TÜRKÇE DİZİ ETİKETLEME İÇİN SİNİR AĞ MODELLERİ

NEURAL MODELS FOR TURKISH SEQUENCE LABELING

YASİN EŞREF

DR. ÖĞR. ÜYESİ BURCU CAN BUĞLALILAR

Danışman

Hacettepe Üniversitesi

Lisansüstü Egitim-Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Mühendisligi Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2019

YASİN EŞREF'in hazırladığı **"Türkçe Dizi Etiketleme İçin Sinir Ağ Modelleri"** adlı bu çalışma aşağıdaki jüri tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof. Dr. İlyas ÇİÇEKLİ
Başkan . . . . . .

Dr. Öğr. Üyesi Burcu Can BUĞLALILAR
Danışman . . . . . .

Doç Dr. Cengiz ACARTÜRK
Üye . . . . . .

Dr. Öğr. Üyesi Gönenç ERCAN
Üye . . . . . .

Dr. Öğr. Üyesi Özkan KILIÇ
Üye . . . . . .

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak . . . /. . . /. . . tarihinde onaylanmıştır.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Fen Bilimleri Enstitüsü Müdürü

Aileme. . .

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

18/06/2019

YASİN EŞREF

YAYINLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe Üniversitesi'ne verdiğimi bildiririm. Bu izinle Üniversite'ye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversite'ye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan "**Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge**" kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H.Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- ☐ Enstitü / Fakulte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.
- ☐ Enstitü / Fakulte yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir.
- ☐ Tezim ile ilgili gizlilik kararı verilmiştir.

18/06/2019

YASİN EŞREF

ÖZET

TÜRKÇE DİZİ ETİKETLEME İÇİN SİNİR AĞ MODELLERİ

Yasin EŞREF

Yüksek Lisans, **Bilgisayar Mühendisliği**
Danışman: Dr. Öğr. Üyesi Burcu CAN BUĞLALILAR
Haziran 2019, 88 sayfa

Türkçe gibi sondan eklemeli dillerde aynı kökten birçok kelimenin türetilmesinden dolayı kelimelerin bir bütün olarak modellenmesi seyreklik problemini de beraberinde getirmektedir. Bundan ötürü, kelimeyi bir bütün olarak ele almaktansa karakterleri üzerinden bir kelimeyi ifade etmek ya da morfem ve morfemin etiket bilgisini hesaba katmak kelime hakkında daha detaylı bilgi vermekte ve seyreklik problemini de azaltmaktadır.

Bu çalışmada Türkçede kelime dizilerini etiketleme problemleri için derin sinir ağlarını kullanan bir model önerilmiştir. Seyreklik problemini çözmek için kelimelerin karakter ve morfem bilgilerin dizi etiketleme problemi kapsamındaki etkisi incelenmiştir.

Bu çalışmada literatürdeki güncel derin öğrenme modelleri Türkçe için farklı kelime veya alt-kelime düzeyinde gösterimler kullanılarak Varlık İsmi Tanıma ve Sözcük Türü İşaretleme problemleri için uygulanmıştır. Sonuçlar, kelimelerin morfem bilgisinin kullanılmasının Türkçede dizi etiketlemeyi iyileştirdiğini göstermektedir. Ayrıca kelime dizilerinin etiketlerini bulurken komşu kelimelerin bilgilerinden de yararlanılarak doğruluğu daha yüksek sonuçlar elde edilmiştir.

Anahtar Kelimeler: doğal dil işleme, dizi etiketleme, varlık ismi tanıma, sözcük türü işaretleme, derin öğrenme, dikkat ağları

ABSTRACT

NEURAL MODELS FOR TURKISH SEQUENCE LABELING

Yasin EŞREF

Master of Science, **Computer Engineering Department**
Supervisor: Asst. Prof. Dr. Burcu CAN BUĞLALILAR
June 2019, 88 pages

Because of the inflection of many word forms from the same root in agglutinative languages such as Turkish, modeling the words as a whole causes sparsity problem. Therefore, rather than handling the word as a whole, expressing a word through its characters or considering the morpheme and morpheme label information gives more detailed information about the word and therefore mitigates the sparsity problem.

In this study, a model using deep neural networks is proposed for the sequence labeling task in Turkish. To cope with the sparsity problem, character and morpheme information is used and the effect of this information on sequence labeling problem is examined. The existing deep learning models are applied using different word or sub-word representations for Named Entity Recognition (NER) and Part-of-Speech Tagging (POS Tagging) in Turkish. The results show that using morpheme information improves the sequence labelling in Turkish. Moreover, more accurate results are obtained by using the contextual information in the model.

Keywords: natural language processing, sequence labeling, named entity recognition, part-of-speech tagging, deep learning, attention machanism

TEŞEKKÜR

İlk olarak yüksek lisans araştırma ve tez çalışmalarımı yaptığım 2 yıl boyunca değerli bilgilerini ve tecrübesini benimle sabırla ve hoşgörüyle paylaşan, çalışmalarım boyunca yol gösteren danışmanım Sayın Dr. Öğr. Üyesi Burcu CAN BUĞLALILAR'a,

Bu tez için yaptıkları değerli yorumları sebebiyle tez jüri üyeleri Prof. Dr. İlyas ÇİÇEKLİ'ye, Doç. Dr. Cengiz ACARTÜRK'e, Dr. Öğr. Üyesi Gönenç ERCAN'a ve Dr. Öğr. Üyesi Özkan KILIÇ'a,

Sevgilerini ve desteklerini benden hiç bir zaman esirgemeyen ve güvenleriyle bana güç veren başta annem Gülseren EŞREF ve babam Mehmet EŞREF olmak üzere çok sevdiğim aileme,

Son olarak hayatıma girdiği günden bu yana bana her zaman inanan, her konuda arkamda olan, tez çalışmalarımı yaparken beni her daim motive eden, en büyük destekçim, hayat arkadaşım, çok sevgili eşim Nurdan EŞREF'e teşekkürlerimi sunarım.

İçindekiler

ÇİZELGELER

ŞEKİLLER

KISALTMALAR

NLP	Doğal Dil İşleme (Natural Language Processing)
NER	Varlık İsmi Tanıma (Named Entity Recognition)
RNN	Tekrarlayan Sinir Ağı (Recurrent Neural Network)
LSTM	Uzun-Kısa Süreli Bellek (Long-short Term Memory)
Bi-LSTM	Çift yönlü Uzun-Kısa Süreli Bellek (Bidirectional Long-short Term Memory)
GRU	Kapılı Tekrarlayan Hücre (Gated Recurrent Unit)
CNN	Evrişimli Sinir Ağı (Convolutional Neural Network)
CBOW	Sürekli Kelime Çantası(Continuous Bag-of-words)
CRF	Koşullu Rastgele Alan (Conditional Random Fields)
HMM	Saklı Markov Modeli (Hidden Markov Model)
MEMM	Maksimum Entropi Markov Modeli (Maximum Entropy Markov Model)

1. GİRİŞ

Bu çalışma kapsamında Türkçe için dizi etiketleme problemlerinde mevcut sonuçların üzerine çıkacak bir model önerilmektedir. Önerilen model son yıllarda doğal dil işleme de dahil olmak üzere birçok alanda sıkça kullanılmaya başlayan derin öğrenme yöntemleri ile geliştirilmiştir. Önerilen model varlık ismi tanıma ve sözcük türü işaretleme olmak üzere iki farklı dizi etiketleme problemi üzerinde denenmiş ve başarılı sonuçlar alınmıştır.

