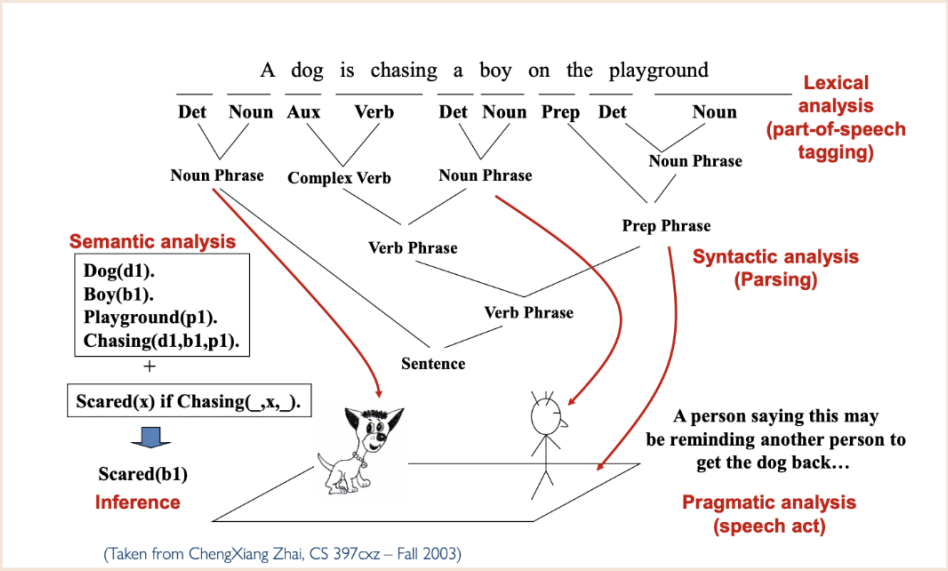
# * Text Representation & Embeddings

## » Natural Language Processing (NLP)




- NLP has difficulties because of ambiguity and common-sense reasoning

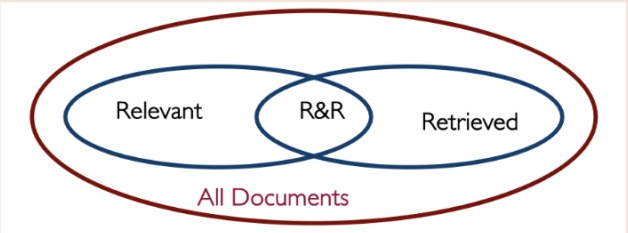
## » Information Retrieval:

- Information retrieval deals with the problem of locating relevant documents with respect to the user input or preference
- Typical systems: Online library catalogue, Online DMS
- Typical Issues: Management of unstructured documents
  Approximate search
  Relevance
- Two modes of Text Access:

1) Pull Mode (search engine)
- Users take initiative
- Ad-hoc info needed (amazon)

2) Push Mode (recommender system)
- Systems take initiative
- Stable info need or system has good knowledge about user's need.

- **Document Selection** (keyword-based retrieval)
  - Query defines a set of requisities
  - Only the documents that satisfy the query are returned
  - Typical approach: Boolean Retrieval Model
- **Document Ranking** (similarity-based retrieval)
  - Documents are ranked on the basis of their relevance w.r.t. to the user query
  - For each document a "degree of relevance" is computed w.r.to the query
  - Typical approach: Vector Space Model




$$\text{precision} = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{\{\text{Retrieved}\}}$$

$$\text{recall} = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{\{\text{Relevant}\}}$$

$$\text{F-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

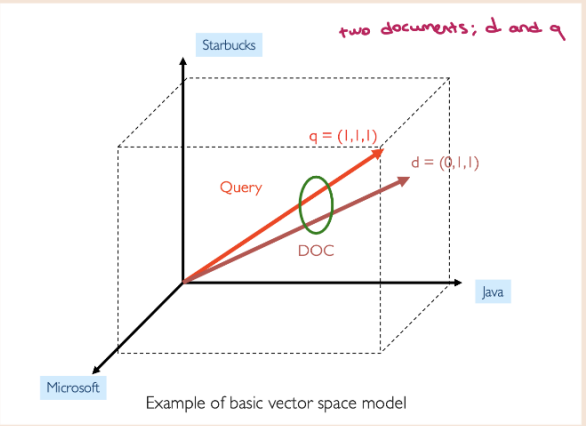
# » Document Similarity

## • Bag-of-Words Representation:

- Text is represented as a **table** → columns (features) identify **vocabulary words**; rows (instances) identify **documents**
- Values can represent: presence of word, frequency of word, ...
- It is a sparse representation since a document usually contains much fewer words than the entire vocabulary → most values are zero

