

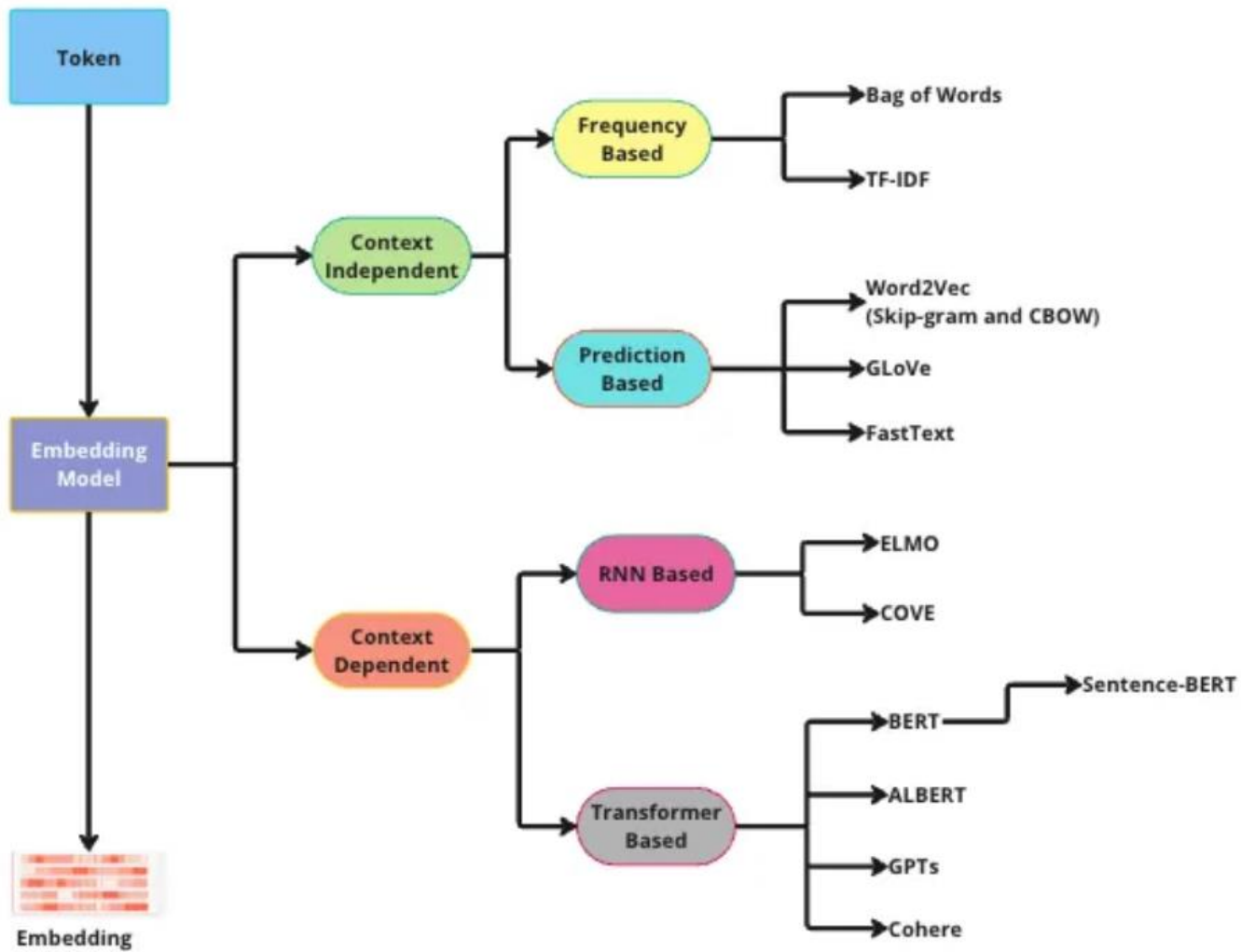


KELİME GÖSTERİLİMLERİ

(Word Representation – Word Embeddings)

Prof. Dr. Banu DİRİ





- Kelime, cümlede kullanımına göre farklı anlamlar kazanabilir
- Anlamsal bilginin çıkarılması metinlerin işlenmesinde önemlidir
- Kelimelerin işlenebilir formattaki haline **Kelime Gösterilimi** denir

Kelime Gösterilimi

❖ İçerikten Bağımsız

- Frekans Tabanlı Kelime Gösterilimi (*Count Vector, Bag of Words, Tf-Idf*)
- Tahminleme Tabanlı Kelime Gösterilimi (*Word2Vec, GloVe, FastText*)

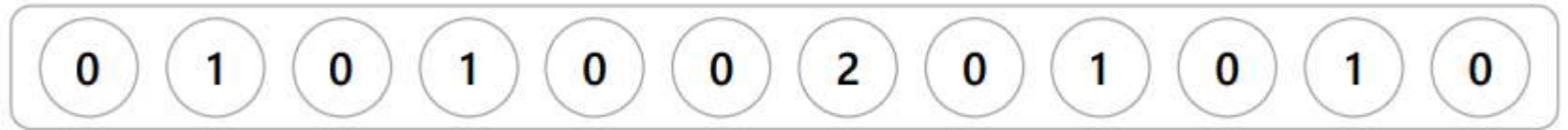
❖ İçerik Bağımlı

- *RNN Tabanlı (ELMO, COVE)*
- *Transformer Tabanlı (BERT, ALBERT, Cohere, GPTs)*

Bilimin değeri işleri karmaşık hale getirmek değil, özündeki basitliği bulmaktır.
Frank Seide

Doğal Dil İşlemede kelimeleri bilgisayarların anlayabilmesi için sayısal değerler haline getiririz.

Vektör Uzay Modelleri (Vector Space Models) bir öğeyi-item (ör:kelime) sayı vektörü olarak gösterir.