1.1. Dizi Etiketleme Problemleri

Bir kelime dizisinin her bir elemanına kategorik bir etiketin atanması ile ilgilenen doğal dil işleme problemlerine dizi etiketleme problemleri denmektedir. Kelimelerin isim, sıfat, eylem gibi sözdizimsel kategorilerinin bulunması (part-of-speech tagging), bir cümlede geçen varlık isimlerinin bulunması (named entity recognition), bir cümledeki isim ve fiil tamlamalarının bulunması (chunking) ve cümledeki eksik kelimelerin bulunması gibi doğal dil işleme (NLP) problemleri genel olarak dizi etiketleme problemleri olarak ele alınmaktadır. Dizi etiketleme problemleri birçok doğal dil işleme uygulamasının ilk adımı olduğundan bu problemlerin çözümlerinde yapılacak iyileştirmeler bu uygulamaların sonuçlarının da iyileşmesine katkı sağlayacaktır.

1.1.1. Varlık İsmi Tanıma

Varlık ismi tanıma (NER) cümlelerde geçen kişi, yer, organizasyon isimleri ile zamansal ve parasal ifadeleri bulup bu kelimeleri önceden tanımlanmış kategorilere atama problemidir. Ancak bu tez çalışmasında kullanılan veri kümeleri sadece ENAMEX tipindeki etiketleri (kişi, konum ve organizasyon etiketler) içermektedir.

Ayrıca varlık isimlerini etiketlemede kullanılan BIOES (Begin, Inside, Outside, End ve Single kelimelerinin baş harfleri) gibi notasyonlar bulunmaktadır. Kelimeler bu harflerden birisi ve eğer varsa etiketin tipini ifade eden kişi (PERSON), konum (LOCATION) ve

organizasyon (ORGANIZATION) bilgisi ile oluşturulmuş etiketler ile etiketlenmektedir. Bu notasyon ile etiketlenmiş varlık isimlerine bir örnek aşağıda verilmiştir:

- Berlin'deki(S_LOCATION) pansiyon(O) sahibesi(O) verdiği(O) cevapta(O) Frau (B_PERSON) van(I_PERSON) Tiedemann'ın(E_PERSON) artık(O) onların(O) yanında(O) oturmadığını(O) bildiriyordu(O) .(O)

Bu tez kapsamında kullanılan BIO notasyonunda ise S ve E harfleri kullanılmamakta varlık isimleri için ilk kelimede B, sonraki kelimelerde I harfi, varlık ismi olmayan kelimelerde ise O harfi kullanılmaktadır. Bu notasyonları kullanmanın bir sebebi de varlık isimlerinin sınırlarının belirlenmesidir.

Varlık isimlerinin bulunması, soru cevaplama (question answering), bilgi çıkarımı (information extraction, information retrieval) gibi birçok doğal dil işleme uygulaması için faydalı bilgiler sağladığından, bu uygulamaların ön adımı olarak da kullanılmaktadır.

1.1.2. Sözcük Türü İşaretleme

Kelimelerin herhangi bir bağlamdaki kullanımlarını ifade eden isim, sıfat, eylem gibi sözdizimsel kategorilerinin bulunması sözcük türü işaretleme problemi olarak ifade edilmektedir. Varlık ismi probleminde olduğu gibi sözcük türü işaretleme de doğal dil işleme alanındaki birçok uygulamada kulllanılmakta, bu uygulamaların ilk adımını oluşturmaktadır.

Aşağıdaki bu tez çalışmasında kullanılan veri kümesinden [1] alınmış, sözcük türleri işaretlenmiş örnek bir cümle verilmiştir:

- Karşısında(Noun) ,(Punc) pantolonu(Noun) dizlerine(Noun) dek(Postp) ıslak(Adj) ,(Punc) önlük(Noun) torbası(Noun) ham(Noun) eriklerle(Noun) dolu(Adj) İbrahim (Noun) dikiliyordu(Verb) .(Punc)

1.2. Motivasyon

Bu tez çalışmasına başlamamızda bizi motive eden sebepler şunlardır:

- Dizi etiketleme problemlerinin doğal dil işleme alanındaki birçok uygulamanın ilk adımını teşkil ediyor olması bu adımda meydana gelecek olan iyileştirmelerin diğer uygulamaları da etkileyeceği düşünülmüştür.
- Son yıllarda başta İngilizce olmak üzere birçok dilde dizi etiketleme problemleri ile ilgili derin öğrenme yöntemleri kullanan çalışmalar yapılmış olmasına rağmen Türkçe için bu alanda henüz fazla çalışma yapılmamıştır.
- Türkçenin sondan eklemeli bir dil olması ve aynı kökten birden fazla kelime türetilebilmesinin neden olduğu seyreklik probleminden dolayı doğal dil işleme çalışmalarındaki başarının aynı problemde çalışılmış diğer dillerden daha geride olduğu görülmüştür.
- Dizi etiketleme problemleri üzerine yapılan çalışmalar daha ziyade sadece bir probleme odaklanmakta ve probleme özgü kaynaklara ihtiyaç duymaktadır.

Tüm bu sebepler göz önünde bulundurulduğunda Türkçede dizi etiketleme problemlerini çözmek için herhangi bir ek kaynağa ihtiyaç duymayan bir model önerilmiştir. Önerilen modelin herhangi bir probleme özgü değil tüm dizi etiketleme problemlerine uygun olması amaçlanmıştır. Modelin geliştirilmesinde güncel derin öğrenme yöntemleri kullanılmıştır. Türkçedeki seyreklik problemini çözmek için kelimelerin karakter ve morfem bilgileri de modele dahil edilmiştir.

Tüm bu hususlar göz önünde bulundurularak geliştirilen model, denendiği iki farklı problem için mevcut çalışmalardan çok daha iyi sonuçlar vermiş ve güncel en iyi sonuç olmuştur. Bu tez kapsamında bizim katkımız şu 3 başlık altında toplanabilir:

- Türkçede uygulanmamış güncel dizi etiketleme modellerini Türkçenin morfolojik yapısına göre uyarlamak

- Kelimelerin karakter ve morfem bilgilerini modele ekleyerek Türkçedeki seyreklik problemini gidermek
- Modele eklenecek olan farklı dikkat ağı mimarileri sayesinde kelime etiketlerinin bulunmasında hem o kelimenin morfemlerinden hem de çevresindeki kelimelerden daha fazla bilgi alınması sağlayarak daha başarılı bir model elde etmek

1.3. Araştırma Soruları

- Türkçe dizi etiketlemede kelimelerin morfem ve karakter bilgisini kullanmak, dizi etiketleme başarısını etkiler mi?
- Derin öğrenme ile farklı kelime gosterimleri kullanarak doğruluğu yüksek bir şekilde varlık ismi tanıma veya sözcük türü işaretleme gerçekleştirilebilir mi?
- Morfem seviyesinde kullanılacak olan bir dikkat ağı ile önemi daha fazla olan morfemlere daha fazla dikkat verilerek kelime etiketinin bulunmasında daha başarılı sonuçlara ulaşılabilir mi?
- Kelimelerin bağlam bilgilerini kullanmak dizi etiketlemedeki başarıyı artırır mı?