## • Vector Space Model:

- A document is vectors in high-dim space corresponding to all keywords
- Relevance is measured with an appropriate similarity measure
- Issues:
  How to select keywords to capture "basic concepts"?
  How to assign weights to each term?
  How to measure the similarity?



Example of basic vector space model

- Simple Similarity Function:

$$f(q,d) = q_1 d_1 + \ldots + q_N d_N$$

→ $q_i$ and $d_i$ represents the presence or absence of word $w_i$

- Ex:

q="news about presidential campaign"

d1 … news about …
d3 … news of presidential campaign …

vocabulary

V= {news, about, presidential, campaign, food .... }

q= (1, 1, 1, 1, 0, …)
d1= (1, 1, 0, 0, 0, …)
f(q,d1)= 1*1+1*1 +1* 0+1*0+0*0 + …= 2
d3= (1, 0, 1, 1, 0, …)
f(q,d3)= 1*1+1*0 +1*1 +1*1+0*0+ … = 3

q="news about presidential campaign"

d1 … news about … f(q,d1)=2
d2 … news about organic food campaign… f(q,d2)=3
d3 … news of presidential campaign … f(q,d3)=3
d4 … news of presidential campaign … … presidential candidate … f(q,d4)=3
d5 … news of organic food campaign… campaign…campaign…campaign… f(q,d5)=2

* qi: vocabulary'de varsa 1 yoksa 0
* Birden fazla olmasını gözardı ediyor

# » Document Representation:

## • Term Frequency

- To take into account the number of times a word appears in the document $d_i$ and $q_i$ → now represent the # of times word $w_i$ appears in the document.

- Ex:

q="news about presidential campaign"

d2 … news about organic food campaign… f(q,d2)=3
q= (1, 1, 1, 1, 0, …)
d2= (1, 1, 0, 1, 1, …)

d3 … news of presidential campaign … f(q,d3)=3
q= (1, 1, 1, 1, 0, …)
d3= (1, 0, 1, 1, 0, …)

d4 … news of presidential campaign … … presidential candidate … f(q,d4)=4!
q= (1, 1, 1, 1, 0, …)
d4= (1, 0, 2, 1, 0, …)

q="news about presidential campaign"

d2 … news about organic food campaign…
d3 … news of presidential campaign …

V= {news, about, presidential, campaign, food .... }

q= (1, 1, 1, 1, 0, …)
d2= (1, 1, 0, 1, 1, …)
f(q,d2)<3

q= (1, 1, 1, 1, 0, …)
d3= (1, 0, 1, 1, 0, …)
f(q,d3)>3

"presidential" and "about" weight the same in the document but the former is more relevant and surprising to be found in a document.

! Matching "presidential" is more relevant than matching "about" → in our example weights are same → We need to penalize words that are less informative ("about") and the ones that are more informative ("presidential")

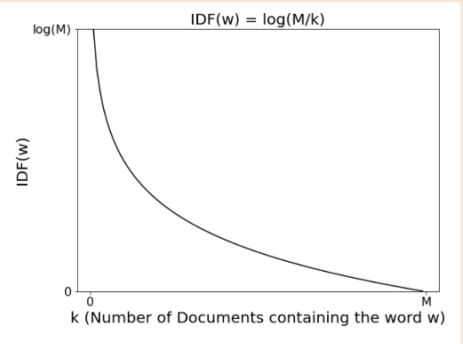
## Inverse Document Frequency (IDF)

- It measures how much info the word provides
- Penalizes the words that frequently occur in many documents

$$IDF(w) = \log \frac{M}{k}$$

→ (k) # of documents in which w appears
→ (M) # of documents

* when w might not in the corpus → division by zero
use adjusted version of formula with (k+1)




