

NATURAL LANGUAGE PROCESSING

LESSON 8 : LEXICAL SIMILARITY

OUTLINE

- **Lexical vs. Semantic Similarity**
- **Similarity**
 - Levenstein Distance
 - Jaccard Similarity
 - Cosine Similarity
- **Vector Space Model**
 - Binary Weighting
 - Term Frequency (TF)
 - TFIDF

LEXICAL vs SEMANTIC SIMILARITY

Lexical similarity just deals with word's position of character while semantic similarity deals with mean.

	Lexical	Semantic
su-şu	0.5	0
su-bardak	0	0.7

LEXICAL (WORD) SIMILARITY

Word similarity is the similarity of words in terms of form or meaning.

- Some methods for finding word similarity;
 - Levenshtein Distance,
 - Jaccard Index,
 - Cosine Similarity.

Levenstein measures the distance (dissimilarity) of two string. While the distance increases, their similarity decreases. But Jaccard and Cosine measure the similarity of strings.

LEVENSHTEIN DISTANCE

The **Levenshtein distance** is a string metric for measuring the difference between two sequences.

Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

LEVENSHTEIN DISTANCE

«relevant» to «elephant»

(1) Remove 'r' => elevant

(2) Add 'p' => elepvant

(3) Put 'h' instead of 'v' => elephant

Levenstein distance is 3

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

LEVENSHTEIN DISTANCE

Some usage areas of Levenstein distance:

- OCR (Optical Character Recognition) apps
- Spell checking

We can find the closest similarities of the words.

JACCARD INDEX

Jaccard index is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets

Jaccard Index = $\frac{\text{the characters in both words}}{\text{the characters in either word}} \times 100$

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \times 100$$

JACCARD INDEX

Its algorithm contains these steps:

1. Count the number of intersected characters which are shared by both words
2. Count the total number of all characters in both sets (shared and un-shared)
3. Divide the number of shared characters by the total number of characters
4. Multiply the number found in (3) by 100

COSINE SIMILARITY

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle.

$$V1 \text{ and } V2 : \text{vectors} \qquad \text{Cos}(V1, V2) = \frac{\vec{v_1} \cdot \vec{v_2}}{\|V1\| * \|V2\|}$$

COSINE SIMILARITY

V1 and V2 : vectors $\text{Cos}(V1, V2) = \frac{\vec{v_1} \cdot \vec{v_2}}{\|V1\| * \|V2\|}$

Assume V1: [3 0 4] and V2 : [0 0 1]

$$\|V1\| = \sqrt{3^2 + 0^2 + 4^2} = 5$$

$$\|V2\| = \sqrt{0^2 + 0^2 + 1^2} = 1$$

$$\text{Cos}(V1, V2) = \frac{3*0+0*0+4*1}{5*1} = 0.8$$

VECTOR SPACE MODEL

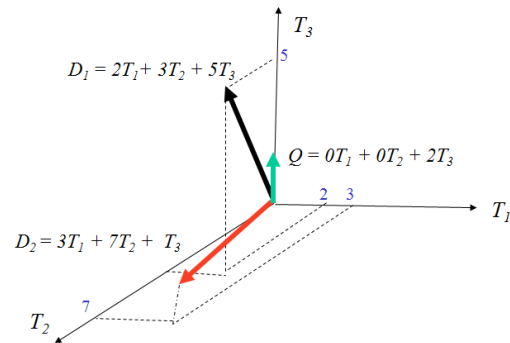
«Keliem beznerlii yötnemlerini örgendim.»

If we don't convert the sentences to numeric form, computer can't understand what does it mean as we understand the upper sentence. To solve this problem we use dimensional vectors instead of sentences.

VECTOR SPACE MODEL

The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental for scoring documents on a query.

Every unique terms in document represents a dimension in vector space.



SENTENCE TO VECTOR

Each value in a vector is the weight of its corresponding term. We can calculate weights by using one of followings:

1. Binary,
2. Term Frequency (TF),
3. Term Frequency - Inverse Document Frequency (TF-IDF).

BINARY WEIGHTING

If a document includes a term, the weight of that term for the document is 1, otherwise 0.

Document1 : Bir kalem ve bir defter aldım.

Document2: Bir kitap aldım.

