Kelime Gösterimleri (Word Representation –Word Embeddings)

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

Kelime gösterilimi :

- Frekans Tabanlı Kelime Gösterilimi (*Count Vector, Tf-Idf*)
- Tahminleme Tabanlı Kelime Gösterilimi (*Word2Vec, GloVe – Derin Öğrenme Yaklaşımlı Yapay Sinir Ağı Yöntemleri)*

❖Tahminleme yaklaşımları ile kelimeler arasındaki anlamsal ilişkiler çıkarılabilir

Erkek →Kadın | Kral → ? (Kraliçe)

❖Kelime vektörleri arasındaki mesafe (*Cosine Similarity*)

❖Kelime Vektörlerinin öğrenilmesi eğiticisiz öğrenmeye girer

❖Her kelime için belirli sayıda özellik öğrenilir

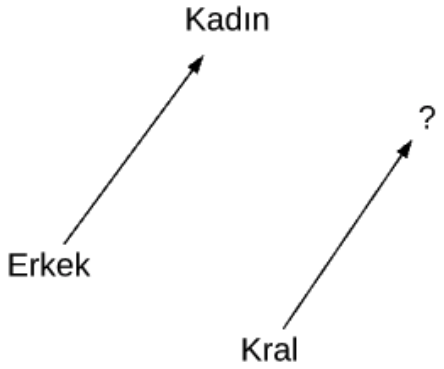
❖Her özellik bir anlamsal bilgi taşır

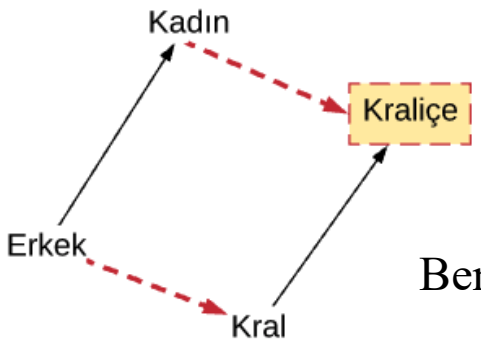
Kelime Vektörlerinin Görselleştirilmesi

Erkek →Kadın | Kral → ?

Verkek – Vkadın =~Vkral – V?

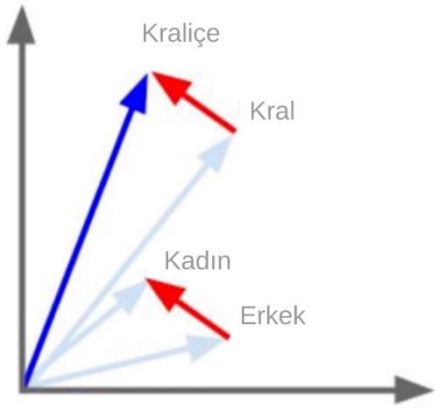
argmax benzerlik(V?, Vkral - Verkek – Vkadın





