**Kelime Vaktöreleri**

Kelime vektörleri, benzer anlama sahip kelimelerin benzer temsillere sahip olmasına izin veren bir eşleme türüdür. Doğal dilin makine öğrenmesi algoritmaları ile işleyebilmek için sayısal bir forma dönüştürülmesi gereklidir. Geleneksel yaklaşımda, corpus’taki kelime sayısı uzunluğunda bir vektör oluşturulup, ilgili kelimenin indeksi 1, vektörün geri kalan elemanları ise 0 olarak atanır. Bu tarz vektörlere “One-Hot” vektörler denir. Genellikle her bir kelimenin indeksinin belirlenmesi için corpus’taki benzersiz kelimeler alfabetik olarak sıralanır. Bu durumda “a” harfi ile başlayan kelimelerin 1 değerli indeksi vektörde başlardayken, “y” “z” gibi harflerle başlayan kelimelerin 1 değerli indeksi vektörde sonlarda yer almaktadır. Kelimelerin one-hot vektörlerle temsil edilmesi ve eğitilmesi oldukça kolay olsa da bu temsilin önemli problemleri vardır. En önemli problemlerden biri, bu temsil doğal dilin yapısını iyi yansıtamamaktadır. Doğal dilde kelimeler aynı anda bulunmalarına bağlı olarak anlam kazanmaktadır fakat One-hot vektörlerde böyle bir anlam çıkarılamamaktadır. Ayrıca corpus’un çok büyük olması durumunda her bir kelime için oluşturulacak olan vektörlerin uzunluğu çok büyük olacak ve gereksiz olarak hafızada yer kaplayan 1 elemanı dışındaki tüm elemanları 0 olan vektörler elde edilecektir, bu durumdaki problem “sparsity” – “seyreklik” problemi olarak geçmektedir. Bu yüzden literatürde kelimeleri daha iyi temsil etmek için yeni algoritmalar geliştirilmiştir. Popüler olarak Word2Vec, FastText, GloVe gibi kelime vektörü bulma algoritmaları kullanılmaktadır. Python dili için yazılmış olan Gensim kütüphanesi ile bu algoritmalar oldukça kolay bir şekilde kendi problemlerimizde kullanılabilir.

**Word2Vec**

2013 yılında Goodle’daki araştırmacılar tarafından bulunmuş bir gözetimsiz yapay sinir ağıdır. Kelime vektörlerinin öğrenilmesi için kullanılır ve oldukça basit bir mantığı vardır. Özünde fully connected katmandan yararlanan bir sinir ağıdır. Ve tek bir gizli katmanı vardır. Gizli katmanındaki nöron sayısı istenilen kelime vektörü uzunluğuna (embedding size) eşittir. Böylece kelime vektörlerinin uzunluğu corpus’ta yer alan kelime sayısından, gizli katmandaki (hidden layer) nöron sayısına indirilmiş olur. Bu nöron sayısı genellikle 50 ile 300 arasında belirlenebilir, fakat bu değerlerin üzerinde ya da altında da olan uzunlukları kullanan çalışmalar olmuştur.

Word2Vec küçük ve çok büyük corpus’larda uygulanabilir. Milyarlarca kelimeden oluşan yazılar üzerinde hızlıca eğitim gerçekleştirip oldukça isabetli kelime vektörleri elde edebilir. 2 adet temel algoritması vardır. Bunlar Skip-gram ve CBOW (Continuous Bag of Words)’dür. Bu iki farklı algoritmayı eğitmek için ise 2 farklı eğitim yöntemi vardır: Hierarchical Softmax ve Negative Sampling.

Skip- Gram:

* Ortadaki kelimeye göre kenardaki kelimeleri tahmin eder.
* Input: Ortadaki kelime, output: kenardaki kelimeler
* Ortadaki kelimenin sağından ve solundan pencere büyüklüğü kadar kelime tahmin edilir.
* Pencere büyüklüğü 3 ise sağdan üç kelime ve soldan üç kelime dikkate alınır.
* Cümlenin başında sadece sağ taraftaki kelimeler (pencere büyüklüğü kadar) alınır. Ve bir sonraki adımda cümle başındaki kelimenin bir sağındaki kelimeye geçilir. Bu durumda ise soldan 1 adet, sağdan ise pencere büyüklüğü kadar kelime alınır. Algoritma bu şekilde cümledeki kelimeler üzerinde kayarak ilerler.
* Burada tahminleme işlemi bir softmax aktivasyon fonksiyonu çıktısıdır. Bu şekilde softmax çıktısının her bir öğesi, contexte belirli bir kelimenin ne kadar olası görülebileceğini açıklamaktadır.
* Bir kelime verildiğinde context’ini (bağlamla alakalı diğer kelimeleri) bulmaya çalışır.