❖ Tahminleme yaklaşımları ile kelimeler arasındaki anlamsal ilişkiler çıkarılabilir

Erkek → Kadın | Kral → ? (Kraliçe)

❖ Kelime vektörleri arasındaki mesafe (*Cosine Similarity*)

❖ Kelime Vektörlerinin öğrenilmesi eğiticişiz öğrenmeye girer

❖ Her kelime için belirli sayıda özellik öğrenilir

❖ Her özellik bir anlamsal bilgi taşır

1- One-hot Gösterimi

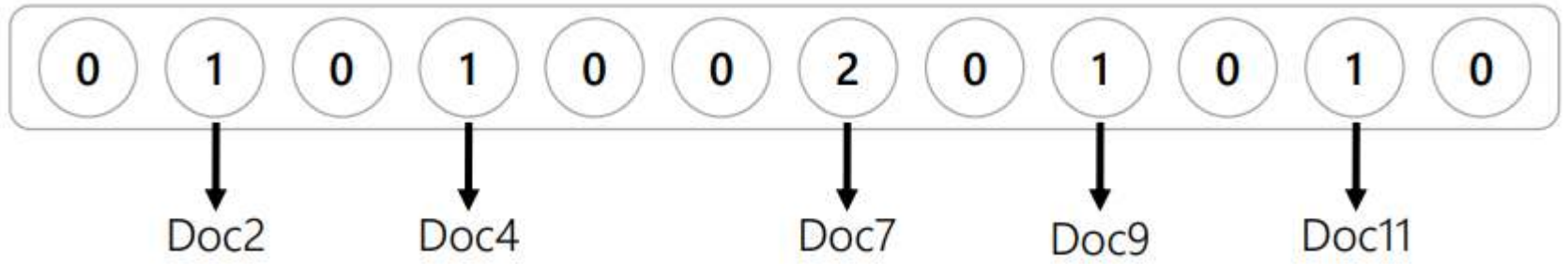
One-hot encoding, kelimeleri tanımlamak için kullanılan yöntemlerden biridir.

Vektörün boyu elimizdeki kelimelerin çeşitliliği kadardır.

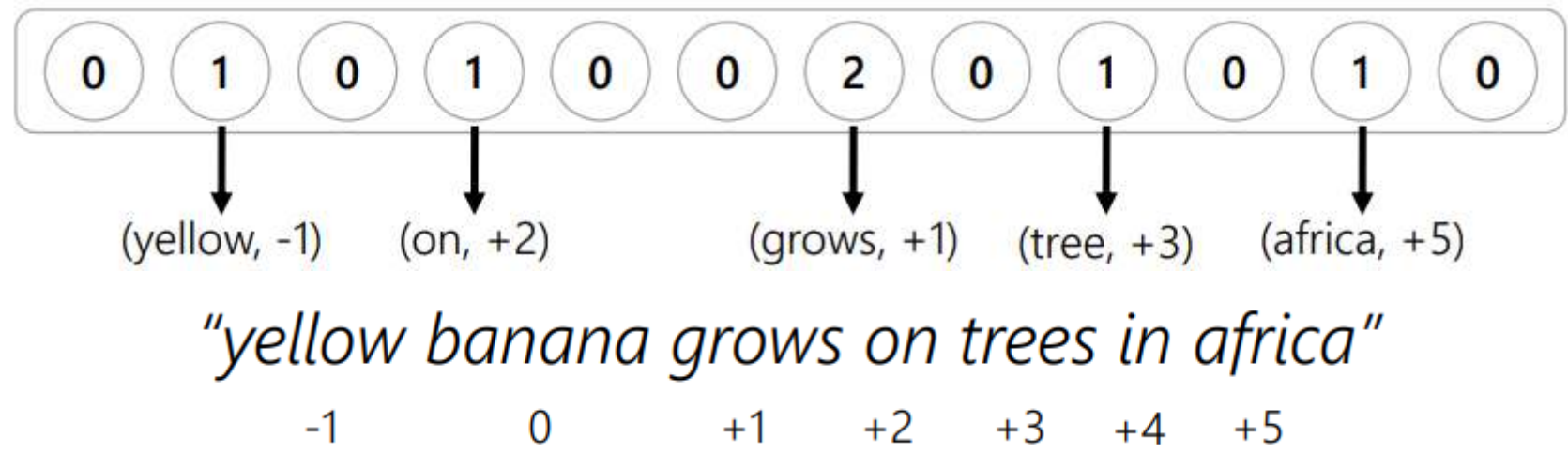
Vektörde sadece ilgili kelimenin bulunduğu yerin değeri 1, diğer tüm kelimelerin değeri 0'dır.

a	adam		zebra	Zil
1	0	0	0
0	1		0	0
0	0		0	0
.	.		.	.
.	.		.	.
.	.		.	.
0	0		1	0
0	0		0	1

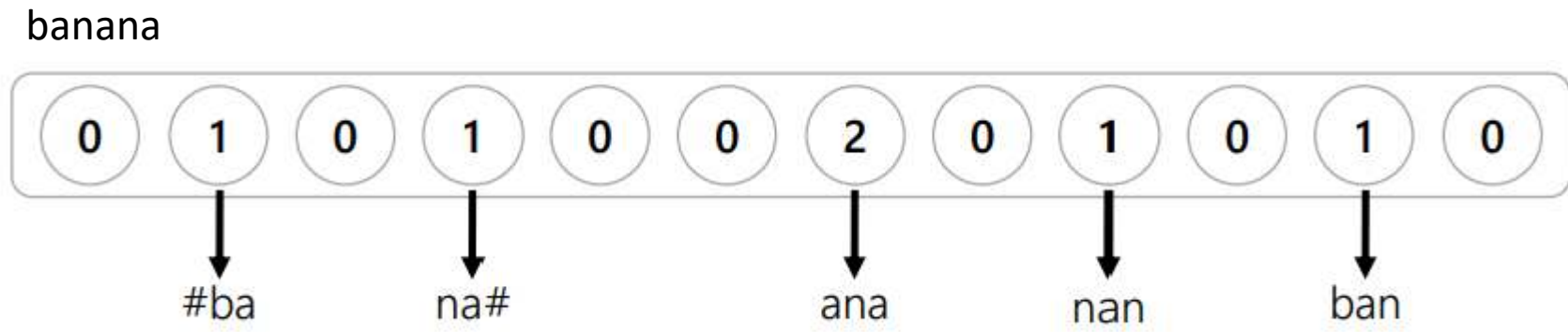
2 - Vektör, kelimenin geçtiği dokümanlara karşılık gelebilir.



3 - Vektör, komşu kelime bağlamına karşılık gelebilir.



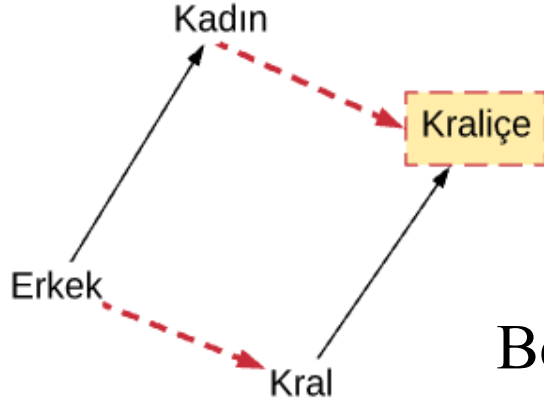
4 - Vektör, kelimedeki karakter trigramlarına karşılık gelebilir.