1.4. Tezin Yapısı

Tez kapsamında önerilen modelde kullanılan derin öğrenme yöntemleri ve bu yöntemlere yardımcı olarak kullanılan istatistiksel yöntemler Bölüm 2.'de verilmiştir. Dizi etiketleme ve kullanılan yöntemler ile ilgili literatürde yer alan daha önceki çalışmalara Bölüm 3.'de değinilmiştir. Bölüm 4.'de önerilen model açıklanmıştır. Bu model üzerinde yapılan deneylerin sonuçlarına ve literatürdeki diğer çalışmalarla yapılan karşılaştırmalara ise Bölüm 5.'de yer verilmiştir. Son olarak Bölüm 6.'da sonuç değerledirmesi yapılarak tez tamamlanmıştır.

2. ALAN BİLGİSİ

Doğal dil işleme problemleri uzun yıllar istatistiksel yöntemler ile çözülmeye çalışılmıştır [2–4]. Kimi çalışamalarda yapay sinir ağlarının kullanıldığı da olmuştur [5]. Sonraları derin öğrenme yöntemlerinin diğer birçok alanda olduğu gibi doğal dil işleme alanında da kullanılması ve yüksek başarılı sonuçlar elde edilmesiyle birlikte günümüzdeki çoğu çalışmada çözüm yöntemleri olarak derin sinir ağları tercih edilmeye başlanmıştır. Derin sinir ağlarının başarısı göz önünde bulundurularak bu tez kapsamında yapılan çalışmalarda da istatistiksel yöntemler yerine derin öğrenme yöntemleri kullanılmıştır.

Tezin bu bölümünde dizi etiketleme problemine çözüm olarak literatürde yaygın kullanılan derin öğrenme yöntemleri ve dizi etiketlemede kullanım örneği pek olmasa bile bu tez çalışması kapsamında önerilen modellerde kullanılan derin öğrenme yöntemleri açıklanacaktır. Ayrıca önerilen modele yardımcı olması amacıyla kullanılmış istatistiki yöntemlerden birisi olan koşullu rastgele alana (CRF) da bu bölümde kısaca değinilecektir.

Öncelikle kelime gösterimleri için kullanılan kelime vektörleri ve kelime vektörlerinin kısıtlarını ortadan kaldırmak veya azaltmak için kullanılan karakter ve morfem vektörleri açıklanacaktır. Sonrasında dizisel problemlerin çözümünde yaygın tercih edilen, bu çalışmada da kelime dizilerindeki özelliklerin çıkarılması amacıyla kullanılan tekrarlayan sinir ağları (RNN) ve bunun bir türevi olan uzun kısa süreli bellek ağları (LSTM) açıklanacaktır. Daha sonra karakter vektörlerinin elde edilmesinde kullanılan evrişimli sinir ağlarından (CNN) bahsedilecektir. Sonrasında yine son yıllarda görüntü işleme alanından doğal dil işlemeye kadar birçok alanda sıkça kullanılmaya başlanan ve modelin veride nereye daha çok dikkat etmesi gerektiğini öğrenmeye çalışan dikkat ağları açıklanacaktır. Son olarak sınıflandırma problemlerinde kullanıldığında sonuçlardaki başarıyı artırdığı görülen ve bizim tezimizde de kullandığımız koşullu ratgele alan (CRF) açıklanacaktır.

2.1. Kelime Vektörleri

Doğal dil işleme problemleri üzerine çalışırken cümleleri oluşturan kelimelerin matematiksel olarak ifade edilmesi gerekir. Böylece kelimeler üzerinde matematiksel ve istatistiksel işlemler gerçekleştirilebilir. Bu amaçla kelime sözlüğü yardımıyla tanımlanan kelime gösterim yöntemlerine genel olarak kelime vektörleri denilmektedir.

Önceleri kelime gösterimleri için sayı vektörü, TF-IDF ve ortak oluşum (co-occurrence) vektörleri gibi frekans bazlı yöntemler kullanılmaktaydı. Bu yöntemler ile oluşturulan kelime vektörleri son derece büyük boyutta ve seyrek vektörler olmaktaydı. Dolayısı ile bu vektörleri kullanmak hem performans açısından sorun teşkil ediyor hem de yapılan çalışmaların sonuçlarını bir noktada sınırlandırıyordu.

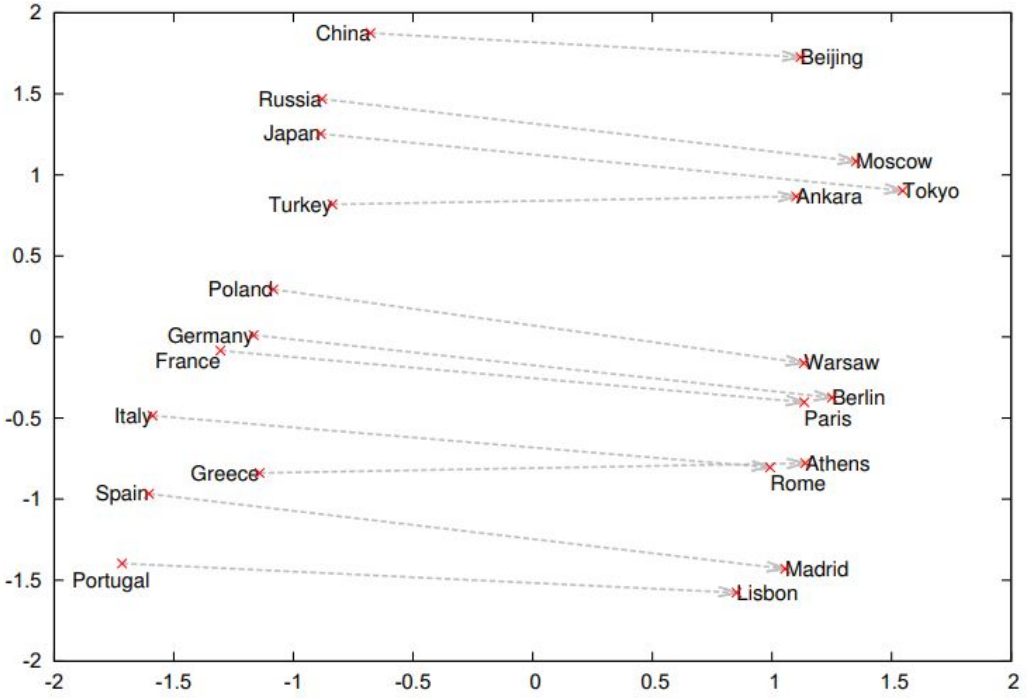
Mikolov'un 2013 yılında yapmış olduğu çalışma bu sınırların ortadan kalkmasını sağlamıştır [6]. Bu çalışmada kelimelerin daha sık ve daha kısa vektörler halinde gösterilmesini sağlayan bir yöntem geliştirilmiştir. Genel olarak Word2Vec olarak adlandırılmakta olan bu yöntem ortaya koyduğu iyi sonuçlar sayesinde kısa süre içerisinde tüm doğal dil işleme problemlerinde yaygın bir şekilde kullanılmaya başlanmıştır.