VECTOR: [aldım bir defter kalem kitap ve]

(union of words in all documents)

D1: [1 1 1 1 0 1]

D2: [1 1 0 0 1 0]

TERM FREQUENCY (TF) WEIGHTING

If a document includes a term, the weight of that term for the document is TF (# of term), otherwise 0.

Document1 : Bir kalem ve bir defter aldım.

Document2: Bir kitap aldım.

VECTOR: [aldım bir defter kalem kitap ve]

(union of words in all documents)

D1: [1 2 1 1 0 1]

D2: [1 1 0 0 1 0]

TF-IDF WEIGHTING

TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

TF represent how many times a term exist in a document, IDF means that a term exists in how many document.

$$\text{TF-IDF} = \text{TF}_{\text{norm}} * \text{IDF}$$

$$\text{TF}_{\text{norm}} = \text{TF} / \text{TF}_{\text{max}}$$

$$\text{IDF} = 1 + \ln \left(\frac{\text{total \# of documents}}{\text{\# of documents including the actual word}} \right)$$

TF-IDF WEIGHTING

Document1 : Bir kalem ve bir defter aldım.

Document2: Bir kitap aldım.

VECTOR : [aldım bir defter kalem kitap ve]

TF(D1): [1 2 1 1 0 1]

TF(D2): [1 1 0 0 1 0]

TF_{norm}(D1):[0.5 1 0.5 0.5 0 0.5]

TF_{norm}(D2):[0.5 0.5 0 0 0.5 0]

TF-IDF WEIGHTING

$$\text{IDF}_{\text{aldım}} = \text{IDF}_{\text{bir}} = 1 + \ln(2/2) = 1$$

$$\text{IDF}_{\text{kalem}} = \text{IDF}_{\text{defter}} = \text{IDF}_{\text{kitap}} = \text{IDF}_{\text{ve}} = 1 + \ln(2/1) = 1.693$$

$$\text{IDF}: [1 \ 1 \ 1.693 \ 1.693 \ 1.693 \ 1.693]$$

$$\text{TF} * \text{IDF} (\text{D1}): [0.5 \ 1 \ 0.85 \ 0.85 \ 0 \ 0.85]$$

$$\text{TF} * \text{IDF} (\text{D2}): [0.5 \ 0.5 \ 0 \ 0 \ 0.85 \ 0]$$

DISADVANTAGE OF VECTOR SPACE MODEL

One of the biggest disadvantages of the vector space model is that it becomes useless when the text size is long. In general, we deal with a very high quantity of data as a result of such feature vectors.

For example, we took 62,000 comments from a website containing comments of IMDB movie reviews, there are more or less 160,000 words. So this means that the dimension of a vector is 160,000. This feature vector is almost 37GB (for 4B int vector) and there will be no free memory space in PC to process that data.

DISADVANTAGE OF VECTOR SPACE MODEL

This dimension problem can be reduced by

- Using **lemmas of the words** in text.


We can use «ağaç» for both «ağacı» and «ağaçlar».

- Get rid of stop and functional words, or use some special keywords
- 

SENTENCE SIMILARITY


We find sentence similarities by combining word similarities.

Where can «sentence similarity» be used?

- Google search engine,
 - Ad-sense,
 - Finding precedent case.
- 

SENTENCE SIMILARITY

Simple steps for sentence similarity:

- Get rid of stop and functional words
 - Use lemmas/stems instead of words
 - Convert strings to numbers by Vector Space Model
 - Measure similarity by a Similarity Function
- 


EXAMPLE

Q: «Metin benzerliğini bulmak için TFIDF ve kosinüs benzerliğini kullanacağız.»

D1: « DDİ bilgisayar biliminin bir konusudur.»

D2: «Metin benzerliğini bazı DDİ yöntemleri sayesinde buluruz.»

Stop Words: «bir», «için», «ve», «sayesinde», «bazı».



SOLUTION WITH JACCARD SIMILARITY

Q: «Metin benzerliğini bulmak için TFIDF ve kosinüs benzerliğini kullanacağız.»

D1: « DDİ bilgisayar biliminin bir konusudur.»

D2: «Metin benzerliğini bazı DDİ yöntemleri sayesinde buluruz.»