İlgililik- Relatedness Kavramları

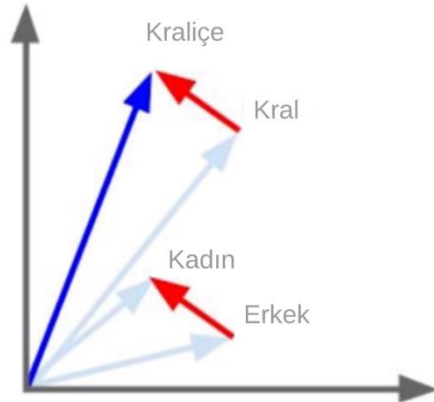
İki vektörün karşılaştırılması (örneğin, kosinüs benzerliği), iki kelimenin ne kadar benzer olduğunu tahmin eder.

Bununla birlikte, ilgililik kavramı, kelimeler için hangi vektör gösterimini seçtiğinize bağlıdır.



Benzerlik Hesabı Sonucu Bulunan Kelime

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (6)$$



Cosine Similarity ile Anlamsal Bilginin Çıkarılması

seattle similar to denver? veya seattle similar to seahawks?
Because they are both cities. Because “Seattle Seahawks”.

Dört tane dokümanımız olsun

Doküman 1 : “seattle seahawks jerseys”

Doküman 2 : “seattle seahawks highlights”

Doküman 3 : “denver broncos jerseys”

Doküman 4 : “denver broncos highlights”

Kelimelerin dokümanda geçip geçmediği bilgisini kullanırsak...

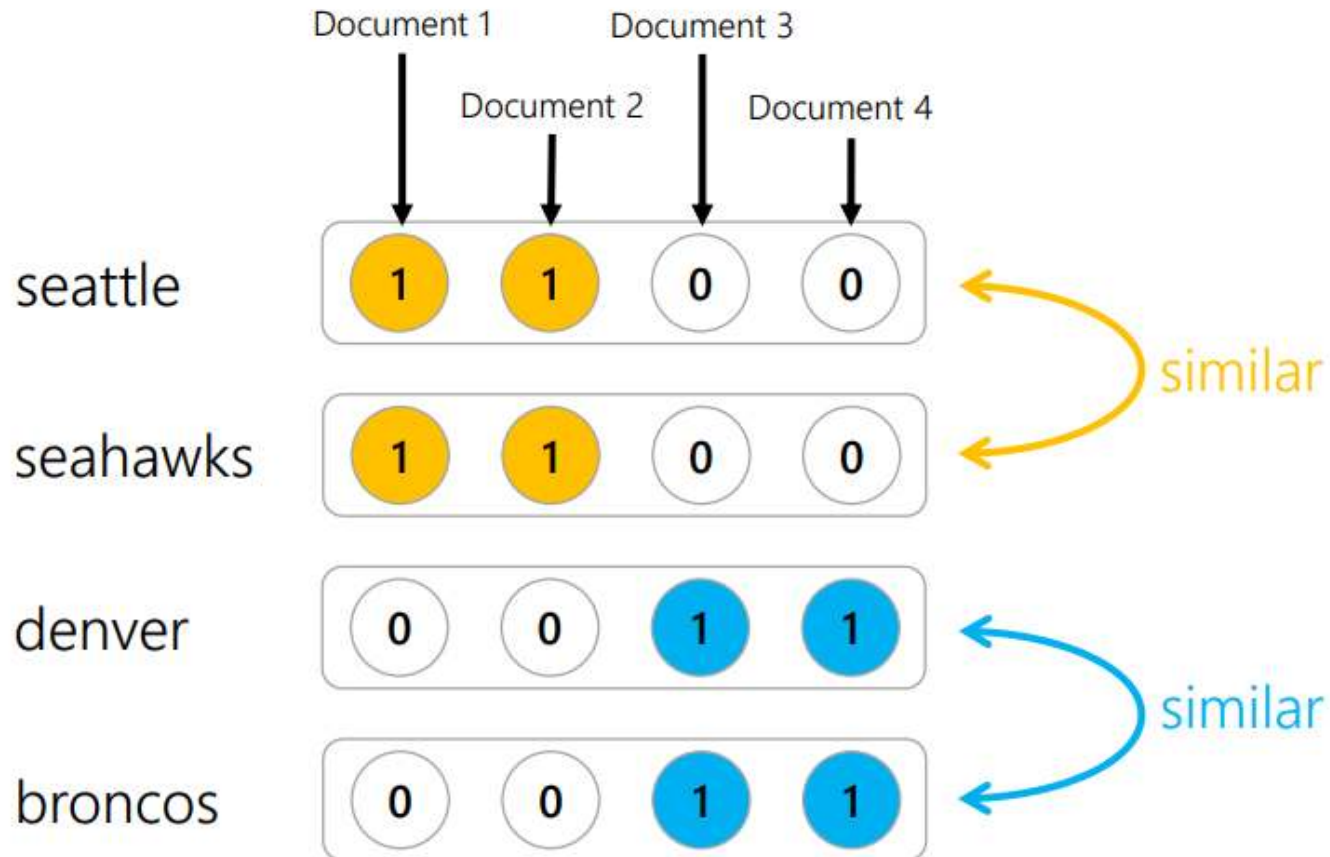
Topical similarity

Doküman 1 : "seattle seahawks jerseys"

Doküman 2 : "seattle seahawks highlights"

Doküman 3 : "denver broncos jerseys"

Doküman 4 : "denver broncos highlights"



Eğer bağlam vektörlerini kullanırsak... (*Typical (by-type)*)