Benzerlik Hesabı Sonucu Bulunan Kelime

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

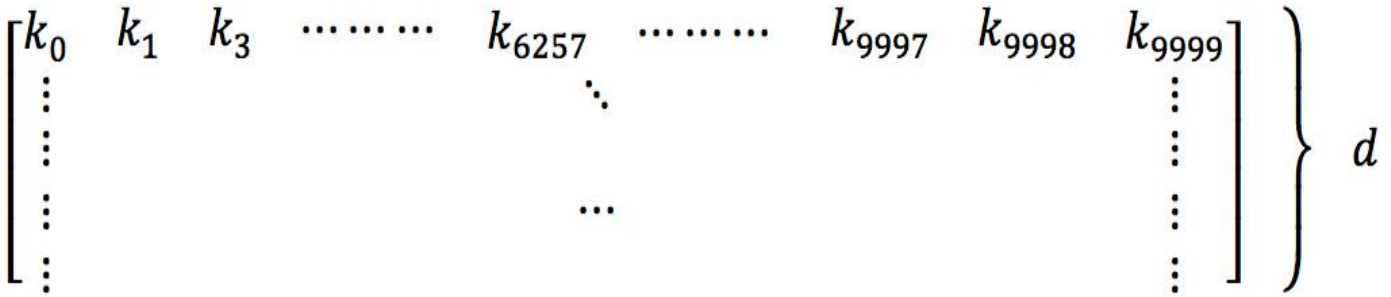


Cosine Similarity ile Anlamsal Bilginin Çıkarılması

One-hot Gösterimi

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

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



d: Vektör Boyutu

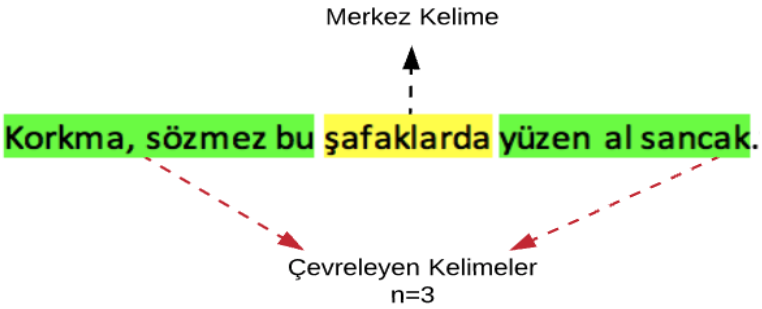
Vektör Matrislerinde:

- 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] ve GloVe [2]

GloVe, Word2Vec yönteminin üzerine bazı performans geliştirmeleri yapılmış halidir.

Word2Vec
- CBOW (Continious of Words)
- Skip-Gram



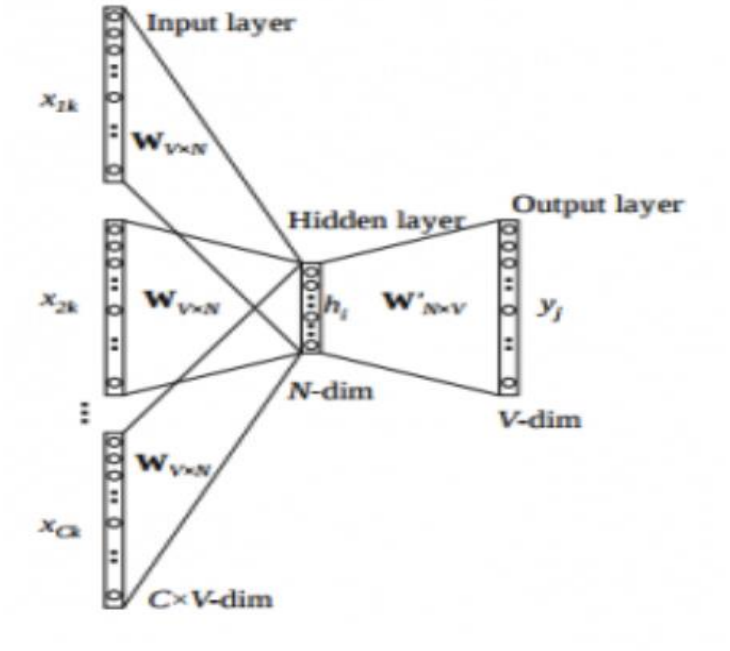
Çerçeve Boyutu (Window Size)

