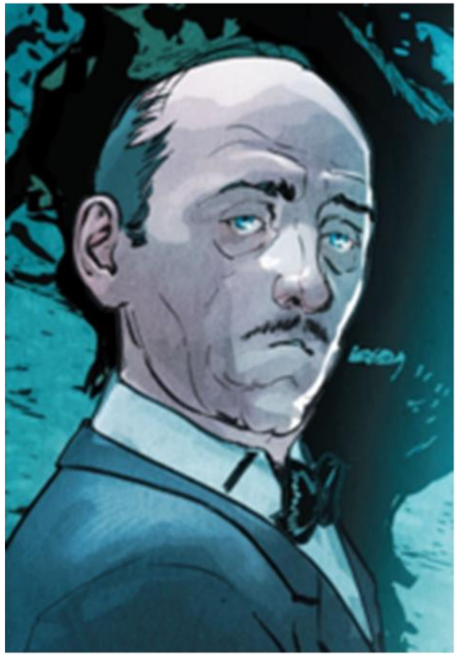
Aggiungi un commento...

Video Machine Translation

Email Auto Responder (AILAB-Udine)

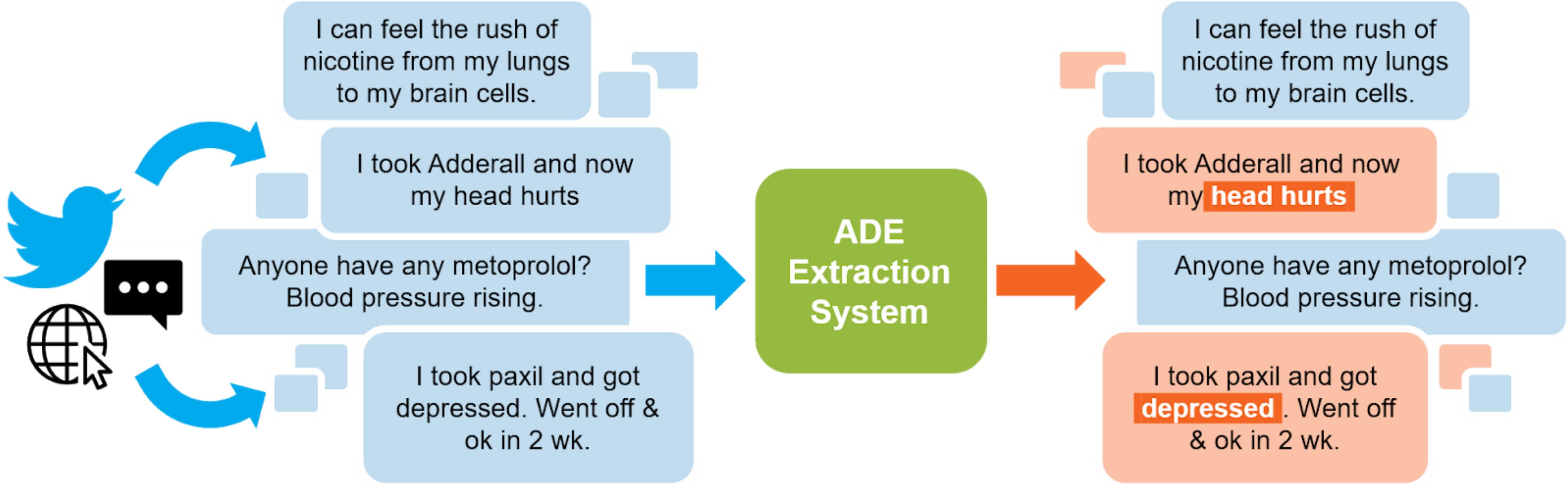


ChatBot



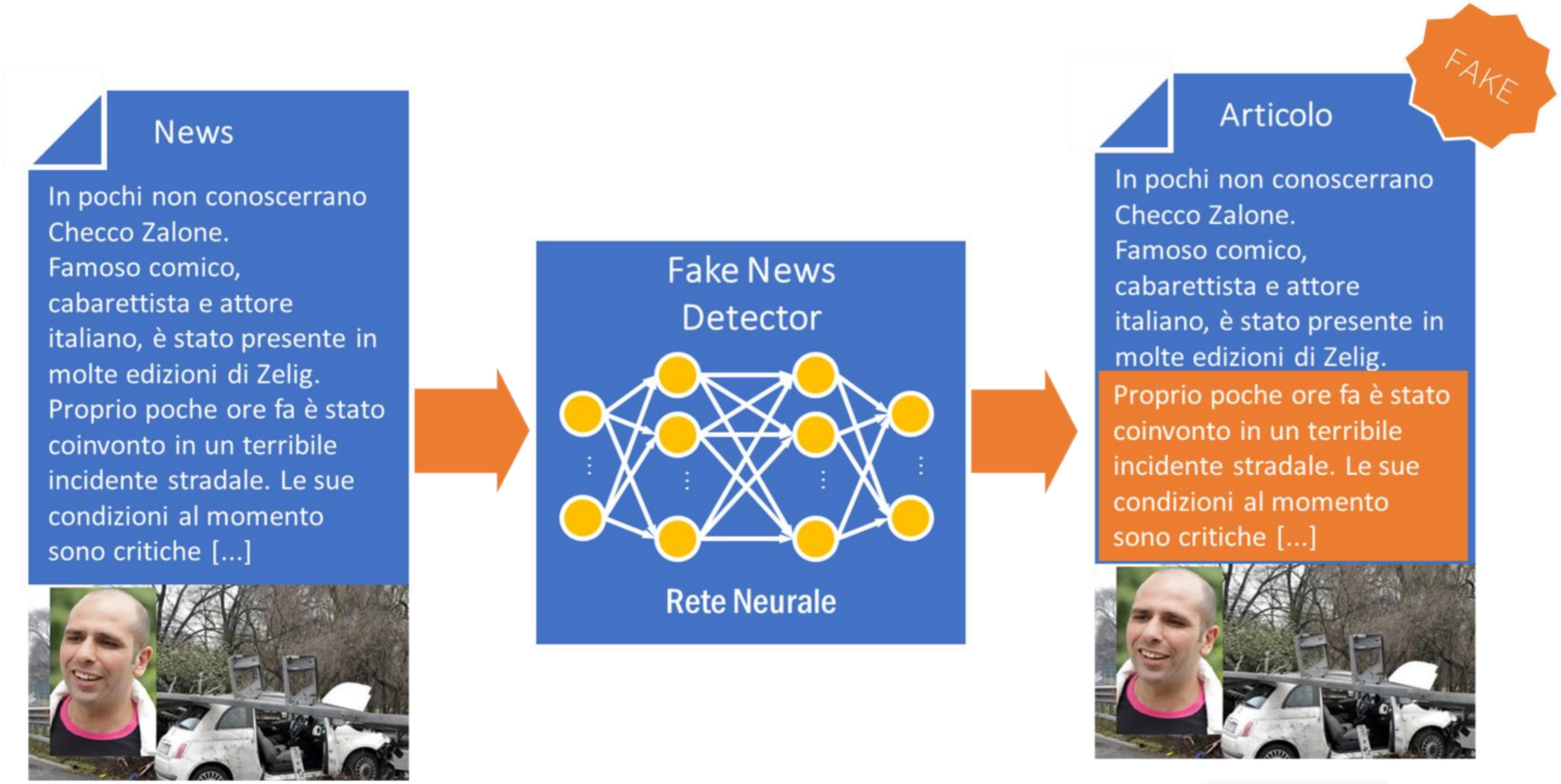

15:58
Alfred
bot
/start 15:54
Benvenuto, sono Alfred, il tuo personale assistente digitale per la gestione della casa. Resto in attesa dei tuoi messaggi, se hai bisogno di aiuto digita il comando /help. 15:54
Buongiorno Alfred 15:54
Buongiorno, come posso esserti utile? 15:54
È accesa la luce in garage? 15:55
Stato del dispositivo luce in garage: spento 15:55
Per favore accendi la luce 15:55
In quale stanza vuoi accenderlo? 15:55
In garage 15:55
Il dispositivo luce in garage è stato acceso 15:55
Qual è lo stato dei climatizzatori? 15:57
Stato del dispositivo climatizzatore: cucina: spento, soggiorno: spento, camera: acceso, bagno: spento, garage: acceso 15:57
Spegni il climatizzatore in garage 15:58
Il dispositivo climatizzatore in garage è stato spento 15:58
Messaggio

COVID-19 Vaccine (AILAB-Udine)

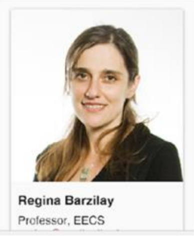


http://ailab.uniud.it/covid-vaccines/

Fake News (AILAB-Udine)