## >> Document Similarity

- We combine Term Frequency (TF) with Inverse Document Freq. (IDF) to build a representation of documents that we can use to compute document similarity (or distance). Given d in the corpus containing M documents, vector elements di are;

$$d_i = TF(w_i, d) \log(M/k_i)$$

→ we can use vectors to compute distance (e.g., euclidean) or similarity (e.g., cosine similarity)

## >> Text Data Preprocessing

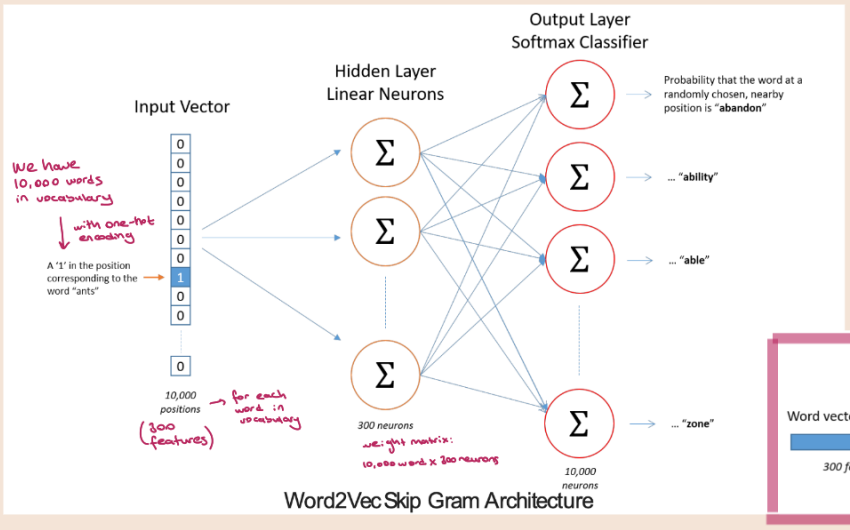
- Remove non-content related info (e.g. <HTML> tags)
- Lowercase the text (can loose info: 'WHO' vs. 'who')
- Remove punctuation (noktalama)
- Eliminate stop words, (e.g. "a", "the", "along"...)
- Word stemming / Lemmatization: Different words that share common prefix are simplified (for ex: "computer", "computing", "computerize" are replaced by "comput")

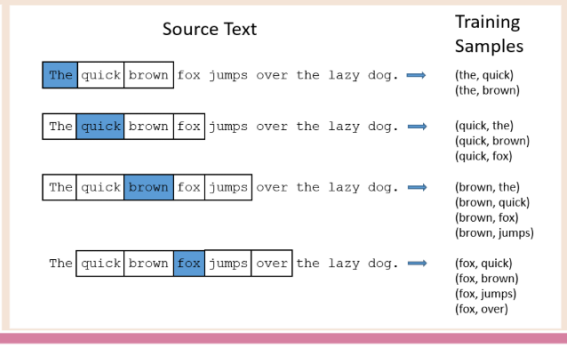
## >> Word Embeddings

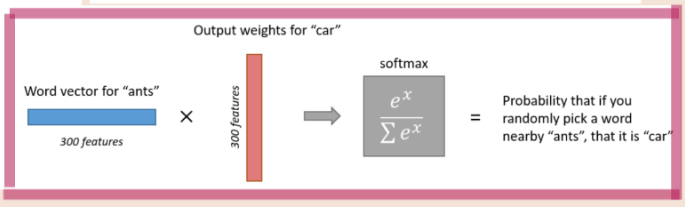
- In Bag-of-Word → words are represented by one position in huge vector (context info or ordering not used)
- In Word-Embeddings → dense representation of words rather than documents
- Stores each word as a point in continious space
- A word is represented by a vector of fixed # of dimensions
- Embeddings are supervised models:
  - Trained to predict missing word based on surrounding context
  - Not many words fit, just those that are semantically related each other
- word embeddings can be viewed as multi-class classification problem
- To predict: inputs are words in current context
  target is missing word from the sequence
  ] problem has massive parameter space

## • Word2Vec :

- It employs a skip gram neural network architecture.
- Trains a simple NN with a single hidden layer to, given a specific word in the middle of a sentence (input word), predict the probability for every word in our vocabulary of being the "nearby word" that we chose
- The output vector is a probability distribution