2.1.1. Word2Vec

Word2Vec büyük miktardaki yapısal olmayan metin verisinden kelimelerin vektör gösterimlerini çıkarmak için kullanılan tahmin tabanlı ve denetimsiz bir öğrenme yöntemidir. Kelimelerin bu yöntemle vektör uzayında dağılımsal gösterimleri benzer kelimeleri gruplayarak doğal dil işleme görevlerindeki öğrenme algoritmalarının daha iyi sonuçlar almasına yardımcı olmaktadır [7].

Birbirleri ile ilişkili olan kelimelerin Word2Vec ile bulunmuş olan kelime vektörleri de vektör uzayında birbirlerine yakın olmaktadır. Benzer bir şekilde anlamsal olarak arasındaki farklar aynı olan kelimelerin vektörleri arasındaki fark da benzer çıkmaktadır. Örneğin kral-kraliçe kelimeleri arasındaki vektörel fark ile erkek-kadın kelimeleri arasındaki vektörel fark da benzer çıkmaktadır. Yani "kral" kelimesinin vektöründen "kraliçe" kelimesinin vektörü çıkarılıp "erkek" kelimesinin vektörü eklendiğinde vektör uzayında

"kadın" kelimesine ait vektöre çok yakın bir vektör elde edilmektedir. Şekil 2.1.'de Mikolov'un çalışmasında yer alan buna benzer bir örneğe yer verilmiştir.



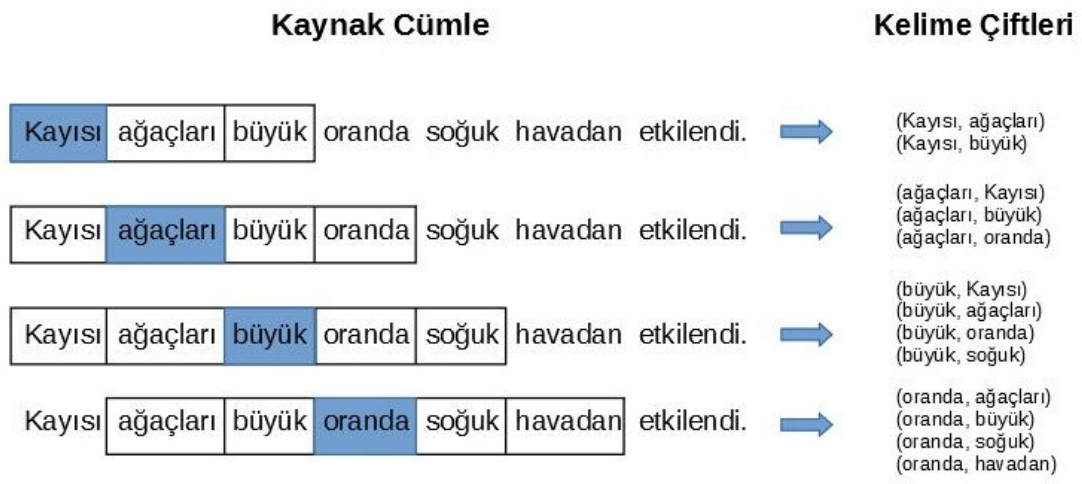
ŞEKIL 2.1. Ülke ve başkent vektörlerinin PCA ile 2 boyutta projeksiyonu [7]

Word2Vec'te Kelime Vektörlerinin Çıkarımı

Kelime vektörlerini öğrenmek için kullandığımız Word2Vec tek bir algoritma değil CBOW (sürekli kelime torbası) ve Skip-gram ismi verilen iki farklı tekniğin birleşimidir. Her iki yöntem de aslında kelimeleri başka kelimeler ile eşleştiren sığ sinir ağlarıdır. Fark sadece hangi kelimelerin girdi hangi kelimelerin çıktı olarak belirlenecek olmasıdır. CBOW dizideki bir kelimeyi kelimenin etrafındaki kelimelerle tahmin ederek kelimenin vektörünü çıkarmaya çalışırken, Skip-gram bir kelimeye göre etrafındaki kelimeleri tahmin etmeya çalışarak kelime vektörünü çıkarmaktadır.

Kelime vektörlerinin bu yöntemlerle öğrenilmesi için sinir ağlarında bu kelimelerin matematiksel olarak ifade edilmesi gerekmektedir. Bu amaçla sıcak kodlama (one-hot) vektörleri kullanılmaktadır. Sıcak kodlanmış vektörler her kelime için uzunluğu toplam kelime sayısı kadar olan ve sadece o kelimeye ait indisteki değeri 1, diğer indislerdeki değerleri sıfır olan vektörlerdir.

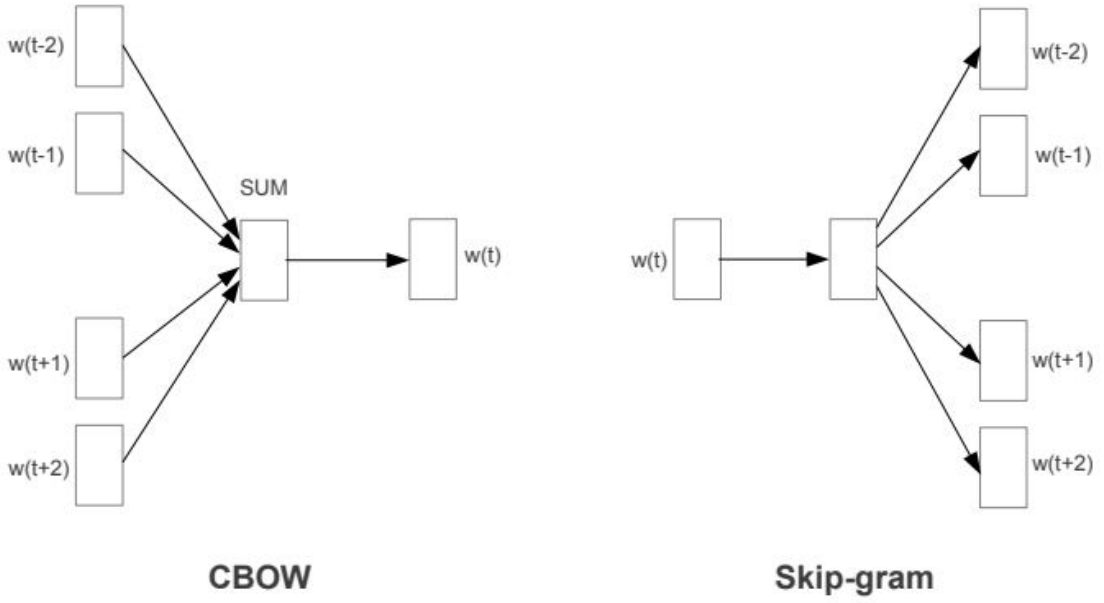
Kelime vektörlerinin öğrenilmesi için eğitilecek olan veri kümesindeki cümlelerden belirli bir komşuluğu olan kelime çiftlerinin de çıkarılması gerekmektedir. Bu belirli bir komşuluk ifadesine pencere boyutu parametresi de denilmektedir. Örneğin pencere boyutu parametresi 2 olarak belirlenen bir çalışmada her kelimenin öncesinde ve sonrasında gelen 2'şer kelime ortadaki kelime ile çiftler oluşturacak şekilde çıkarılır. Bunun bir örneğine Şekil 2.2.'de yer verilmiştir. Bu şekildeki mavi kutu içerisinde olan kelimeler o kelimeye ait kelime çiftleri için merkezdeki kelimeyi ifade etmektedirler.