Stop Words: «bir», «için», «ve», «sayesinde», «bazı».

of shared words(Q,D1) = 0 $J(Q,D1)=0/11 = 0$

of shared words(Q,D2) = 2 $J(Q,D2)=2/11 = 0.18$

of total words = 11

According to Jaccard Similarity, the query is similar to second document with 18% similarity rate.

SOLUTION WITH COSINE SIMILARITY

Q: «Metin benzerliğini bulmak için TFIDF ve kosinüs benzerliğini kullanacağız.»

D1: «DDİ bilgisayar biliminin bir konusudur.»

D2: «Metin benzerliğini bazı DDİ yöntemleri sayesinde buluruz.»

	DDİ	Bilgisayar	Bilim	Konu	Metin	Benzerlik	Bulmak	TFIDF	Kosinüs	Kullanmak	Yöntem
TF	0	0	0	0	1	2	1	1	1	1	0
IDF	1.41	2.1	2.1	2.1	1.41	1.41	1.41	2.1	2.1	2.1	1.1
TF _{norm}	0	0	0	0	0.5	1	0.5	0.5	0.5	0.5	0
TF*IDF	0	0	0	0	0.7	1.41	0.2	1.05	1.05	1.05	0

For Query (Q) sentence

SOLUTION WITH COSINE SIMILARITY

Q: «Metin benzerliğini bulmak için TFIDF ve kosinüs benzerliğini kullanacağız.»

D1: «DDİ bilgisayar biliminin bir konusudur.»

D2: «Metin benzerliğini bazı DDİ yöntemleri sayesinde buluruz.»

	DDİ	Bilgisayar	Bilim	Konu	Metin	Benzerlik	Bulmak	TFIDF	Kosinüs	Kullanmak	Yöntem
TF	1	1	1	1	0	0	0	0	0	0	0
IDF	1.41	2.1	2.1	2.1	1.41	1.41	1.41	2.1	2.1	2.1	1.1
TF _{norm}	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0
TF*IDF	0.7	1	1	1	0	0	0	0	0	0	0

For Document 1 (D1) sentence

SOLUTION WITH COSINE SIMILARITY

Q: «Metin benzerliğini bulmak için TFIDF ve kosinüs benzerliğini kullanacağız.»

D1: «DDİ bilgisayar biliminin bir konusudur.»

D2: «Metin benzerliğini bazı DDİ yöntemleri sayesinde buluruz.»

	DDİ	Bilgisayar	Bilim	Konu	Metin	Benzerlik	Bulmak	TFIDF	Kosinüs	Kullanmak	Yöntem
TF	1	0	0	0	1	1	1	0	0	0	1
IDF	1.41	2.1	2.1	2.1	1.41	1.41	1.41	2.1	2.1	2.1	1.1
TF _{norm}	0.5	0	0	0	0.5	0.5	0.5	0	0	0	0.5
TF*IDF	0.5	0	0	0	0.7	0.7	0.7	0	0	0	0.5

For Document 2 (D2) sentence

SOLUTION WITH COSINE SIMILARITY

[DDİ Bilgisayar Bilim Konu Metin Benzerlik Bulmak TFIDF Kosinüs Kullanmak Yöntem]

Q: [0 0 0 0 0.7 1.41 0.2 1.05 1.05 1.05 0]


D1: [0.7 1 1 1 0 0 0 0 0 0]

D2: [0.5 0 0 0 0 0.7 0.7 0.7 0 0 0.5]

$\text{Cos}(Q, D1) = 0$

$\text{Cos}(Q, D2) = 0.55$

According to Cosine Similarity, the Query (Q) is similar to the second document with 55% value.



WHAT ABOUT LEVENSHTein?

Can we measure document similarities by using Levenshtein distance?

If so, how can we do it?



<https://www.twinword.com/api/text-similarity.php>

Free Tool & API Demo

First, input some text:

When will you come to us to drink tea?

Second, input some other text to compare with:

Whenever you want, you can come.

Evaluate

Results:

```
{
  "similarity": 0.50123574131851,
  "value": 100247.1482637,
  "version": "4.0.0",
  "author": "twinword inc.",
  "email": "feedback@twinword.com",
  "result_code": "200",
  "result_msg": "Success"
}
```

Do not confuse similarity
with a question
answering system.

Here we measured
«Lexical Similarity».

PROJECT

Preparing a bot for a FAQ (Frequently Asked Questions) database.