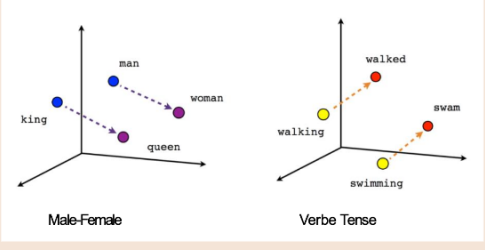


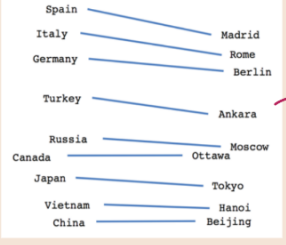




## » Properties of Word Embeddings

1) Translation in space is meaningful
2) Semantics are additive, neighbors in space are semantically related
3) Induces relationships between words such as; part-of-speech, type-of, geographic relationships




→ country & capital

## » Subword Embeddings

- Word embeddings work well when vocabulary is fixed (no new words in test set)
- However, we don't have embedding on unseen word
- Fasttext:
  - Split words into character sequences
  - Learns embeddings for character n-grams
  - Combines the embeddings to form words

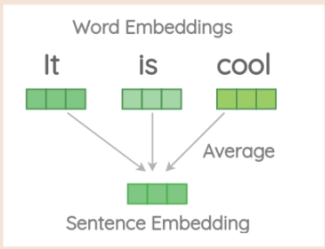
## » Application of Word Embeddings

- Word Similarity: Edit Distance, WordNet, Porter's Stemmer, Lemmatization
- Machine Translation
- Part-of-Speech and Named Entity Recognition
- Relation Extraction
- Sentiment Analysis

## » Sentence Embeddings

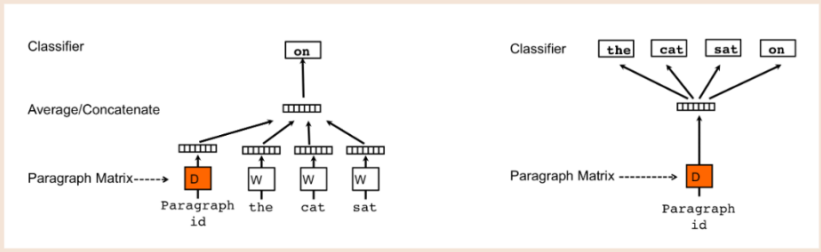
### • Naive Approach:

- Pros: Easy to implement
  Decent performance for basic tasks
- Cons: Embeddings don't take into account the context of words
  Words order does not effect the embeddings




### • Doc2Vec

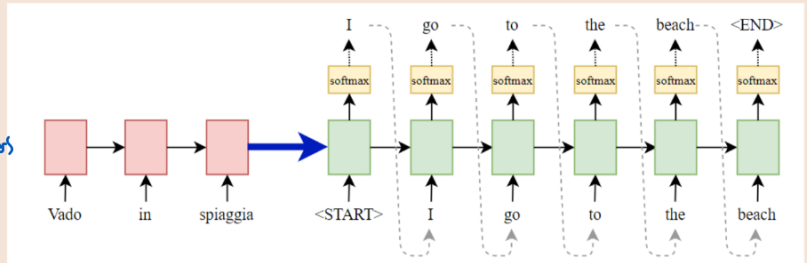
- Extension of Word2Vec for fixed-length documents
- Applications: classification and retrieval of documents with standard structure.