ŞEKIL 2.2. Skip-gram modelinde eğitilecek kelime çiftlerinin çıkarımı

Veri kümesindeki tüm kelimelerin tek sıcak kodlanmış (one-hot) vektörleri ve tüm cümlelerdeki kelime çiftleri çıkarıldıktan sonra kelime vektörlerinin elde edileceği sığ yapay sinir ağı oluşturulabilecektir. Bu sinir ağının girdi ve çıktılarını kelime çiftlerinin sıcak kodlanmış vektörleri oluşturacaktır. Eğer Skip-gram algoritması kullanılıyorsa kelime çiftlerinden merkezdeki kelime girdi, diğer kelime çıktı olacak, CBOW yöntemi kullanılıyor ise merkezdeki kelimeler çıktı , diğer kelimeler ise girdi olacaktır.

Oluşturulacak olan 3 seviyeli bu sinir ağının tek gizli katmanının düğüm sayısı da oluşacak olan kelime vektörlerinin boyutunu ifade eder. Bu ağ eğitildikten sonra gizli katmanının hesaplanmasında kullanılan ağırlık matrisi üzerinde çalıştığımız veri kümesinin kelime vektörleri olacaktır. Şekil 2.3.'de her iki algoritma için de sinir ağ modeline yer verilmiştir. Burada $w(t)$ derlemdeki t. kelimeyi ifade etmektedir.



ŞEKIL 2.3. CBOW ve Skip-gram yöntemleri ile kelime vektörlerinin çıkarımı [6]

Kelime vektörleri bu iki yöntemden birisi kullanılarak elde edildikten sonra doğal dil işleme alanında yapılacak çalışmalara girdi olarak verilmekte ve çoğu zaman tekrar eğitilmesine gerek kalmadan kullanılabilmektedirler. Nitekim önceden eğitilmiş kelime vektörleri doğal dil işleme alanındaki birçok problemde kullanılmış ve daha iyi sonuçlar elde edilmesini sağlamıştır.

2.1.2. Diğer Kelime Vektörü Yaklaşımları

Mikolov ve diğ. (2013) [7] çalışmasından sonra kelime vektör gösterimleri üzerine çok sayıda çalışma yapılmıştır. Bu çalışmalardan birisinde Pennington ve diğ. (2014) [8] **GloVe** adını verdikleri yeni bir kelime vektör gösterimini tasarlamıştır. Bu yaklaşımda kelime vektörleri oluşturulurken Word2Vec'deki gibi kelimeye ait sadece lokal bilgilerin kullanılması yerine kelimenin tüm veri kümesindeki global istatistiklerine de bakılmaktadır. GloVe bunu sağlamak için eşdizimlilik (co-occurrence) matrisini kullanmaktadır.

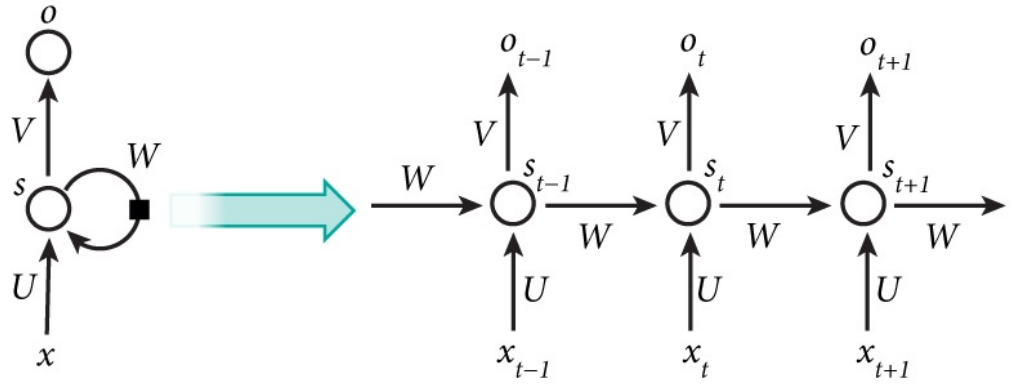
Facebook Yapay Zeka ekibinden Bojanowski ve diğ. (2017) [9] yapmış oldukları çalışmada **FastText** adını verdikleri bir kelime gösterim yöntemini ortaya atmıştır. Hem Word2Vec hem de GloVe yaklaşımında en küçük birim kelime olarak kabul ederken bu

çalışmadaki yaklaşımda kelimelerin alt kümesi olan karakter n-gram'ları en küçük birim olarak kabul edilmiştir. Yani bir kelimenin vektörü aslında bu kelimenin karakter n-gram'larının vektörlerinden hesaplanmaktadır. Dolayısıyla veri kümesinde nadir geçen kelimeler veya hiç bulunmayan kelimelerde FastText gerçeğe daha yakın vektörler üretmektedir.

2.2. Tekrarlayan Sinir Ağları

2.2.1. Tanım

Geleneksel yapay sinir ağlarında tüm girdiler ve çıktılar birbirinden bağımsızdırlar. Bu sebeple cümlelerdeki kelimeler gibi birbirleri ile ilişkili olan dizisel girdi veya çıktıların olduğu problemlerde geleneksel sinir ağ modelinden farklı bir modele ihtiyaç bulunmaktadır. Tekrarlayan sinir ağları (RNN) da bu sorunu çözen bir model olarak ortaya çıkmıştır. Tekrarlayan sinir ağı önceki adımdan elde edilen çıktının mevcut adıma girdi olarak verildiği bir yapay sinir ağ modelidir.