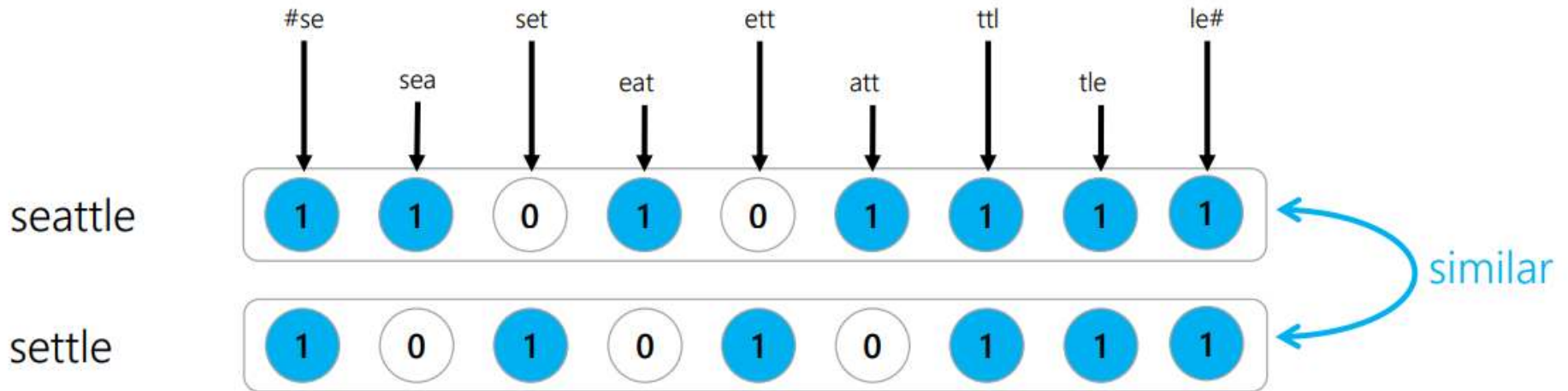
Doküman 1 : "seattle seahawks jerseys"
Doküman 2 : "seattle seahawks highlights"
Doküman 3 : "denver broncos jerseys"
Doküman 4 : "denver broncos highlights"



Eğer karakter 3-grams vektörlerini kullanırsak...

edit-distance

- Doküman 1 : "seattle seahawks jerseys"
- Doküman 2 : "seattle seahawks highlights"
- Doküman 3 : "denver broncos jerseys"
- Doküman 4 : "denver broncos highlights"



Her word-context çiftini
Pointwise Mutual Information
ile ölçelim.

$$\text{pmi}(x; y) \equiv \log \frac{p(x, y)}{p(x)p(y)}$$

Enter a word

Words	Similarity Coefficient
sydney	1
melbourne	0.4376428
brisbane	0.4071144
perth	0.3362517
adelaide	0.2916113
auckland	0.2493333

Enter a word

Words	Similarity Coefficient
batman	1
spiderman	0.1429663
superman	0.137329
ghostbusters	0.1045547
tinkerbell	0.08972809
starwars	0.07744732

Her word-context çiftini
Cosine Similarity ile ölçelim.

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

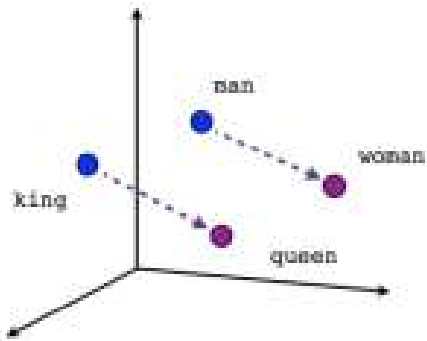
Enter a word

Words	Similarity Coefficient
java	1
c	0.1601557
javascript	0.145963
powershell	0.1096152
python	0.09570167
vb	0.0907691

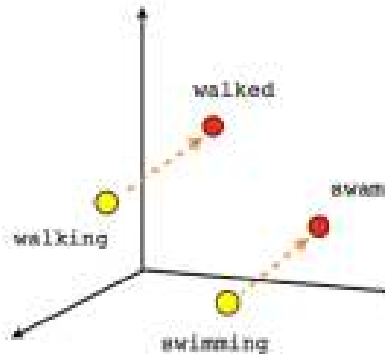
Enter a word

Words	Similarity Coefficient
pasta	1
spaghetti	0.1822345
lasagna	0.1541065
macaroni	0.1090949
salad	0.1030677
casserole	0.09800283

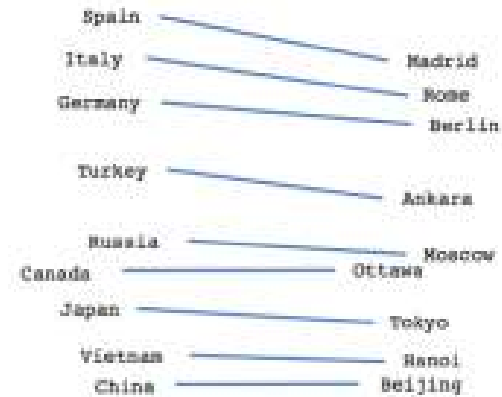
Kelime benzerliği-analojisi



Male-Female



Verb tense



Country-Capital

man is to woman as king is to ?

good is to best as smart is to ?

china is to beijing as russia is to ?

word-context kelime modeli, kelime analojisi için iyi bir çözümdür.

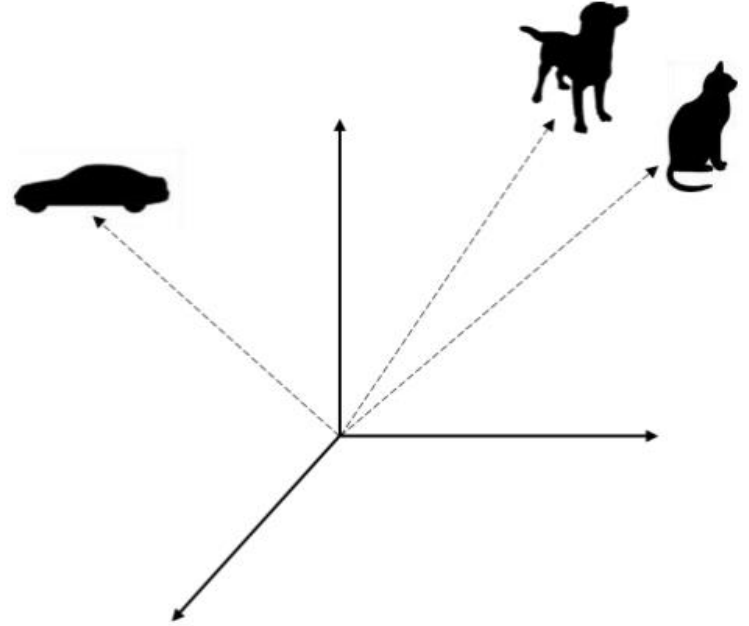
$$[\text{king}] - [\text{man}] + [\text{woman}] \approx [\text{queen}]$$

Kelime Gömmeleri-Word Embeddings

Şimdiye kadar görmüş olduğumuz vektörler çok yüksek boyutlu (binlerce) ve seyrekli (sparse).

Ancak, aynı sezgileri kullanan kelimeler için düşük boyutlu yoğun vektörleri öğrenme teknikleri de vardır.

Bu yoğun vektörlere **gömme-embedding** denir.



Word Embedding benzer anlamdaki kelimelerin yakın şekilde temsil edilmesine yarayan bir yöntemdir. Google'da çalışan Tomas Mikolov ve ekibi tarafından 2013 yılında geliştirilmiştir.