Project in collaboration with Massachusetts Institute of Technology (MIT) – Boston USA



Word Embedding

Word Representation

Word representation techniques represent words as feature vectors.

Combined with Deep Learning these techniques are state of the art in NLP in almost all the tasks.

In the following slides we will see basic and more advance techniques.

Basic Word Representation - Integers

The **simplest way to represent words as numbers** is for a given **vocabulary to assign a unique integer to each word**.

The **set of unique words used in the text corpus** is referred to as the **vocabulary.**

Word	Number
a	1
able	2
about	3
...	...
hand	615
...	...
happy	621
...	...
zebra	1000

Basic Word Representation - Integers

This numbering scheme is simple,

One of the problems is that the order of the words,

For instance, **alphabetical order doesn't make much sense from a semantic perspective**.

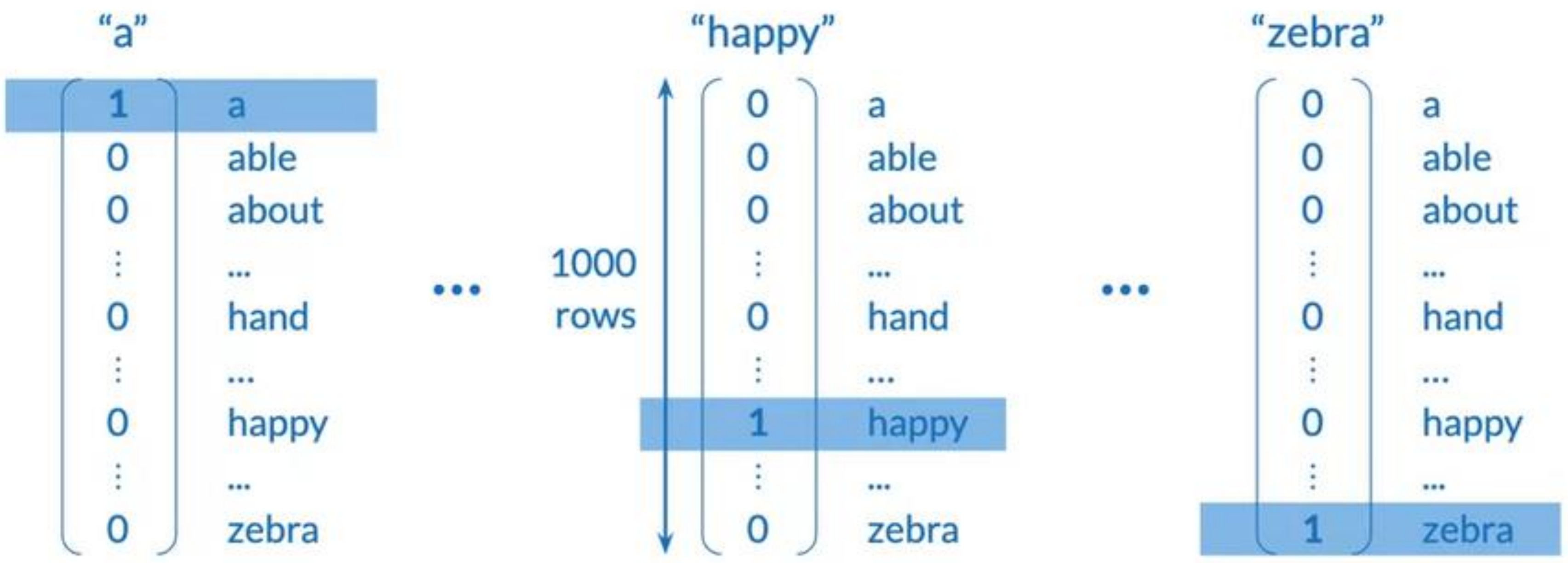
+ Simple
- Ordering: little semantic sense

hand < happy < zebra

615 ?! 621 ?! 1000

Basic Word Representation – One-hot Vectors

One-hot Vectors represent the words using a column vector where each element corresponds to a word of the vocabulary.