ŞEKIL 2.4. RNN ağı ve çalışma anındaki açılmış hali

Şekil 2.4. bir RNN ağının çalışma anındaki açılmış halini göstermektedir. Bu ağ dizinin uzunluğu kadar açılacaktır. Örneğin 10 uzunluğundaki bir cümle için açıldığında her kelime için bir katmanın olduğu 10 katmanlı bir sinir ağı olacaktır. Bu ağda x_t'ler girdileri, s_t'ler gizli durumları, o_t'ler ise t adımının çıktısını ifade etmektedir.Gerçekleştirilen

işlemler matematiksel olarak aşağıdaki gibi ifade edilir:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{1}$$

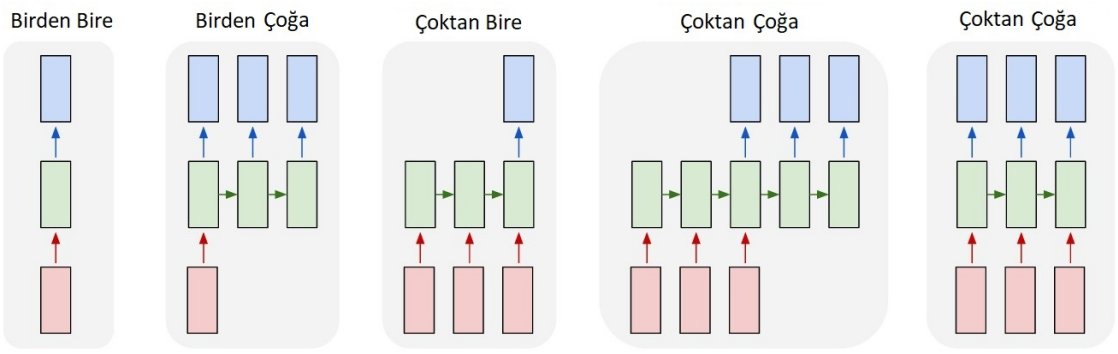
$$o_t = softmax(Vs_t) \tag{2}$$

Gizli durumları ifade eden s_t değerini hesaplamak için $t-1$ adımının gizli durum çıktısı (s_{t-1}) ve t adımının girdisi (x_t) kullanılır. Dolayısı ile t anındaki gizli durum o adıma kadarki tüm girdiler kullanılarak hesaplandığı için ağın o andaki hafızası gibi de düşünülebilmektedir. t adımındaki girdiler (s_{t-1} ve x_t) modelin ağırlık parametreleri (U, W) ile çarpılıp *tanh* veya *ReLU* gibi bir aktivasyon fonsiyonundan geçirilerek s_t değeri hesaplanır.

t adımının çıktısı olan o_t ise büyük ölçüde s_t değerine bağımlıdır ve bu değerin bir başka model parametresi (V) ile çarpıldıktan sonra *softmax* fonksiyonundan geçirilmesi ile bulunur.

RNN ağları geleneksel sinir ağlarından farklı olarak her seviyede aynı parametreleri (U, W, V) ortaklaşa kullanır. Her adımda değişen sadece girdilerdir. Yani RNN aynı işlemi dizideki her eleman için bir önceki adımın çıktısını da kullanarak tekrarlar.

Tekrarlayan sinir ağlarının geleneksel sinir ağlarına göre bir diğer avantajı da sadece bir girdiyi bir çıktıya eşleme işlemi için değil aynı zamanda birden çoğa veya çoktan çoğa gibi çoklu girdi ve/veya çoklu çıktı işlemlerinin de yapılabilmesidir. Örneğin makine çevirisi problemleri çoklu girdinin ve çoklu çıktının olduğu, tekrarlayan sinir ağı ve türevleri ile çözülebilen bir problemdir.



ŞEKIL 2.5. Problem türüne göre farklı RNN açılımları [10]

2.2.2. Çift yönlü RNN

Bazı problemlerde bir t anındaki çıktı sadece kendisinden önceki adımların durumlarına değil aynı zamanda kendisinden sonra gelen adımların durumlarına da bağlı olabilmektedir. Örnek olarak cümlelerdeki kelimeler sadece kendisinden önce gelmiş kelimelerle değil kendisinden sonra gelecek olan kelimelerle de anlamsal veya sözdizimsel olarak bağımlı olabilmektedir.

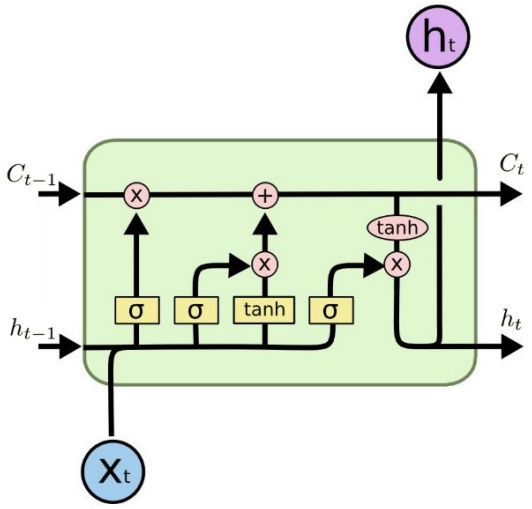
Bu tarz problemleri çözmek için ileri yönlü oluşturulmuş tekrarlayan sinir ağına ek olarak bir de ters yönlü çalışan tekrarlayan sinir ağı oluşturulmaktadır. Bu geri yönlü ağın çıktıları ileri yönlü olan ağın çıktıları ile birleştirilerek t adımının sonucu bulunmaktadır. Bu tarz iki yönlü de bağımlılıklara bakılan tekrarlayan sinir ağına çift yönlü RNN denilmektedir. Doğal dil işleme problemlerinde problemin doğası gereği çoğu zaman çift yönlü ağlar tercih edilmektedir.

2.2.3. Uzun Kısa Süreli Bellek Ağları

Tekrarlayan sinir ağının öğrenimi sırasında her adımda türev alınarak optimum noktaya ulaşılmaya çalışılmaktadır. Ancak öğrenilen diziler çok uzadığında arka arkaya bu türevleri almak Kaybolan ya da Patlayan Eğim (Vanishing Gradient, Exploding Gradient) problemlerine neden olmakta yani bir noktadan sonra ya öğrenme çok yavaşlamaya ya da alakasız, önemsiz noktalar öğrenilmeye başlamaktadır. Bu sebeple uzun cümlelerde aralarındaki mesafe fazla olan kelimelerin arasındaki ilişki tekrarlayan sinir ağları ile çok iyi öğrenilememektedir. Bu yüzden tekrarlayan sinir ağları uzun cümlelerde genelde istenildiği kadar iyi sonuç vermezler.

Bu bahsedilen sorun uzun kısa süreli bellek ağları ile çözülebilmektedir. Uzun kısa süreli bellek ağları (LSTM) tekrarlayan sinir ağ modelinin uzun süreli bağımlılıkları da öğrenebildiği özel bir türevidir [11]. Uzun kısa süreli bellek ağının farkı; gizli durumları (s_t) hesaplarken kullandığı fonksiyonlar ve yöntemlerdir. LSTM ağında s_t değeri her adımdaki LSTM hücresi içerisinde hesaplanmaktadır. LSTM hücresi ağın o adımdaki belleği olarak tanımlanabilmektedir. Bir LSTM hücresinin içerisindeki genel akışa Şekil 2.6.'de

verilmiştir. Bu şekilde C_t değeri hücrenin durumunu, h_t değeri ise hücrenin gizli durumunu ifade etmektedir.



ŞEKIL 2.6. Bir LSTM hücresinin içerisindeki kapıların ve genel akışın gösterimi [12]

Her bir LSTM hücresi içerisinde önceki adımlardan gelen bilgilerden hangisinin unutulacağı hangisinin hatırlanacağına karar vermek gibi farklı işlemleri yerine getiren kapılar bulunmaktadır. Girdi kapısı (i_t), çıktı kapısı (o_t), unutma kapısı (f_t) gibi isimler verilen bu kapılar sayesinde LSTM ağı arasındaki mesafe fazla olan girdiler arasındaki ilişkileri de yakalayabilmekte, alakasız ve anlamsız noktaları fazladan öğrenmemektedir. Her bir hücre içerisinde gerçekleştirilen işlemler adım adım açıklanacaktır.