Matrix Factorization

Factorize word-context matrix.

	Context ₁	Context ₁	Context _k
Word ₁				
Word ₂				
⋮				
Word _n				

E.g.,

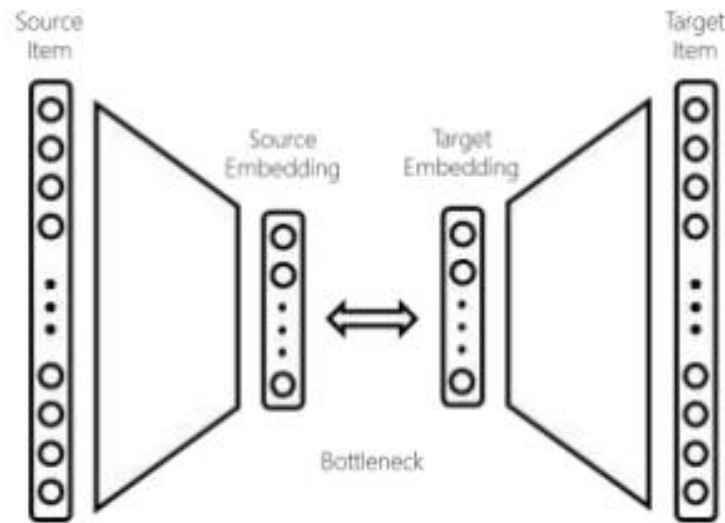
LDA (Word-Document),

GloVe (Word-NeighboringWord)

Deerwester, Dumais, Landauer, Furnas, and Harshman, Indexing by latent semantic analysis, JASIS, 1990. Pennington, Socher, and Manning, GloVe: Global Vectors for Word Representation, EMNLP, 2014.

Neural Networks

A neural network with a bottleneck, word and context as input and output respectively.



E.g.,

Word2vec (Word-NeighboringWord)

Mikolov, Sutskever, Chen, Corrado, and Dean, Distributed representations of words and phrases and their compositionality, NIPS, 2013.

Dokümanları sıralarken Word embeddings kullanımı

Geleneksel IR sistemler **Term matching** kullanır,

→ # of times the doc says Albuquerque

Word embedding kullanan modeller ise

→ # of terms in the doc that relate to Albuquerque

Albuquerque is the most populous *city* in the U.S. state of *New Mexico*. The high-altitude *city* serves as the county seat of *Bernalillo* County, and it is situated in the *central* part of the state, straddling the *Rio Grande*. The *city population* is 557,169 as of the July 1, 2014, *population* estimate from the United States Census Bureau, and ranks as the 32nd-largest *city* in the U.S. The *Metropolitan Statistical Area* (or MSA) has a *population* of 902,797 according to the United States Census Bureau's most recently available estimate for July 1, 2013.

Passage *about* Albuquerque

Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in *Albuquerque, New Mexico* in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

Passage not about Albuquerque

Sözlükte bulunan her kelimenin vektörlerinin saklandığı yapıya Vektör Matrisi (**Embedding Matrix**) denir

$$\left[\begin{array}{cccccccccc} k_0 & k_1 & k_3 & \dots\dots\dots & k_{6257} & \dots\dots\dots & k_{9997} & k_{9998} & k_{9999} \\ \vdots & & & & \ddots & & & & \vdots \\ \vdots & & & & & & & & \vdots \\ \vdots & & & & \dots & & & & \vdots \\ \vdots & & & & & & & & \vdots \end{array} \right] \left. \vphantom{\begin{array}{c} \\ \\ \\ \\ \end{array}} \right\} d$$

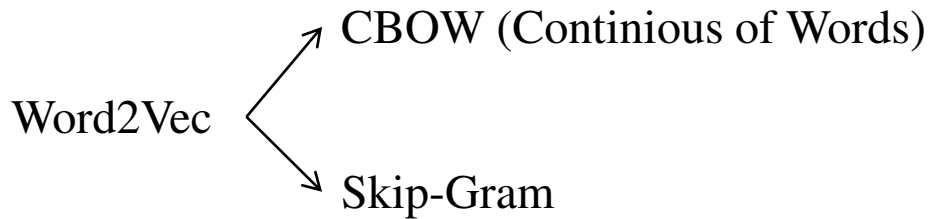
d: Vektör Boyutu

Vektör Matrislerinde:

- Bütün kelimelerin vektörleri bulunur.
- Her kelime vektörü bir sütunda yer alır.
- Kelime vektörünün boyutu **d** dir
- Sözlükte bulunan kelime sayısı **m** ise
- Vektör Matrisi boyutu (**d x m**) dir

Kelime vektörleri için kullanılan yöntemlerden en bilinenleri :
Word2Vec [1], GloVe [2], BERT

GloVe, Word2Vec yönteminin üzerine bazı performans geliştirmeleri yapılmış halidir.



Çerçeve Boyutu (Window Size)