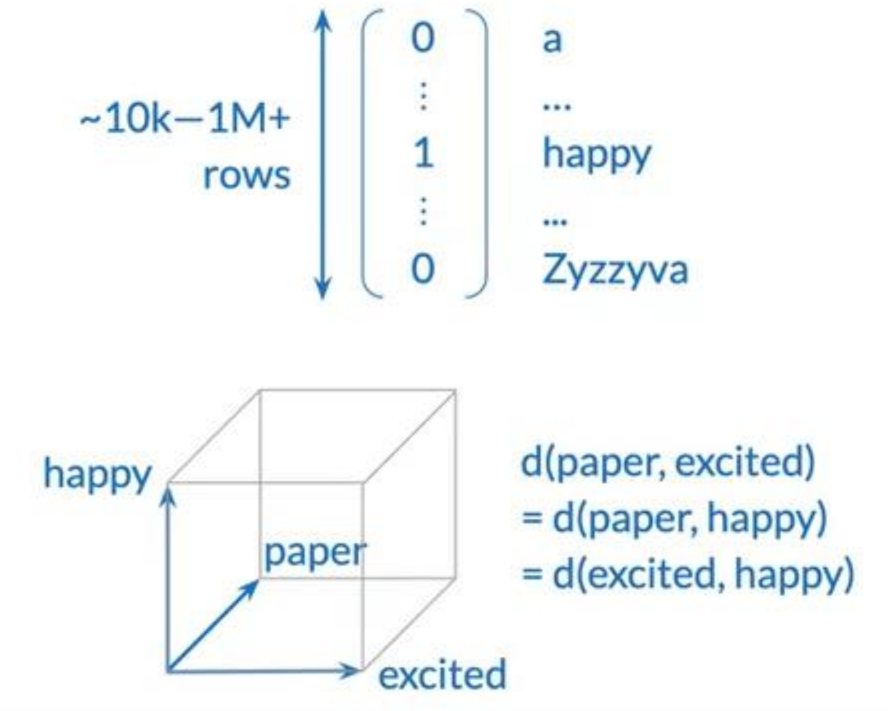
Basic Word Representation – One-hot Vectors

\+ Simple

\+ No implied ordering

\- Huge vectors

\- No embedded meaning

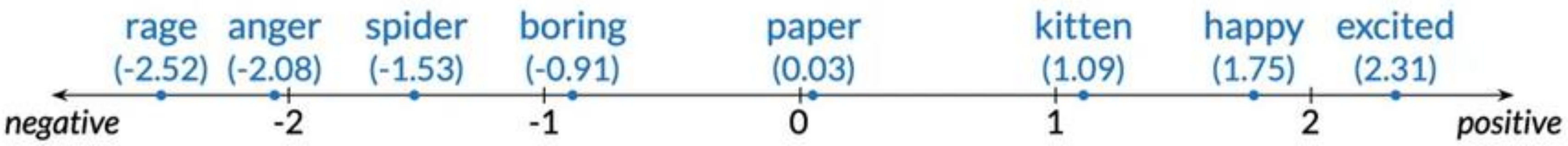


Basic Word Representation – Word Embedding

Word embeddings are vectors that are carrying meaning.

Consider a horizontal number line like this. Words on the left are considered negative in some way and words on the right are considered positive in some.

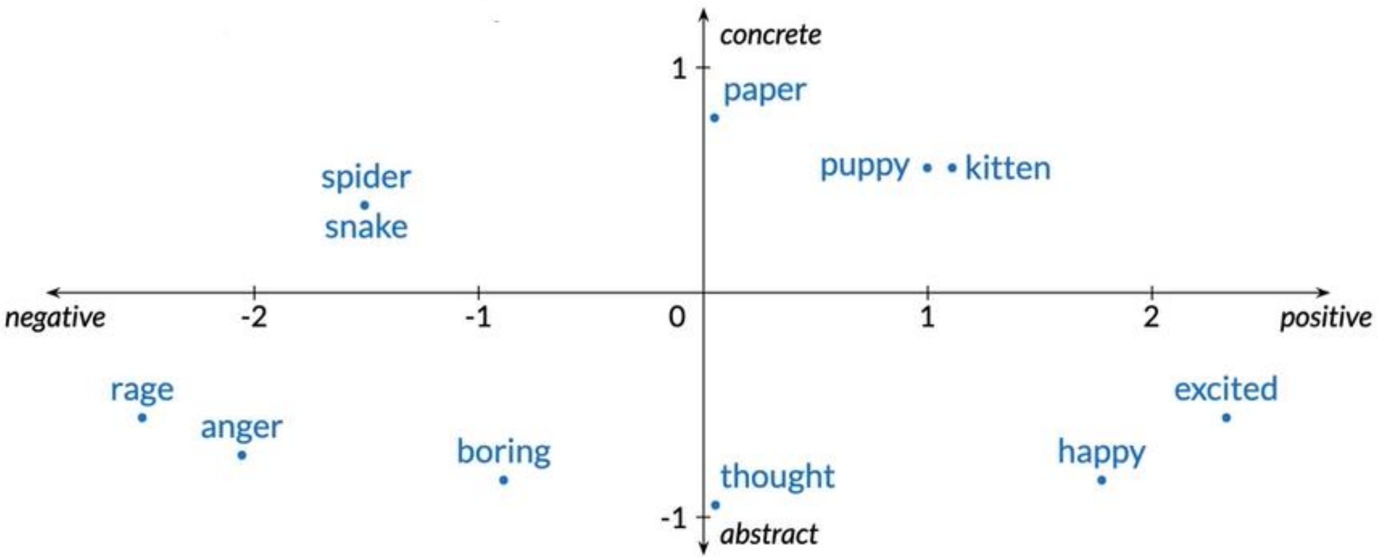
Now, you can say that's "happy" and "excited" are more similar to each other compared to with the word "paper"



Basic Word Representation – Word Embedding

You can extend this by adding a vertical number line, words that are higher on this line are more concrete physical objects. Whereas words lower on this line are more abstract ideas.

What you've done here is to represent the vocabulary of words with a small vector of length 2.