Hücre içerisindeki ilk adım hücrenin durumundan hangi bilgilerin atılacağına karar vermektir. Bu işlem unutma kapısı adı verilen kapı içerisinde Formül (3) ile gerçekleştirilmektedir:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{3}$$

Bir önceki adımın gizli durumu (h_{t-1}) ve o adımdaki hücre girdisi (x_t) bu işlemin girdilerini oluşturmaktadır. W_f ve b_f ise bu adımda kullanılacak olan ağırlık ve ek girdi (bias) değerleridir. İşlem sonucunu (f_t) hesaplamak için 0 ile 1 arasında değer üreten sigmoid (σ) aktivasyon fonksiyonu kullanılmaktadır. Bu fonksiyonun sonucunun 1 değerine yakınlaşması gelen bilgiyi tutmak, 0 değerine yakınlaşması ise gelen bilgiyi unutmak anlamına gelmektedir.

Hücre içerisinde gerçekleştirilecek bir sonraki adım hücrenin durumunda, gelen yeni bilgilerden hangisini tutacağımıza karar vermektir. Bu işlem iki adımda gerçekleştirilmektedir. İlk olarak girdi kapısı içerisinde hangi değerlerin güncelleneceğine karar verilirken (4) ayrı bir $tanh$ katmanında ise hücre durumuna eklenecek yeni aday bilgiler ($\widehat{C}_t$) tespit edilmektedir (5). Her iki işlem de girdi olarak unutma kapısına benzer bir şekilde bir önceki adımın gizli durumunu (h_{t-1}) ve o adımdaki hücre girdisini (x_t) almaktadırlar. W_i ve W_c bu adımların ağırlık parametreleri, b_f ve b_c ise ek girdi değerleridir. Bu iki adım sonucunda elde edilen değerler (i_t ve $\widehat{C}_t$) daha önceden elde edilmiş unutma kapısı sonucu (f_t) da kullanılarak Formül (6)'deki gibi birleştirilecektir. Böylece hücrenin eski durumu (C_{t-1}) yeni durumuna (C_t) geçmiş olacaktır:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{4}$$

$$\widehat{C}_t = tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{5}$$

$$C_t = f_t \times C_{t-1} + i_t \times \widehat{C}_t \tag{6}$$

Unutma ve girdi kapılarından sonraki diğer bir kapı da çıktı kapısıdır. Bu kapı sadece karar verilmiş olan bilgilerin çıktıya verilmesini sağlamaktadır. Bu kapı da diğer kapılar gibi girdi olarak bir önceki adımın gizli durumunu (h_{t-1}) ve o adımdaki hücre girdisini (x_t) almaktadır. W_o ve b_o bu adımda kullanılacak olan ağırlık ve ek girdi değerleridir. Formül (7)'de gösterildiği gibi çıktı kapısının sonucu (o_t) da sigmoid aktivasyon fonksiyonu ile bulunmaktadır. LSTM hücresinin çıktısı (h_t) çıktı kapısından gelecek olan bilgilere (o_t) ve hücrenin Formül (6) ile bulunmuş olan durumuna (C_t) bağlıdır (8):

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{7}$$

$$h_t = o_t \times tanh(C_t) \tag{8}$$

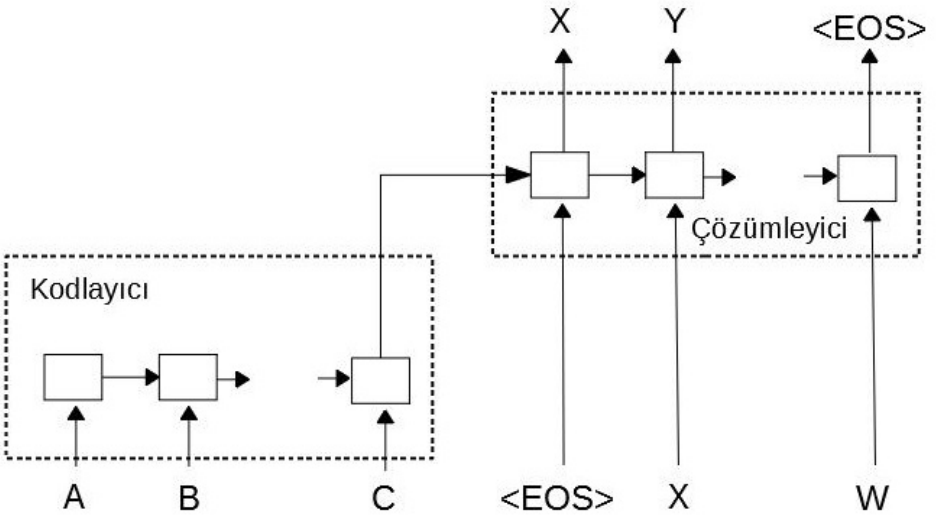
2.2.4. Diziden Diziye Modeli

Girdi olarak bir dizi alan bir başka öğe dizisini çıktı olarak veren modellere diziden diziye (sequence to sequence) model denilmektedir. Bu model aslında daha önce bahsedilen çoktan çoğa tekrarlayan sinir ağ modelinin bir çeşididir. Ancak özellikle dikkat ağlarının

anlatılacağı bölümde bu modelden de sıkça bahsedileceği için ayrı bir başlıkta ele alınmıştır.

Çoktan çoğa tekrarlayan sinir ağ modelleri denildiğinde genelde tek bir RNN (veya türevleri olan LSTM, GRU) yığını anlaşılmaktadır. Bu tarz modellerde tekrarlayan sinir ağ modeli bir diziyi girdi olarak alırken bir başka diziyi çıktı olarak vermektedir. Örneğin girdi olarak cümlenin kelime vektörlerini çıktı olarak kelimelerin etiket bilgilerini üretmeye çalışan varlık ismi tanıma problemleri genelde bu yöntemle modellenmektedir.

Diğer taraftan diziden diziye olan modellerde ise iki RNN (veya türevleri olan LSTM, GRU) yığını kullanılır. İlk RNN yığını girdi dizisini sabit uzunlukta bir vektöre kodlar, ikinci RNN ise kodlanan vektörü alarak çıktıyı üretecek şekilde çözümler. Bu sebeple bu modele Kodlayıcı-Çözümleyici modeli de denmektedir. Diziden diziye modeli çoktan-bire ve birden-çoğa tekrarlayan sinir ağ modellerinin birleştirilmiş hali gibi düşünebilir.



ŞEKIL 2.7. Diziden diziye (kodlayıcı-çözümleyici) model görünümü. <EOS>, kelime başı ve sonunu ifade eden birer sembol olacak şekilde kullanılmıştır.