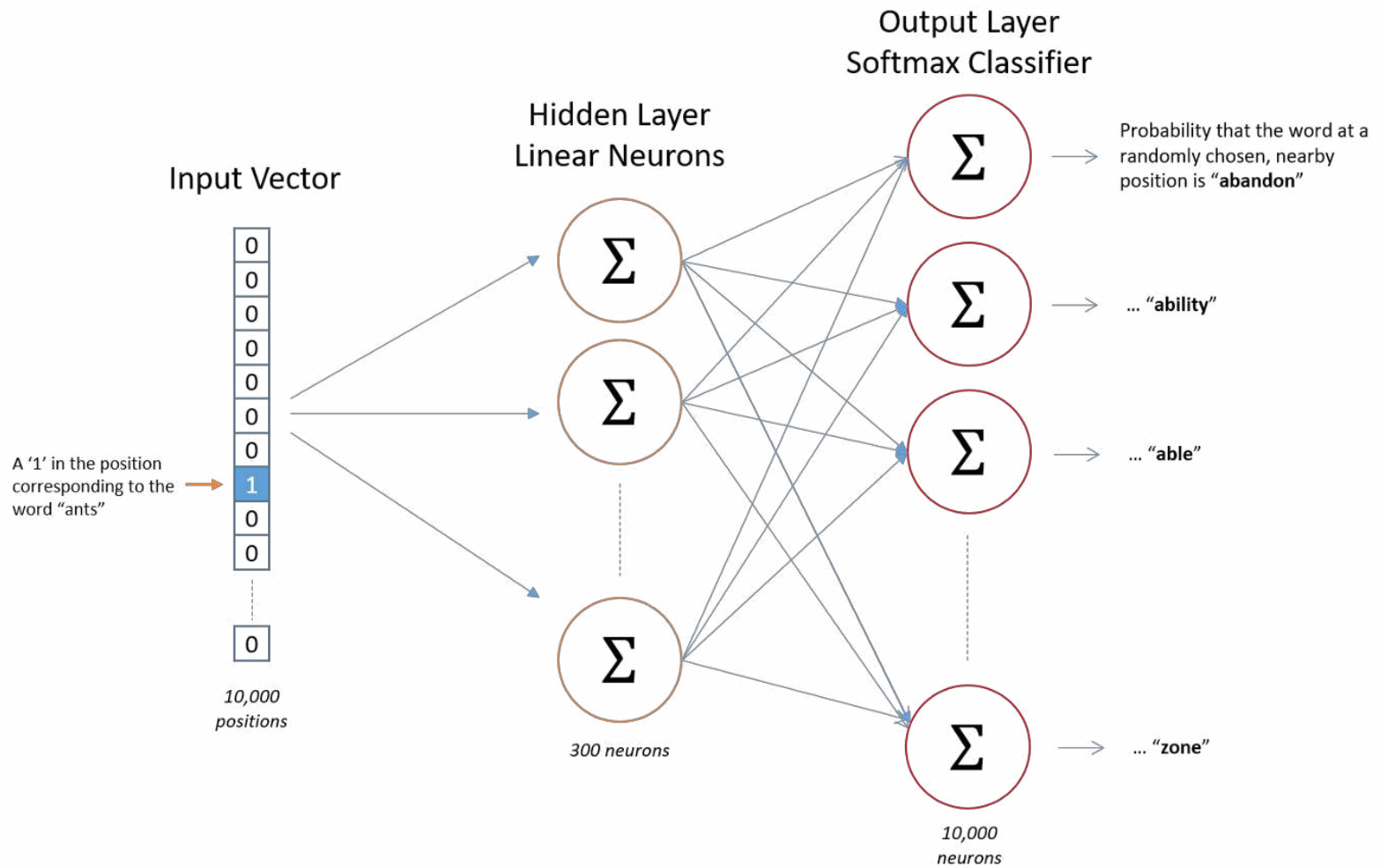
- ❖ Word2Vec için hiperparametrelerden biridir
- ❖ Analiz edilen kelimenin sağında ve solundaki **n** kelimeyi belirlemek için kullanılır
- ❖ Çerçeve merkezindeki kelimeye **Merkez Kelime**
- ❖ Bu kelimeye **n** yakınlıktaki kelimelere **Çevreleyen Kelimeler** denir

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[2] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Word2Vec

- Bir adet girdi, çıktı ve gizli katmandan oluşan bir yapay sinir ağıdır.
- Kelime vektörlerini oluştururken pencere genişliği, embedding boyutu, vs hiper parametreleri kullanır.
- Pencere genişliği, hedef kelimenin sağında ve solunda kaç kelime olması gerektiğini belirtir.
- Embedding boyutu ise, her bir kelimenin kaç boyutlu vektör olarak tanımlanacağını belirtir.
- Embedding boyutu aynı zamanda gizli katmandaki nöron sayısına karşılık gelir.
- İki kelimenin birbirine olan benzerliğini veya iki cümlenin birbirine olan yakınlığını bulmak, doküman özetleme gibi doğal dil işlemede birçok alanda kullanılabilir.

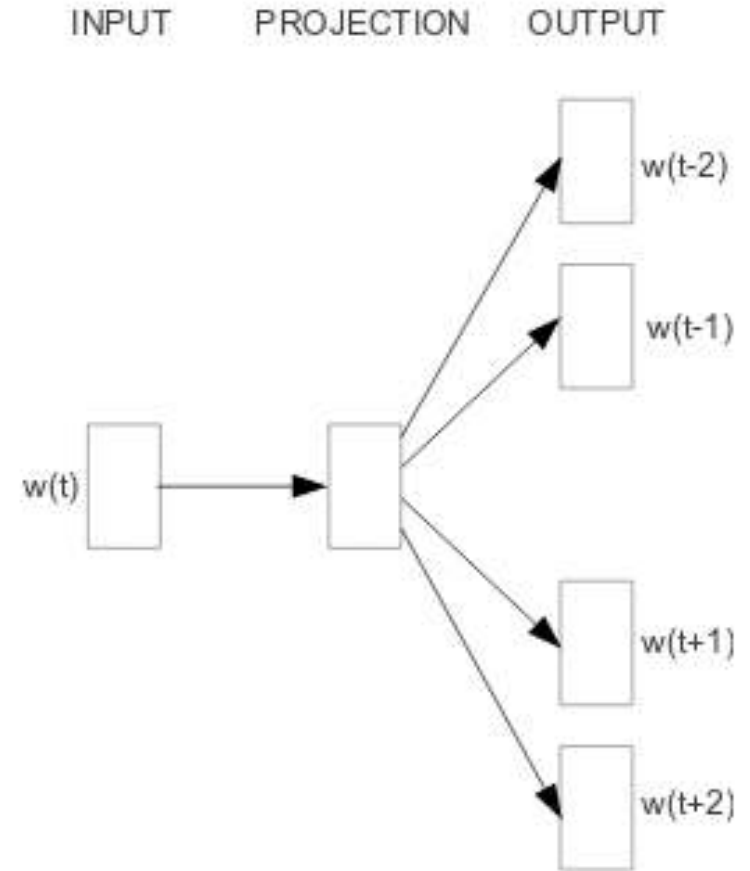


Skip-gram model

- Girdi hedef kelime iken, çıktılar hedef kelimenin etrafındaki kelimelerdir.
- Girdi ile çıktıları olasılıksal olarak birbirine benzeterek anlamsal olarak en uygun şekilde temsil etmek hedeflenir.

Örnek: “*Korkma sönmez bu şafaklarda yüzen al sancak*”

cümlesi için ‘şafaklarda’ kelimesi girdi iken ve **pencere boyutu 7** alındığında ‘korkma’, ‘sönmez’, ‘bu’, ‘yüzen’, ‘al’ ve ‘sancak’ kelimeleri de çıktı olarak verilir.