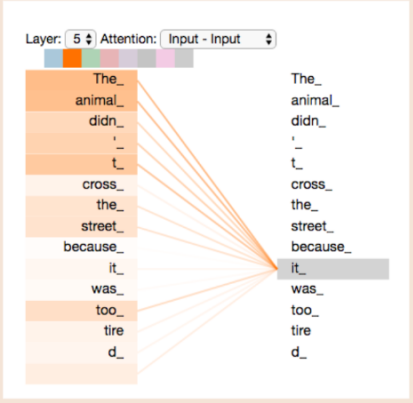

### • Seq2Seq

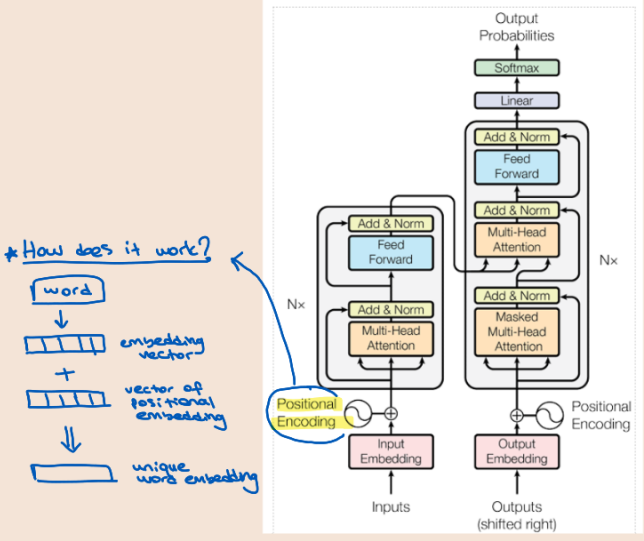
- Challenges:
  - Vanishing gradient issue for long sentences
  - Sequential implementation

transformers ile çözüldü

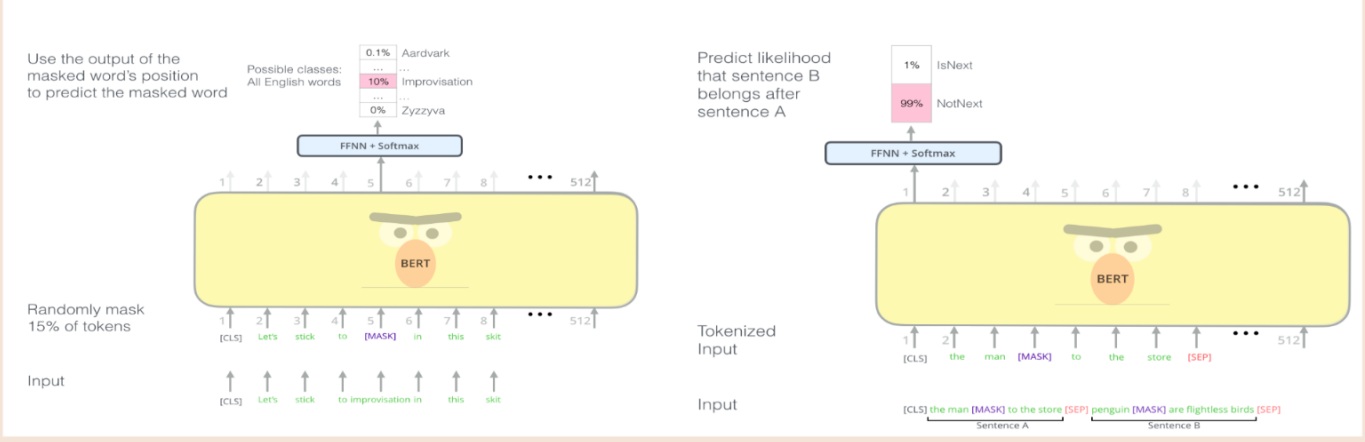


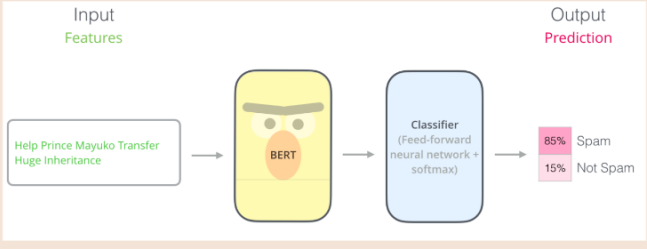

### • The Breakthrough of Transformers




### • BERT



## ⇉ Entity Embeddings for Categorical Variables

### • One-Hot-Encoding & Lack of Meaningful Relations

- One-Hot encoding is used to preprocess categorical data for algorithms that work on numerical values only
- It can generate massive amount of data (vocabulary of 100,000 terms would generate 100,000 binary variables)
- Embeddings, translate large sparse vectors into a lower-dimensional space that preserves semantic relationships
- Entity embeddings map categorical variables into Euclidean spaces
- The mapping is learned by a neural network using a supervised training process
- It reduces memory usage and speeds up neural networks compared with one-hot encoding
- Maps similar values close to each other in the embedding space possibly highlighting intrinsic properties of the categorical variables.