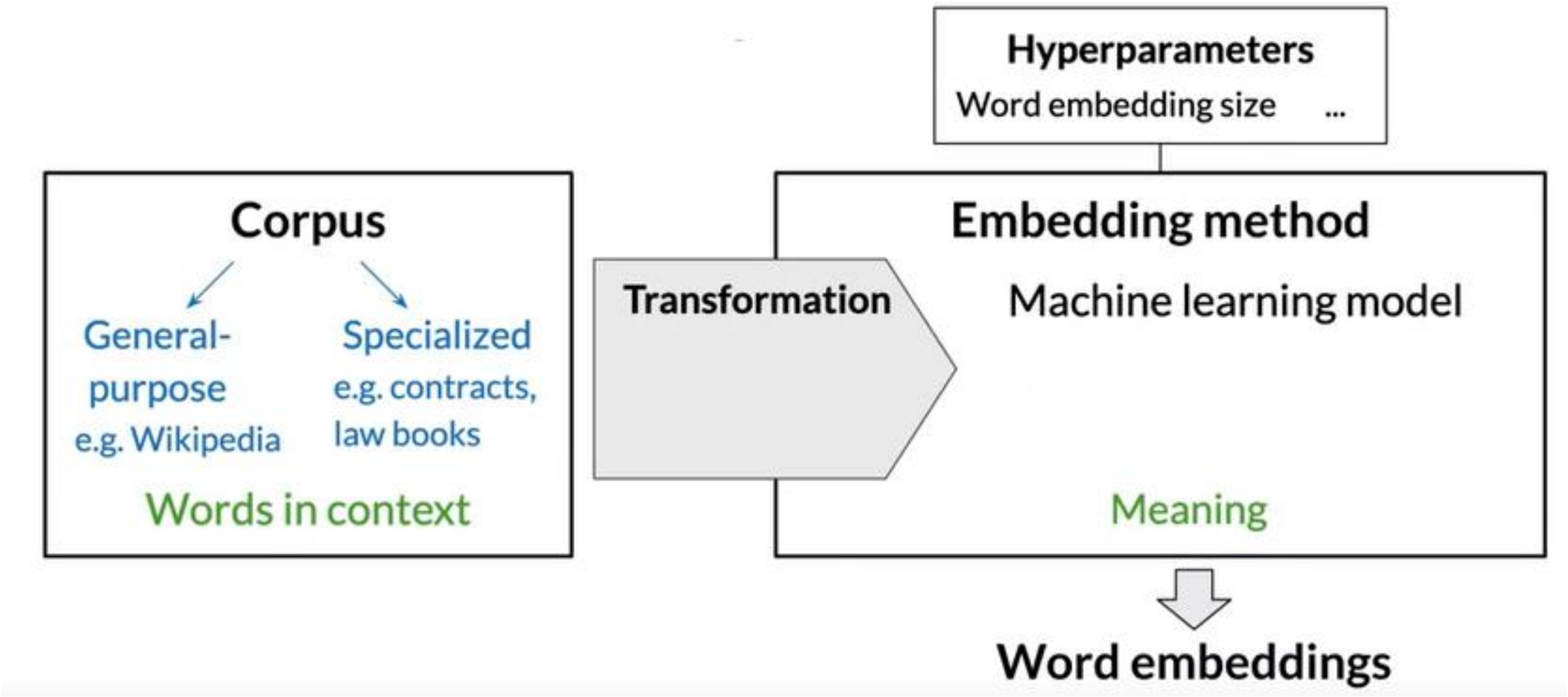
Basic Word Representation – Word Embedding

- Low dimension

- Embed meaning
    - e.g. semantic distance

    forest ≈ tree   forest ≠ ticket

Basic Word Representation – Word Embedding Process

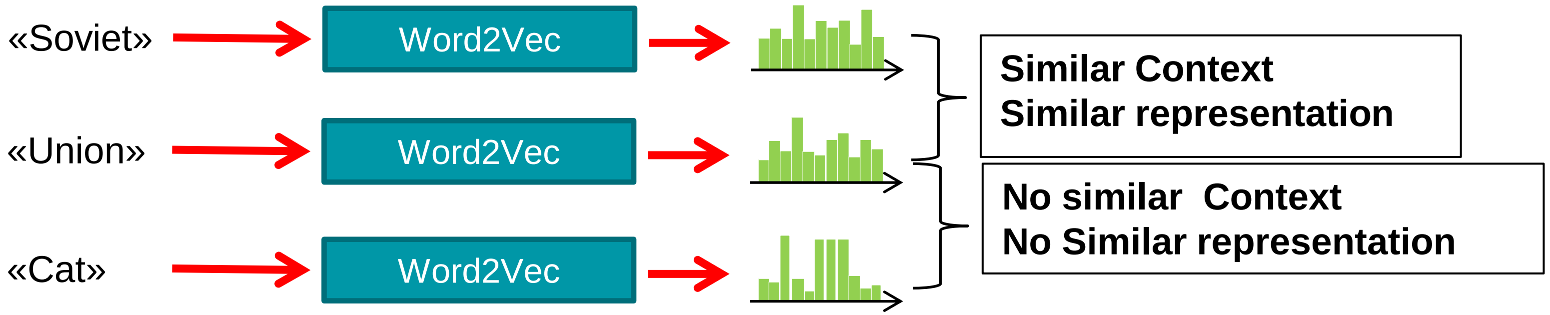


Word2Vec

Word Embedding

- Word2Vec network is a technique for building a rich semantic word embedding space (Google in 2013)
- Key idea: two words have similar word embedding representations if they have a similar contexts

- For example:

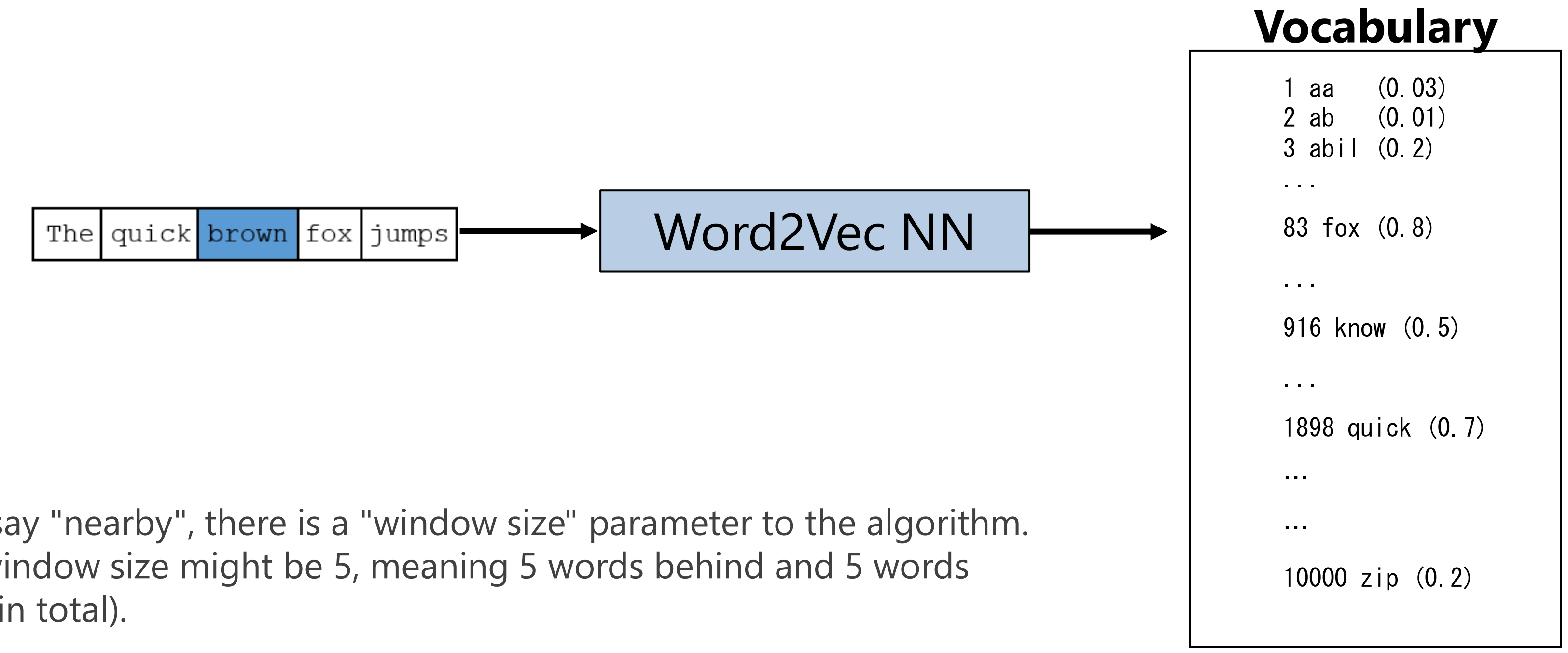


Word2Vec is based on Skip-Gram Neural Network Architecture.

The Fake Task

Let's suppose we have a vocabulary

Given a specific word in the middle of a sentence (the input word), the network is going to tell us the probability for every word in our vocabulary of being the "nearby word" that we chose.

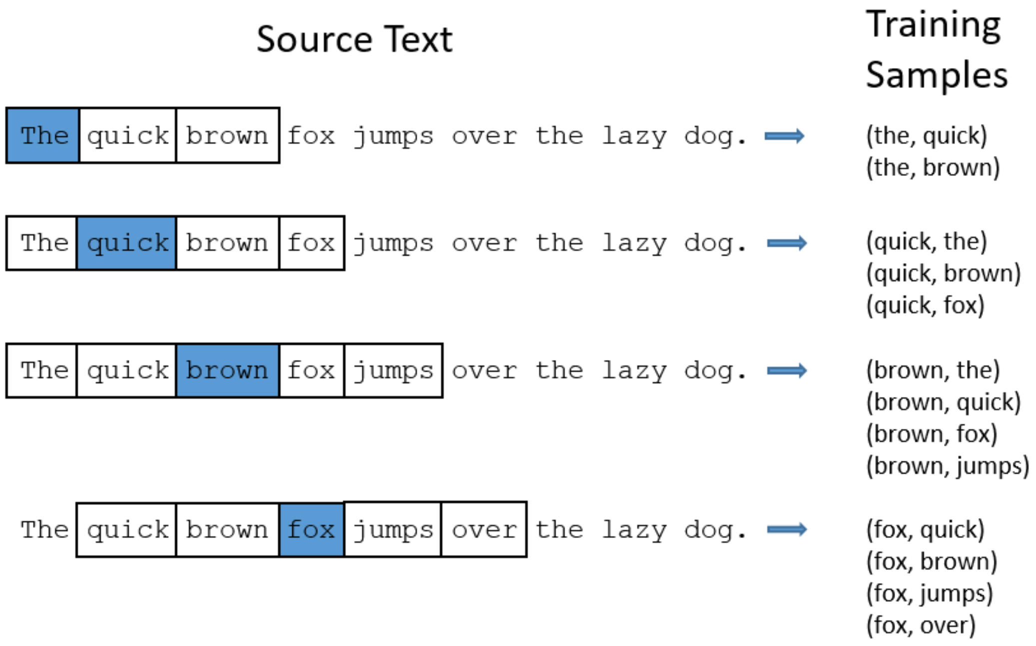


When we say "nearby", there is a "window size" parameter to the algorithm. A typical window size might be 5, meaning 5 words behind and 5 words ahead (10 in total).

Training Procedure

We’ll train the neural network to do this by feeding it word pairs found in our training documents.

Here Window Size: 2



For example, see the first sample: the first part “the” is input word, the “quick” is the target word.

Training Procedure

The network is going to learn the statistics from the number of times each pairing shows up.