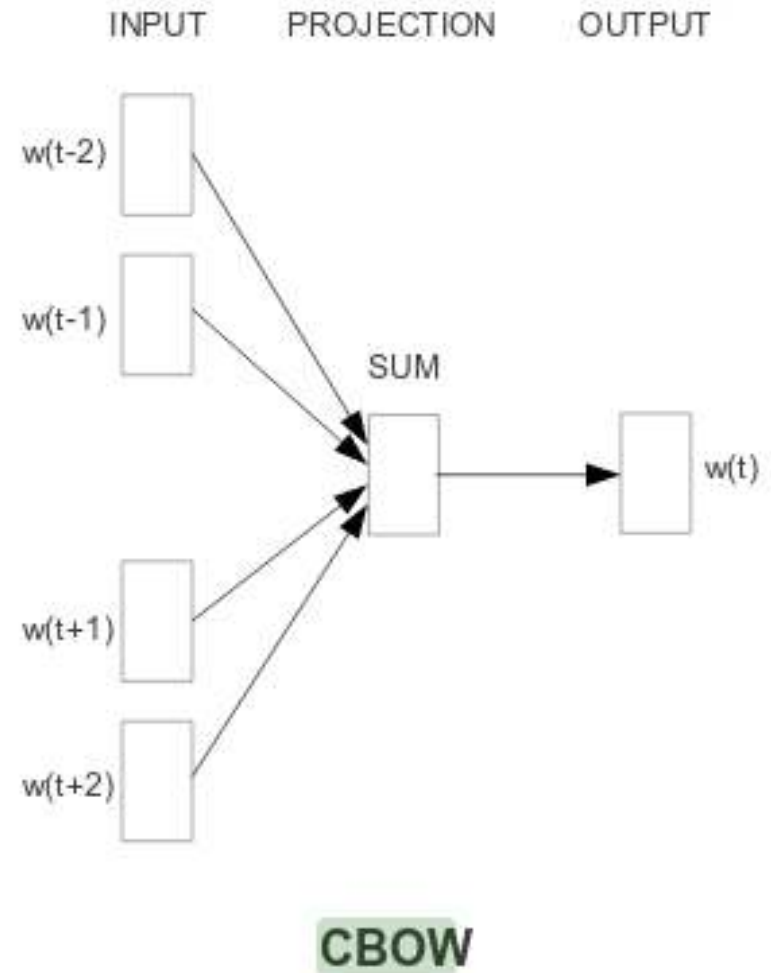
Skip-gram

CBOW modeli – (Continuous Bag of Words)

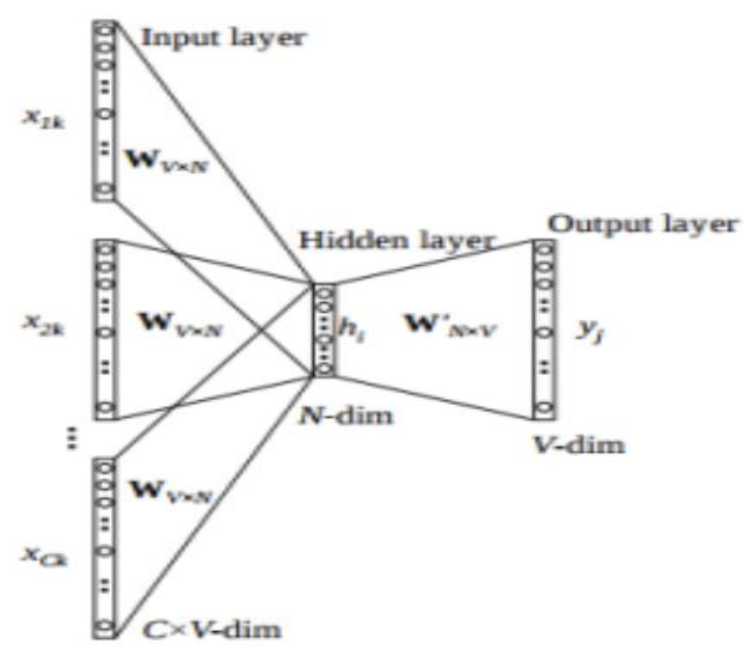
- Buradaki fikir bir kelimenin etrafındaki kelimeler verildiğinde hangi kelimenin bu kelimeler içinde görülme olasılığının en yüksek olduğunun bilinmek istenmesidir.

Örnek : “*Korkma sönmez bu şafaklarda yüzen al sancak*”

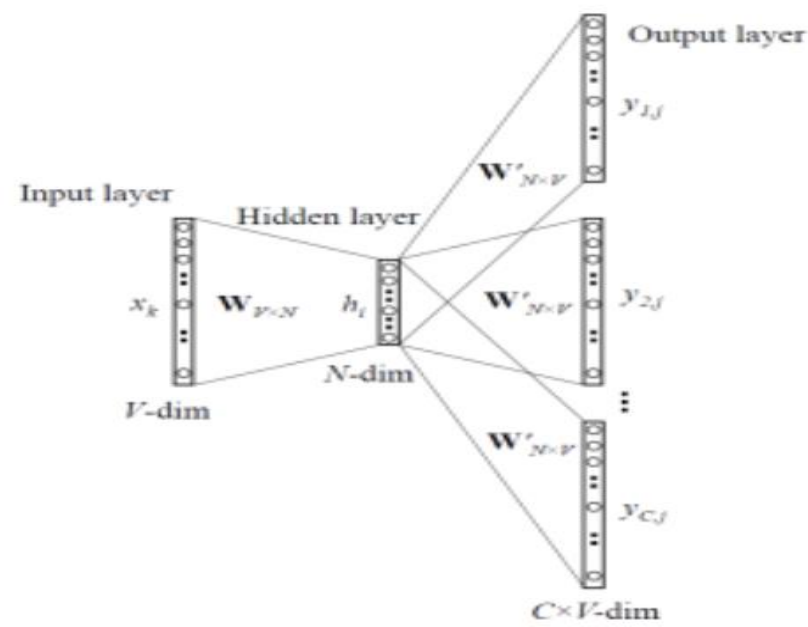
‘*korkma*’, ‘*sönmez*’, ‘*bu*’, ‘*yüzen*’, ‘*al*’ ve ‘*sancak*’ kelimeleri girdi olarak verilirken, çıktı hedef kelime olarak verilir.



Daha detaylandırarak verecek olursak:



a) CBOW Ağ Yapısı



b) Skip-Gram Ağ Yapısı

- CBOW modeli küçük veri setlerinde daha iyi sonuç verir
- Büyük veri setlerinde ise Skip-gram modeli daha iyi çalışır
- CBOW modeli için daha az işlem gücü gerekir
- Skip-gram modeli daha fazla işlem gücü gerektirir
- CBOW modeli iki veya daha çok anlamı olan kelimeleri anlamakta başarılı değildir
- Skip-gram iki veya daha çok anlamı olan kelimeleri daha iyi öğrenir

FastText

- ❖ 2016 yılında Facebook tarafından geliştirilmiş.
- ❖ Word2Vec'in bir uzantısıdır.
- ❖ Kelimeler yapay sinir ağına girdi olarak verilmez.
- ❖ Kelimeler, karakter “n-gram” halinde parçalanarak verilir.