- 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

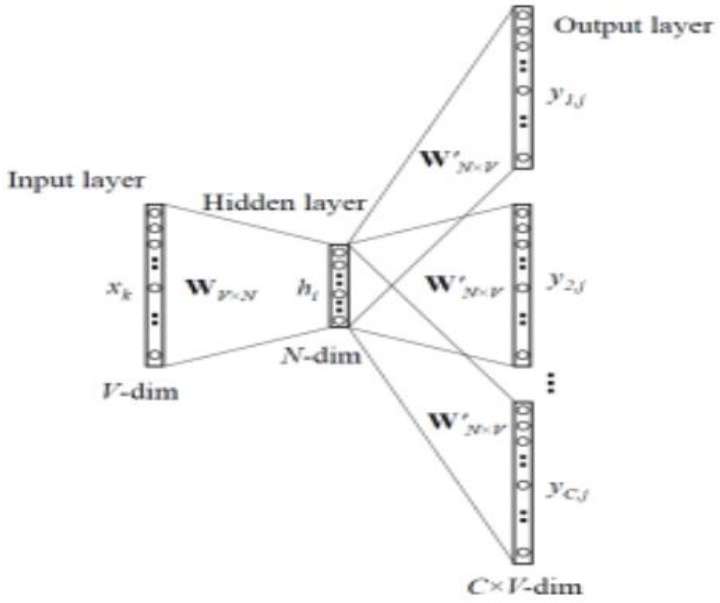
[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[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).

- CBOW modelinde çevreleyen kelimeler girdi, merkez kelime çıktı olarak işlenir
- Skip-Gram modelinde ise merkez kelime, çevreleyen kelimeler çıktı olarak işlenir



a) CBOW Ağ Yapısı



b) Skip-Gram Ağ Yapısı

Word2Vector aslında birçok kullanımı olan bir araçtır. İki kelimenin birbirine benzerliğinden iki cümlenin birbirine yakınlığına kadar kullanılabilir. Bu amaçla özetlemede de kullanılabilir. Word2Vector arka planda bir yapay sinir ağı kullanır.

NLP de önemli bir kavram olan *document*term* matrisi, ***m*n*** lik bir matris olarak düşünecek olursak, elimizde ***m*** tane doküman ve ***n*** tane de tekil kelimeden oluşur.

	t_1	t_2	t…….	t_{n-1}	t_n
doc_1					
doc_2					
doc…					
doc_m					

Buradaki ***n*** değeri sözlükte yer alan tekil kelime sayısıdır. Bu sayı oldukça büyük olabilir. Bu matris her terim her dokümanda kaç kez geçiyorsa o değer ile doldurulur. Kelime sayısı ne kadar fazla ise bu matris o kadar sparse olur.

Bu matris oluşturulurken kelimeler birbirinden bağımsızdır. Mesela t_1 fenerbahçe, t_2 Galatasaray olsun. Bu iki kelime bize göre birbirine yakındır. Ama, kitap kelimesine göre uzaktır. Bilgisayar için Fenerbahçe, Galatasaray ve kitap kelimeleri arasında bir benzerlik yoktur.

Word2Vector'ün yaptığı bu kelimeler arasındaki benzerliği çıkarabilmektir. Bunun için ilk yapılması gereken her kelime için aynı uzunlukta bir vektör oluşturulmasını sağlamaktır. Kelime vektörünün boyutunun ne olacağı parametrik bir değerdir. Biz bu anlatımda bu değeri 200 olarak kabul edelim.

Yani hesaplanması gereken ***n*** tane kelime vektörü ve her bir kelime vektörünün de boyutu 200.

t_1=fenerbahçe ($)_{200}$
t_2=galatasaray ($)_{200}$

.....
t_n=kitap ($)_{200}$

Şimdi asıl sorun acaba bu vektörleri nasıl oluşturmalıyız?

Ki bu değerler bize kelimenin birbirine yakınlığı ile ilgili bilgi versin.

Burada kullanılması gereken bir yapay sinir ağı modelidir. Sadece tek saklı katmanı olan bir YSA yeterlidir.