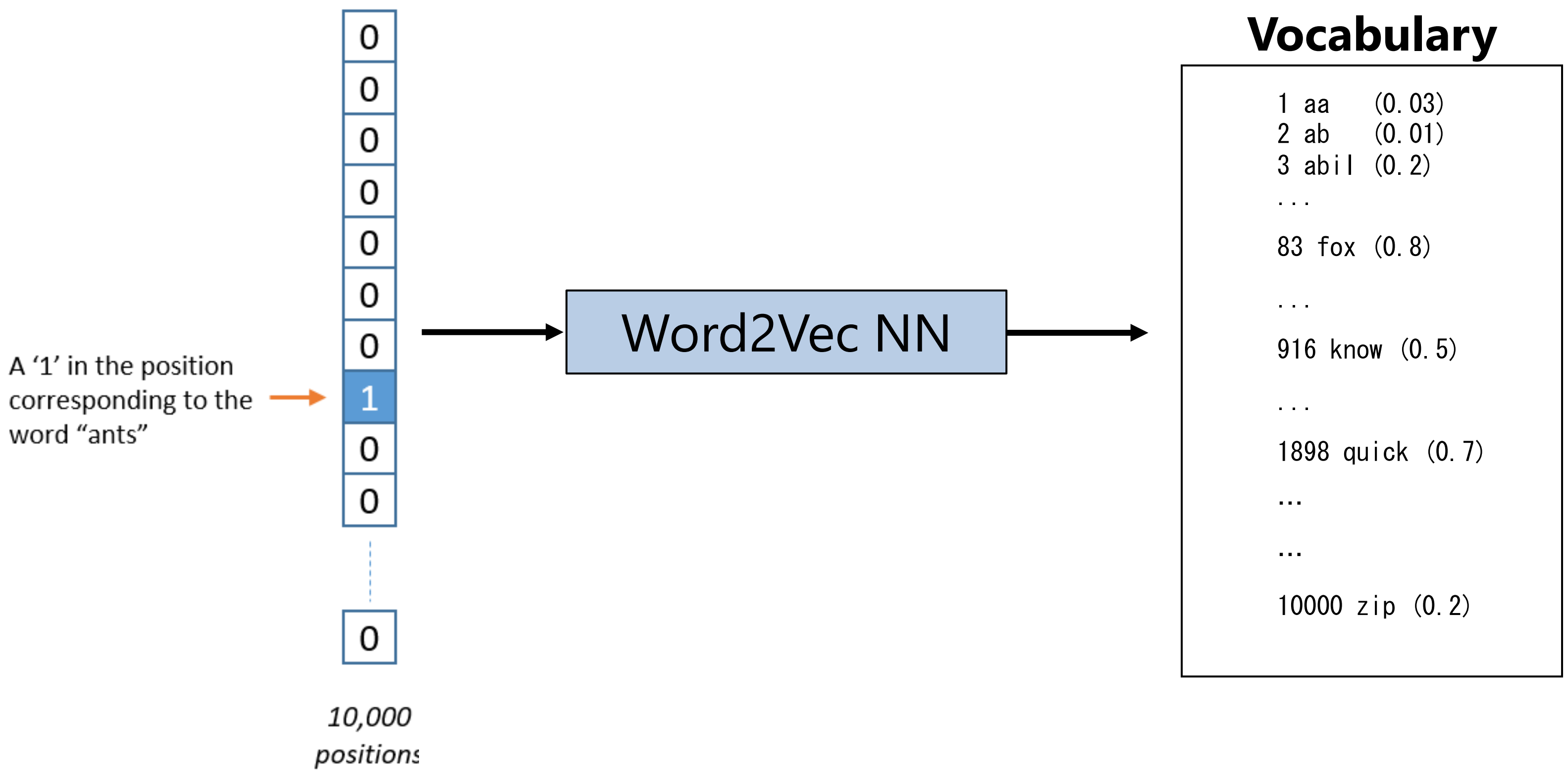
The network is probably going to get many more training samples of (“Soviet”, “Union”) than it is of (“Soviet”, “Cat”).

When the training is finished, **if you give it the word “Soviet” as input, then it will output a much higher probability for “Union” or “Russia” than it will for “Cat”.**

Model Details

Let's suppose we have a vocabulary of 10,000 unique words

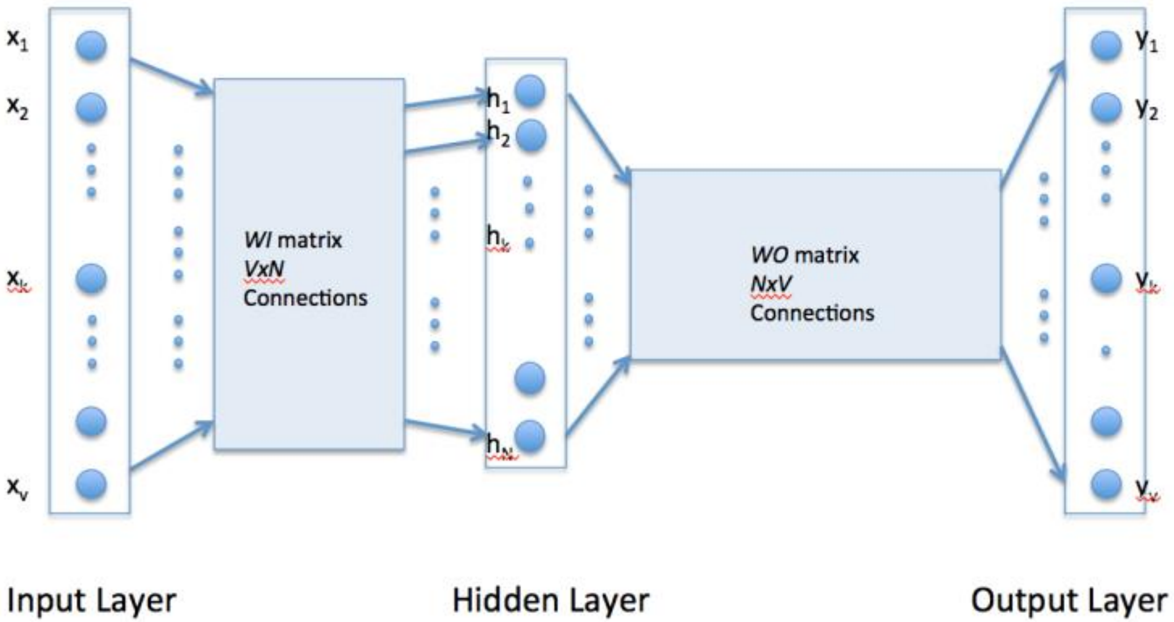
One-hot vector for the input word: e.g. "ants"



Model Details

The neural network is composed by an input layer, Hidden layer and Output layer

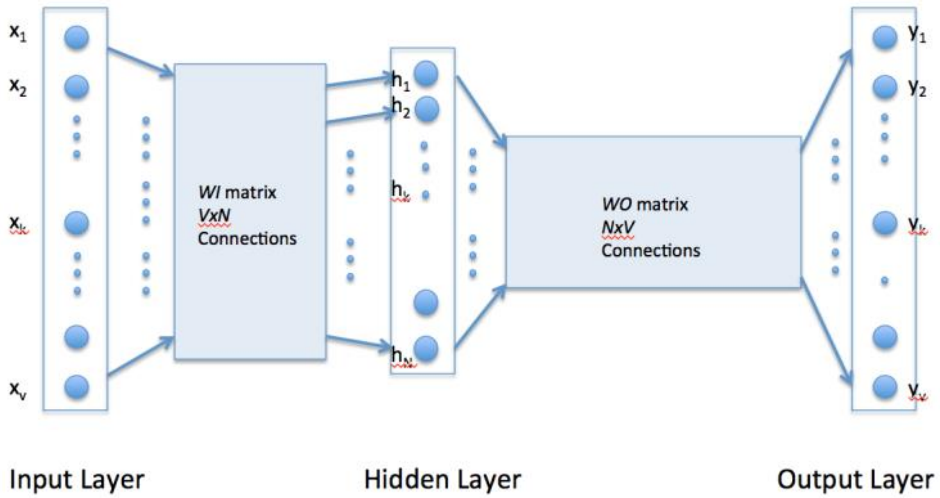
There is no activation function on the hidden layer neurons.