Örnek *doğal* sözcüğü için **tri gram** değeri *doğ*, *oğa*, *ğal*'dir.

- ❖ N-gram ifadesindeki n (3) değeri, kelimenin kaçar kaçar bölüneceğini söyler.
- ❖ *doğal*'ın kelime vektörü tüm n-gram vektörlerinin toplamıdır.

- ❖ Eğitimin sonunda, eğitim setinde verilen tüm n-gramlar için kelime vektörleri oluşturulmuş olur.
- ❖ Az sıklıkta geçen kelimelerin n-gramlarının ortaya çıkma olasılığı düşük olduğu için, bu kelimeler daha doğru bir şekilde temsil edilebilir.
- ❖ Bazı kelimeler eğitim veri setinde olmamasına rağmen n-gramlar sayesinde vektör değerleri hesaplanır.
- ❖ N-gram sayısı kelime sayısından çok fazla olacağı için eğitim süresi de uzar.
- ❖ Dokümanlarda az sayıda bulunan kelimeler Word2Vec'e göre daha iyi bir şekilde ifade edilir.
- ❖ Hatalı yazılmış kelimeler için de yaklaşık bir değer ataması yapılabilir.

GloVe (Global Vector)

- ❖ Skip-gram, CBOW gibi modeller anlamsal bilgileri yakalar ancak, birlikte kullanılma istatistiklerini kullanmazlar.
- ❖ Matris ayrıştırma yöntemleri de istatistik bilgisini kullanmasına rağmen anlamsal ilişkileri yakalayamazlar.
- ❖ Pennington ve ark. tarafından önerilen “GloVe” modeli, olasılık istatistiklerinden yararlanarak yeni bir objektif fonksiyon oluşturup, bu problemi çözmeyi amaçlar.
- ❖ GloVe modelinde güncellenen hata fonksiyonu kullanılır

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

- ❖ GloVe modeli temel olarak hata fonksiyonunun (J) modellenmesi sırasında kelimelerin olasılık oranlarını da kullanır.
- ❖ Kelimelerin birlikte kullanım oranları ile güncellenen hata fonksiyonu ile CBOW ve Skip-Gram gibi çerçeve gezdirerek çevreleyen kelimelerin belirlenmesi işlemi de ortadan kaldırılmıştır.
- ❖ Pij ile birlikte bulunma olasılığı yüksek kelimeler birbirlerine yakın geçtilerse bu kelimeler öğrenme işleminde diğer kelimelerden daha önemli rol oynar.

Ülke → Başkent ilişkisi

```
1 word_vectors.most_similar(positive=["londra","iran"],negative=["ingiltere"])  
[('tahran', 0.6414337754249573),  
 ('bağdat', 0.5921595096588135),  
 ('ırak', 0.5417447090148926),  
 ('kahire', 0.5351825952529907),  
 ('erbil', 0.5221700072288513),  
 ('suriye', 0.5216017365455627),  
 ('washington', 0.5198656320571899),  
 ('başkenti', 0.5075976252555847),  
 ('arap', 0.4841397702693939),  
 ('york', 0.47895127534866333)]
```


- ❖ Kelime vektörlerinin öğrenilmesi maliyetli bir işlemdir.
- ❖ Kelime vektörlerinin başarılı bir şekilde öğrenilebilmesi için çok büyük corpus/derlem üzerinde öğrenme işleminin yapılması gerekmektedir.

- Wikipedia 2014 + Gigaword 5: 6 Milyar token, 400.000 tekil kelime sayısı, öğrenilen kelime vektörleri 50, 100, 200 ve 300 boyutlu
- Common Crawl 1: Dünya genelinde web sayfaları crawl edilmiş, 42 milyar token, 1,9 milyon tekil kelime sayısı, öğrenilen kelime vektörleri 300 boyutlu
- Common Crawl 2: Dünya genelinde web sayfaları crawl edilmiş, 840 milyar token, 2,2 milyon tekil kelime sayısı, öğrenilen kelime vektörleri 300 boyutlu
- Twitter: 2 milyar tweet verisinden elde edilen 27 milyar token, 1,2 milyon tekil kelime sayısı, öğrenilen kelime vektörleri 25, 50, 100 ve 200 boyutlu

- Google, Google News, 100 milyar token, 3 milyon tekil kelime sayısı, öğrenilen vektör boyutu 300 [3]
- Facebook, 294 dil için, 300 boyutlu kelime vektörleri [4]
- Türkçe diline özel olarak eğitilmiş kelime vektörleri [5]

[3] Google's trained Word2Vec model in Python, <http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/>

[4] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606.

[5] Word Embedding based Semantic Relation Detection for Turkish Language, <https://github.com/savasy/TurkishWordEmbeddings>

Bidirectional Encoder Representations from Transformers –BERT

Google, 2018

- Cümleyi hem soldan sağa hem de sağdan sola değerlendiriyor.
- Cümlelerin anlamını ve kelimelerin birbiriyle olan ilişkileri daha iyi çıkarılmış oluyor
- 800M kelimelik BookCorpus ve 2.5B kelimelik Wikipedia veriseti olarak kullanılmıştır
- *bert_large* (16 adet TPU-Tensor Processing Unit) ve *bert_base* adı (4 adet TPU) ile 4 gün train edilmiştir
- Chatbot, metin sınıflandırma gibi problemlerin çözümünde kullanmak için modelin üstüne ekstra katmanlar eklenmesi gerekir

- BERT, çift-yönlü olması dışında *Masked Language Modeling (MLM)* ve *Next Sentence Prediction (NSP)* adı verilen iki teknikle eğitilir
- Bir cümle modele girdiğinde, cümledeki kelimelerin %15'inde MLM tekniği kullanılıp, bu kelimelerin %80'i [MASK] token'ı ile, %10'u rastgele başka bir kelimeyle değiştirilir ve geri kalan %10 da değiştirilmeden bırakılır
- MLM tekniğinde, maskelenen kelime, açık şekilde beslenen kelimelerle tahmin edilmeye çalışılır
- MLM, cümle içerisindeki kelimeler arasındaki ilişki üzerinde dururken, NSP ise cümleler arasındaki ilişki çıkarmaya çalışır
- Training esnasında ikili olarak gelen cümle çiftinde, ikinci cümlenin ilk cümlenin devamı olup olmadığı tahmin edilir
- Hazır-eğitilmiş modeller, fine-tuning adı verilen teknikle yeni problemlerin çözümünde kullanılabilir