Tabii ki, tek bir cümle üzerinde yukarıda bahsedilen kelimeler üzerindeki ilerleme çok performanslı bir sonuç vermez. Fakat wikipedia’nın tamamı gibi çok büyük bir corpus üzerinde işlem gerçekleştirilirse elde edilen vektörler, kelimelerin birbiriyle ilişkisini çok daha iyi yansıtacaktır.

CBOW:

* Skip-gram’ın tersine işlem yaptığı hali gibi düşünülebilir.
* Network yapısı aynıdır, sadece input ve output katmanlarının yerinin karşılıklı olarak değiştirilmiş olduğu düşünülebilir.
* Kenardaki kelimelere göre ortadaki kelime tahmin edilir.
* Input: kenardaki kelimeler output: ortadaki kelime
* Skip gram ile aynı şekilde yazının en başından (solundan) başlayıp kelimeler üzerinde adım adım ilerleyerek kelime vektörleri öğrenilmektedir.
* Hedef bir “context” verildiğinde hangi kelimenin olası olarak bu kelimeler ile yer alacağının bulunmasıdır.

|  |  |
| --- | --- |
| Skip-gram | CBOW |
| CBOW’a göre küçük corpus’ta daha iyi sonuç verir. | Daha büyük corpus’a ihtiyaç duyar. |
| Nadir kelimeler daha iyi temsil edilir. | Nadir kelimeleri temsil etmekte sıkıntı yaşayabilir. Sık kullanılan kelimeler daha iyi temsil edilir. |
| CBOW’a göre daha yavaş çalışmaktadır. | Eğitim daha hızlı gerçekleşir. |

Karşılaştırma yapıldığında sonuç, yukarıdaki tablo gibi olsa da Skip-gram ve CBOW’un ikisi de oldukça başarılı sonuç vermektedir. Bu yüzden kelime vektörleri oluşturulurken ikisi de kullanılabilir.

Sonuç olarak elde edilen kelime vektörleri, Word2Vec ile iyi bir öğrenme gerçekleştiyse, çiftler halinde uzaysal düzlemde belirli anlamlar ifade etmektedir. Vektörler üzerinde nokta çarpımı gibi yöntemlerle iki kelime arasındaki benzerlik ölçülebilir. Veya iki kelime arasındaki uzaklığın (farkın) anlamı başka bir kelime çifti ile aynı olabilir. Örneğin “King”-“Man” arasındaki farka bakarak “Queen” verildiğinde “Woman” bulunabilir.

**FastText**

Word2Vec one-hot vektörlere göre çok başarılı kelime vektörleri üretse de önemli bir problemi vardır. Bu problem, eğitim veri setinde görülmeyen kelimelerin temsil edilememesidir. Daha fazla kelime içeren daha büyük bir eğitim veri seti kullanılsa dahi çok nadir kullanılan bazı kelimelerin vektörleri hiçbir şekilde elde edilemez. Bu problemin çözülmesi için Facebook 2016 yılında FastText adı verilen, temelde Word2Vec algoritmasına bir eklenti gibi olan algoritmayı geliştirmiştir. FastText’in Word2Vec’ten farklı kelimeleri teker teker ağa beslememesidir. Bunun yerine her bir kelimeyi alt kelimelerine (sub-words) ayırarak ağa besler. Bu yaklaşıma “n-gram” da denilmektedir. Örneğin 3-gram olarak “house” kelimesi için inputlar oluşturulumak istendiğinde “hou”, “ous” ve “use” alt kelimeleri ağa beslenecektir. Ağdan çıktı olarak alt kelimelerin kelime vektörleri alınacaktır. “House” kelimesinin kelime vektörü ise, alt kelimelerin vektörlerinin toplamı olacaktır. Bu şekilde nadir kelimelerin de temsil edilebilmesine olanak sağlanmış olur çünkü alt kelimelerin bazılarının başka kelimelerde de yer alma olasılığı yüksektir. FastText algoritmasını eğitmek Word2Vec’ten daha yavaştır (alt kelime sayısı daha fazla olacağı için). Ama yavaşlık sonucunda daha iyi temsil edilmiş kelime vektörleri elde edilir ve nadir kelimeler de uygun şekilde temsil edilebilir.

**Öğrenci No**: 91190000048

**Ad Soyad**: Başak Katuk