The Hidden Layer

For our example, we’re going to say that we’re learning word vectors with 300 features.

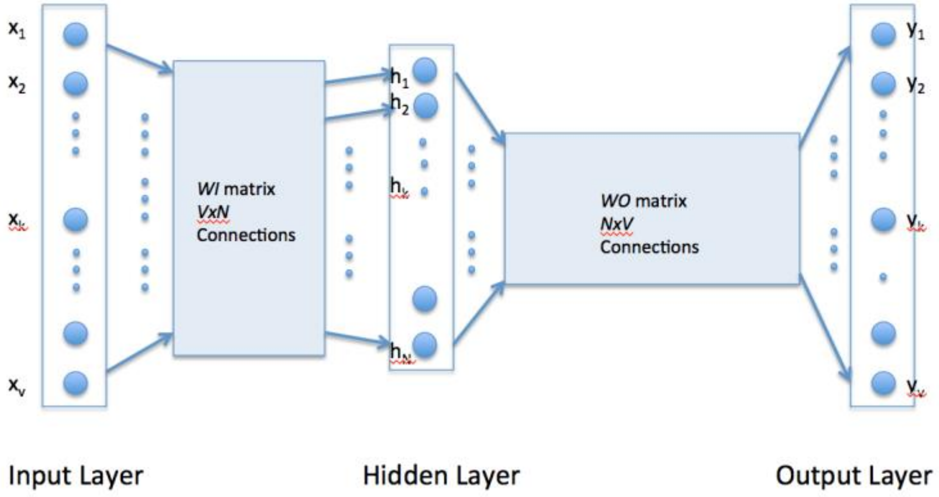
Hidden layer is represented by **a weight matrix** with 10,000 rows (one for every word in our vocabulary) and 300 columns (one for every hidden neuron).



If you look at the rows of this weight matrix, these are actually what will be our word vectors!

The Output Layer

The Output layer has the same number of neurons as the input



Softmax Function* is used in order to obtain probabilities for words in the output layer:

$$\hat{y}_k = \Pr(word_k | word_{context}) = \frac{\exp(y_k)}{\sum_{n=1}^{V} \exp(y_k)}$$

* We will see in the next classes

Toy Example

Corpus:

“the cat climbed a tree”, “the dog saw a cat”, “the dog chased the cat”

The vocabulary

Words	Index
a	1
cat	2
chased	3
climbed	4
dog	5
saw	6
the	7
tree	8

Toy Example

Let's assume dimension of Word2Vec representation 3

WI =

-0.094491	-0.443977	0.313917
-0.490796	-0.229903	0.065460
0.072921	0.172246	-0.357751
0.104514	-0.463000	0.079367
-0.226080	-0.154659	-0.038422
0.406115	-0.192794	-0.441992
0.181755	0.088268	0.277574
-0.055334	0.491792	0.263102

WO=

0.023074	0.479901	0.432148	0.375480	-0.364732	-0.119840	0.266070	-0.351000
-0.368008	0.424778	-0.257104	-0.148817	0.033922	0.353874	-0.144942	0.130904
0.422434	0.364503	0.467865	-0.020302	-0.423890	-0.438777	0.268529	-0.446787

Toy Example

Training example:

- The input word “cat” $X = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$
- The target word “climbed” $[0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$

The output of the hidden layer neurons:

$$H^T = X^T WI = [-0.490796\ -0.229903\ 0.065460]$$

The activation vector for the output layer neurons:

$$H^T WO = [0.100934\ \ -0.309331\ \ -0.122361\ \ -0.151399\ \ 0.143463\ \ -0.051262\ \ -0.079686\ \ 0.112928]$$

Toy Example

$H^T W O \; = \; [0.100934 \;\; -0.309331 \;\; -0.122361 \;\; -0.151399 \;\; 0.143463 \;\; -0.051262 \;\; -0.079686 \;\; 0.112928]$

Since we need probabilities for word in the output layer, we use the softmax function: $\hat{y}_k = \Pr(word_k | word_{context}) = \frac{\exp(y_k)}{\sum_{n=1}^{V} \exp(y_k)}$

The probabilities for eight words in the corpus are:

[0.143073   0.094925   0.114441   0.111166   0.149289   0.122874   0.119431   0.144800]

The target word "climbed" $[0\;0\;0\;1\;0\;0\;0\;0]^T$

Error = [0.143073   0.094925   0.114441   −0.888834   0.149289   0.122874   0.119431   0.144800]

Updating weights matrices using backpropagation.