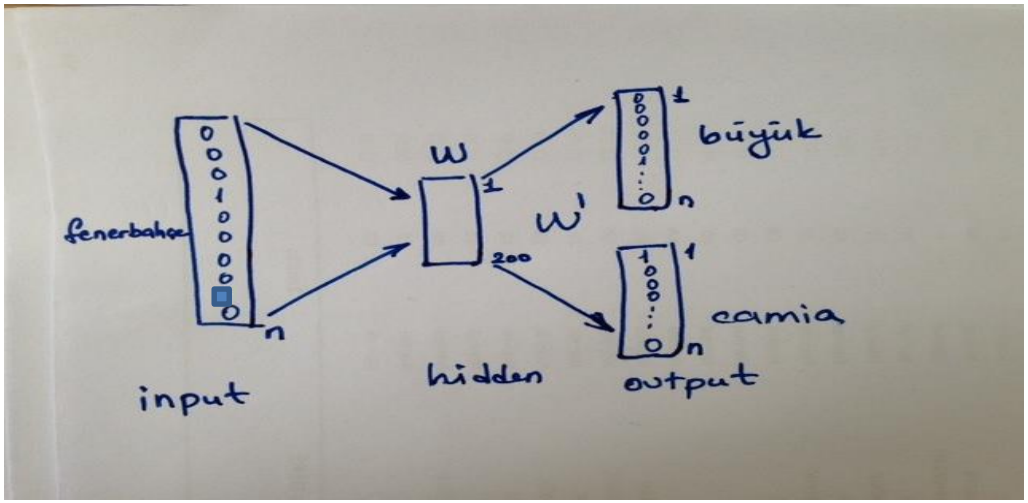
Öncelikli olarak elimizdeki corpus şöyle olsun.

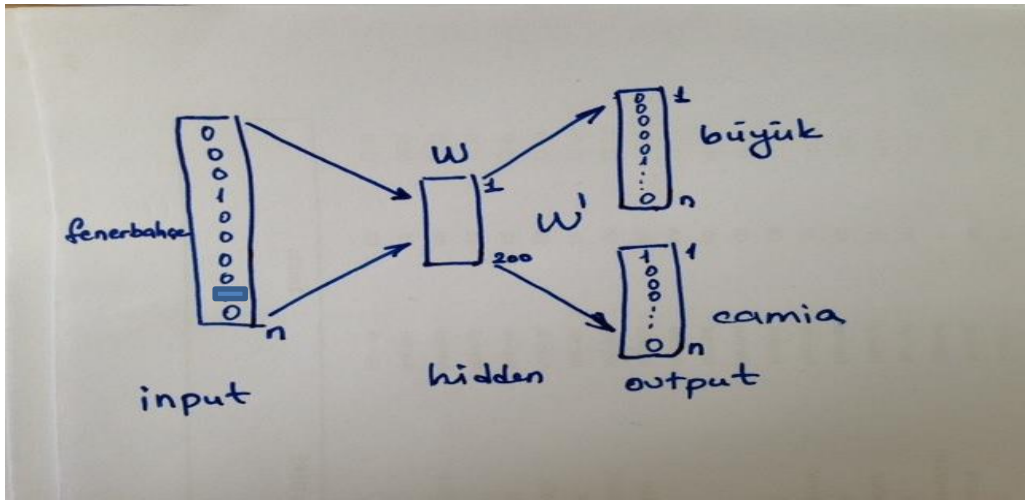
$w_1\ w_2\ w_3\ w_4\ w_5\ w_2\ w_3\ w_6\ w_7 \ldots\ldots\ W_{45}$

Bir (window size) belirleyelim. Bu örnek için 3 olsun.
Amacımız bu doküman üzerinde kaydırarak bu üç boyutlu pencereyi gezdirmek.
Öncelikle elimizdeki tüm corpus içerisinden tekil kelimeler çıkarılır
Bir sözlük oluşturulur.
Sözlük içerisinde her kelimenin bir indeks sırası vardır.

w_1	w_2	w_3
büyük	fenerbahçe	camiası

Biz şimdi Fenerbahçe kelimesi için şekildeki gibi bir yapay sinir ağı oluşturalım.





•Input layer da n adet kelime olduğu için n*1 boyutunda bir vektör yer alır.

•Amacımız Fenerbahçe için vektör çıkarmak ise bu vektörde sadece Fenerbahçe nin sözlükteki sırasının olduğu göze **1** değeri diğerlerine **0** yazılır.