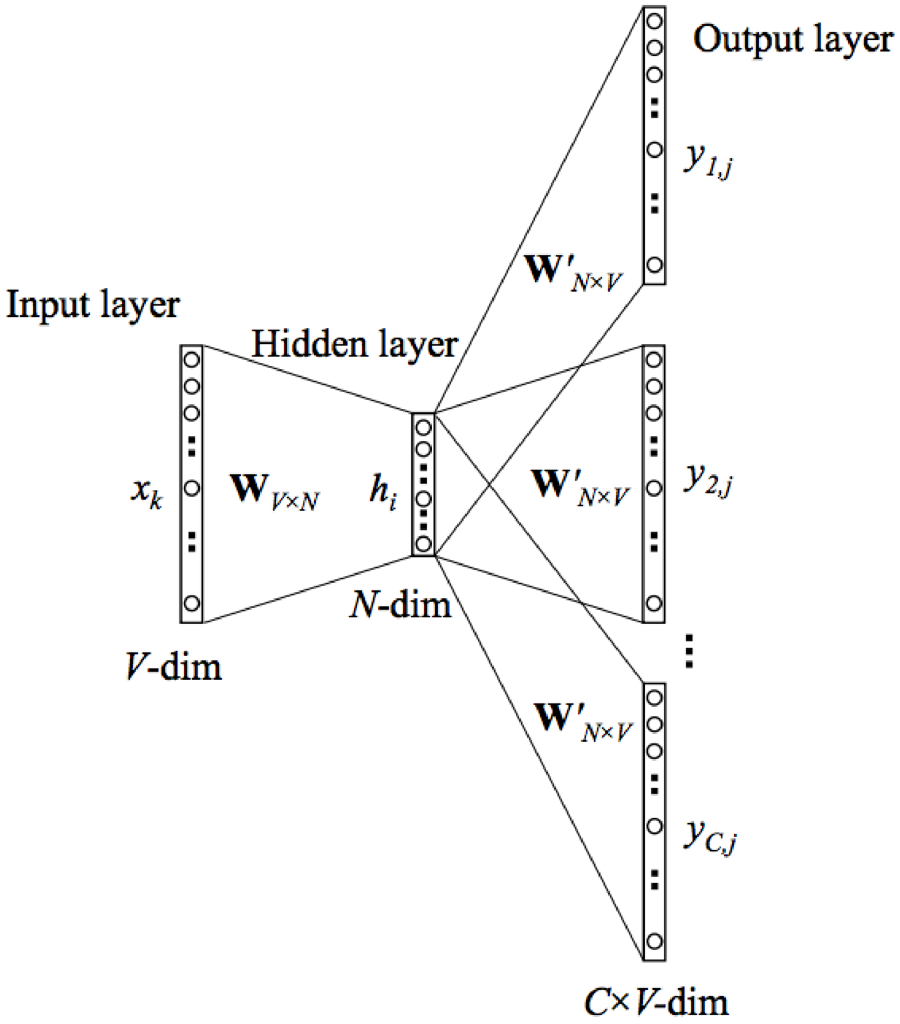
Intuition

If **two different words** have very similar “contexts” then our model needs to output very similar results for these two words.

And one way for the network to output similar context predictions for these two words is **if *the word vectors are similar***.

So, if two words have similar contexts, then our network is motivated to learn similar word vectors for these two words!

Full Skip-Gram

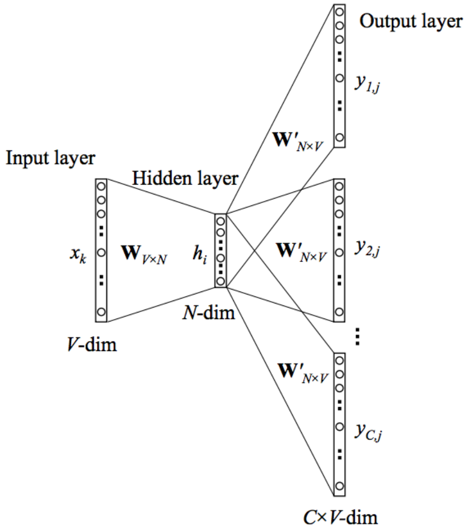


Full Skip-Gram

Let's suppose a windows of 1 words.

Training example:

- The input word "cat" $X = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$
- The target word1: "climbed" $T_{w1} = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$
- The target word2: "tree" $T_{w2} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$



The $y_{1,j} = y_{2,j}$, because they share the same weight matrices.

$E_1 = y_{1,j} - T_{w1}$ and $E_2 = y_{2,j} - T_{w2}$

$$E = E_1 + E_2$$

Updating weights matrices using backpropagation.

Huge Neural Network

The skip-gram model for Word2Vec is a large neural network.

If we have word vectors with 300 components and a vocabulary of 10,000 words, each weight matrix will have 300 x 10,000 = 3 Million weights!!!

Running gradient descent on a neural network that large is going to be slow.

It requires a huge amount of training data

Improvements for Word2Vec

There are three innovations introduced in this technique:

- Treating common word pairs or phrases as single “words” in their model.
- Subsampling frequent words to decrease the number of training examples.
- Modifying the optimization objective with a technique they called “Negative Sampling”, which causes each training sample to update only a small percentage of the model’s weights.

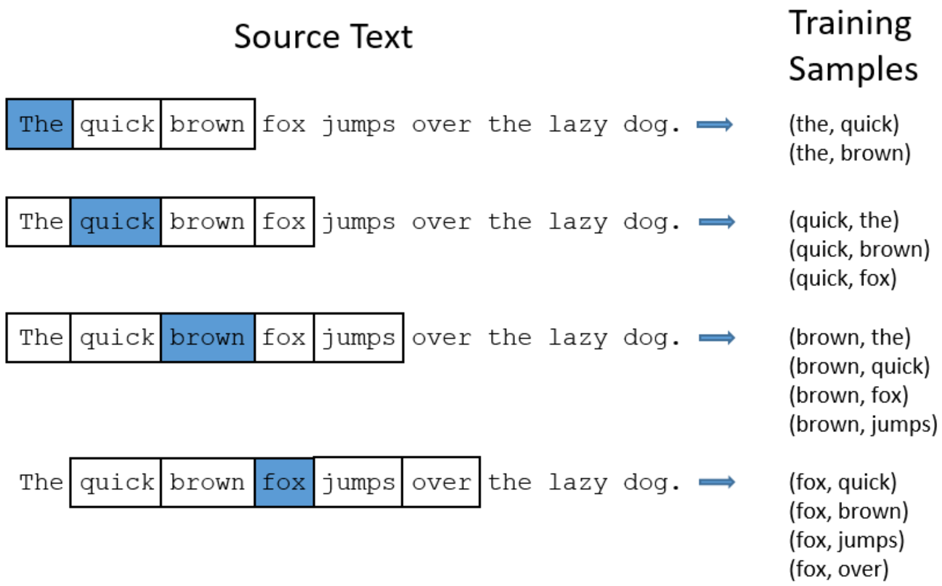
Word pairs and Phrases

The authors pointed out that a word pair like “Boston Globe” (a newspaper) has a much different meaning than the individual words “Boston” and “Globe”.

So it makes sense to treat “Boston Globe”, wherever it occurs in the text, as a single word with its own word vector representation.

Words that appear always together are merged.

Subsampling Frequent Words



There are two “problems” with common words like “the”:

(“fox”, “the”) doesn’t tell us much about the meaning of “fox”.

We will have many more samples of (“the”, …)

Frequent words are removed.

Negative Sampling

When training the network on the word pair (“cat”, “climbed”), recall that the “label” or “correct output” of the network is a one-hot vector. That is, for the output neuron corresponding to “quick” to output a 1, and for *all* of the other thousands of output neurons to output a 0.

With negative sampling, we are instead going to randomly select just a small number of “negative” words (let’s say 5) to update the weights for.

We will also still update the weights for our “positive” word (which is the word “climbed” in our current example).

Word2Vec Implementation

Word2vect (Google 2013)

- https://code.google.com/archive/p/word2vec/

Python

- High level interface
- https://radimrehurek.com/gensim/models/word2vec.html
- Parameters:
  - Size: this parameter is used to set the size or dimension for the word vector
  - Windows: size of context
  - Min_count = This parameter specifies the minimum word count needed across the courpus

Additional Material

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/