•Yani sözlükte fenarbahçe dördüncü sırada ise dördüncü gözün değeri 1 diğerleri sıfırdır

Bu işlem ***one hat vector*** olarak adlandırılır.

▪ Hidden layer da hesaplanan ve corpus süresince güncellenen her kelime için 200*1 boyutunda bir vektör hesaplanır.

▪ Bu işlem corpus süresince devam ettirildiğinde n kelime için de 200*n boyutunda bir matris elde edilmiş olacaktır.

▪ Bu ağın amacı bu matrisi oluşturmaktır. Matrisin her satırı 200 boyutundaki n farklı kelime için gerekli olan vektörü verecektir.

▪ Pencereyi corpus üzerinde kaydırdıkça bazı kelimelerin vektörleri geçiş sıklığına göre defalarca güncellenecektir.

▪Output layer da ise 2 tane vektör yer alır. Bunlardan biri Fenerbahçe kelimesinin solunda yer alan *büyük* diğeri de sağında yer alan *camia* kelimesi için olan vektör.

▪Bu vektörler de ***n*** boyutundadır ve kelimenin sözlükteki sırası neyse vektördeki o göze **1** diğerlerine de **0** konulur.

▪Ve yapay sinir ağı çalıştırılır. Oluşan yapay sinir ağı **fullconnected** bir yapıya sahiptir.
W→ n*200
W' →200*(v*2)

CBOW		Skip-Gram	
Girdi	Çıktı	Girdi	Çıktı
Korkma, sözmez, bu, yüzen, al, sancak	Şafaklarda	Şafaklarda	Korkma, sözmez, bu, yüzen, al, sancak

- CBOW modelleri küçük veri setlerinde daha iyi sonuçlar verir
- Büyük veri setlerinde Skip-gram modeli daha iyi çalışır
- CBOW modeli için daha az işlem gücü gerekir
- Skip-gram modeli daha fazla işlem gücü tüketir
- CBOW modeli iki veya daha çok anlamlı kelimeleri anlamakta başarılı değildir
- Skip-gram iki veya daha çok anlamlı kelimeleri daha iyi öğrenir

GloVe, çevreleyen kelimelerin belirlenmesinde istatistiksel tabanlı bir yaklaşım kullanır

CBOW ve Skip-Gram da olduğu gibi çerçeve gezdirerek çevreleyen kelimelerin belirlenmesi işlemi yapılmaz

GloVe modelinde güncellenen hata fonksiyonu

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

▪ 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 daki gibi çerçeve gezdirerek çevreleyen kelimelerin belirlenmesi işlemi ortadan kaldırılmıştır

• Pij ile birlikte bulunma olasılığı yüksek kelimeler birbirlerine yakın geçtilerse bu kelimeler öğrenme işleminde diğer kelimelerden daha önemli rol oynar

- Kelime vektörlerinin öğrenilmesi maliyetli bir işlemdir
- Kelime vektörlerinin başarılı bir şekilde öğrenilebilmesi için çok büyük corpuslar üzerinde öğrenme işleminin yapılması gerekmektedir

• Wikipedia 2014 + Gigaword 5: 6 Milyar token, tekil kelime sayısı 400.000, öğrenilen kelime vektörleri 50, 100, 200 ve 300 boyutlu

• Common Crawl 1: Dünya genelinde web sayfaları crawl edilmiş, 42 milyar token, elde edilen tekil kelime sayısı 1,9 milyon, öğrenilen kelime vektörleri 300 boyutlu

• Common Crawl 2: Dünya genelinde web sayfaları crawl edilmiş, 840 milyar token, tekil kelime sayısı 2,2 milyon, öğrenilen kelime vektörleri 300 boyutlu

• Twitter: 2 milyar tweet'ten elde edilen 27 milyar token, tekil kelime sayısı 1,2 milyon, öğrenilen kelime vektörleri 25, 50, 100 ve 200 boyutlu

Google, Google News, 100 milyar token, tekil kelime sayısı ise 3 milyon, 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