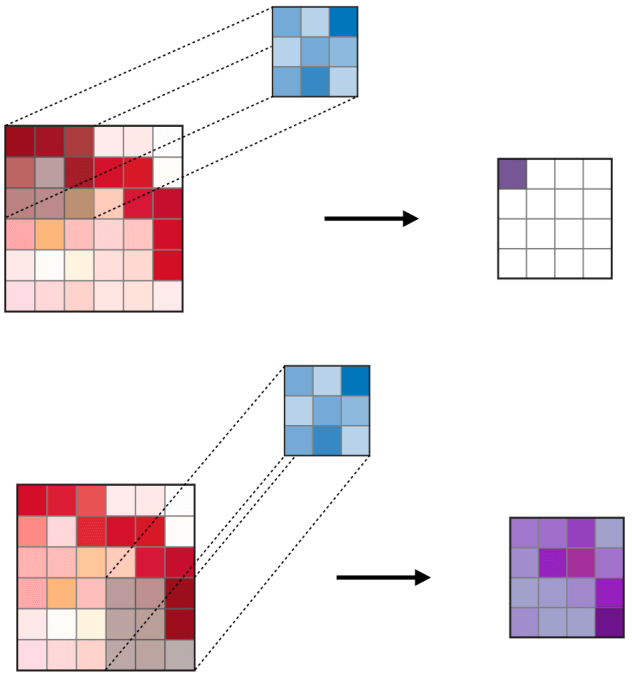
Bu yöntem makine çevirisi gibi problemlerin çözümünde sıkça tercih edilmektedir. Ancak makine çevirisinde bazı kısıtlara da neden olabilmektedir. Bu modelde ilk RNN (kodlayıcı) giriş dizisini, ikinci RNN'in (çözümleyici) tam bir çeviri oluşturması için sabit boyutlu bir vektör olarak sıkıştırmaktadır. Bu vektörün çevrilecek cümleye ait her detayı, her bilgiyi içermesi gerekecektir. Bu vektörün uzunluğu cümleden cümleye değişmediği için

vektör uzunluğu herhangi bir uzunluktaki cümlenin sıkıştırılabileceği büyüklükte olmalıdır. Çalışmalar bu vektörün küçük olmasının veya çevirisi yapılacak cümle boyutunun çok büyük olmasının çeviri kalitesini ciddi miktarda düşürdüğünü göstermiştir [13].

2.3. Evrişimli Sinir Ağları

Derin öğrenme yöntemleri içerisinde sayısız kullanım durumu sağlayabilen bir dizi yöntem ve model ortaya çıkmıştır. Bu modellerden birisi de evrişimli sinir ağlarıdır. Evrişimli sinir ağları (CNN) görüntü işleme ve bilgisayarlı görü alanında kullanılmaya başlamış ve bu alanlardaki büyük buluşlarda kullanılan yöntem olmuştur. Son yıllarda evrişimli sinir ağları metin sınıflandırmadan [14] duygu analizine [15] kadar birçok farklı doğal dil işleme probleminde de kullanılmaya başlamıştır.

Evrişimli yapay sinir ağ modelleri genelde evrişim (convolution), ortaklama (pooling) gibi farklı katmanlardan oluşmaktadır. Bu katmanlar aşağıda daha detaylı açıklanacaktır.

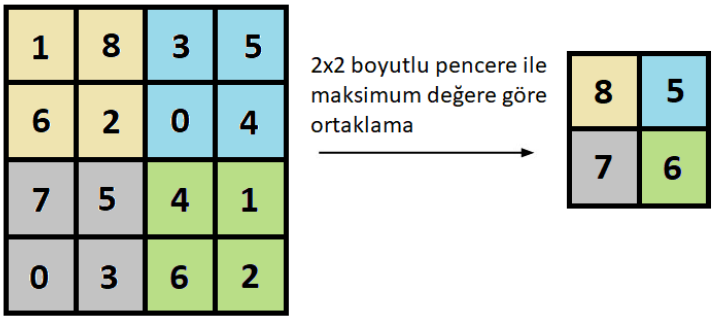


ŞEKIL 2.8. 3x3 boyutundaki filtrenin girdi matrisi üzerine kaydırılarak veriye ait özelliklerin çıkarılması

Evrişim katmanı; tanımlanan bir filtreyi verinin üzerinde baştan sona kaydırarak verinin boyutunu düşürerek verideki özellikleri çıkarma işlemidir. Örneğin Şekil 2.8.'de oluşturulmuş olan 3x3 boyutundaki bir filtre iki boyutlu bir matris üzerinde baştan sona kaydırılmaktadır. Her kaydırma işleminde Formül (9)'deki gibi filtrenin öğeleri (w_{ii}) üzerine denk geldikleri matrisin değerleri (x_{ii}) ile çarpılıp toplanarak tek bir değer üretilmektedir. Bu sebeple işlemin sonunda daha küçük boyutta bir çıktı matrisi elde edilmektedir.

$$s = x_{11}w_{11} + ... + x_{33}w_{33} + b \tag{9}$$

Evrişim katmanında kullanılan filtredeki değerlere göre çıktı olarak üretilen matris değişecek dolayısıyla farklı değerler için veriye ait farklı özellikler öğrenilebilecektir. Bu filtrenin boyutu da aslında bir üst parametredir ve farklı boyutlarda olması modelin öğrenme becerisini etkilemektedir. Ayrıca örnek verilen filtre ve girdi verisi iki boyutlu olsa da bu evrişim işlemi tek boyut ya da 3 boyut gibi farklı boyutlarda da yapılabilmektedir. Filtre ile ilgili bir diğer üst parametre ise adım (stride) aralığıdır. Bu değer, her hesaplamadan sonra filtreyi kaydırırken filtrenin kaç birim kaydırılması gerektiğini belirtmektedir.



ŞEKIL 2.9. 2x2 boyutundaki pencere ile maksimum değere göre ortaklama

Ortaklama (pooling) katmanı; evrişim katmanının çıktısını kullanan bir örnekleme işlemidir. Bu katmanda da yine bir pencere uygulanarak evrişim katmanı sonucunda üretilen matrisin boyutu düşürülmektedir. Ortaklama yaparken tercih edilen ortaklama yöntemine göre bakılan penceredeki değerlerin maksimumunu ya da ortalamasını almak gibi bir yöntem tercih edilebilir. Örneğin Şekil 2.9.'da maksimum ortaklama yöntemi ile bakılan penceredeki en büyük değerlerin alındığı daha küçük boyutlu yeni bir matris elde edilmiştir.

Evrişim ve ortaklama katmanından geçen veriler tam bağlantılı (fully connected) sinir ağı veya tekrarlayan sinir ağı gibi herhangi başka bir ağ modeline girdi olacaksa öncesinde verinin düzleştirilmesi (flattening) gerekecektir.

Evrişimli sinir ağlarının doğal dil işleme problemlerinde kullanılmasıyla dizi etiketleme problemlerinde de kelime özelliklerini çıkarma maksadıyla CNN tercih edilmeye başlamıştır [16–18]. Kelimeler evrişimli sinir ağına verilip buradan kelimelere ait özellikler çıkarılmakta ve bu çıkarılan bu özellikler de kelimenin karakter vektörü olarak kullanılmaktadır.

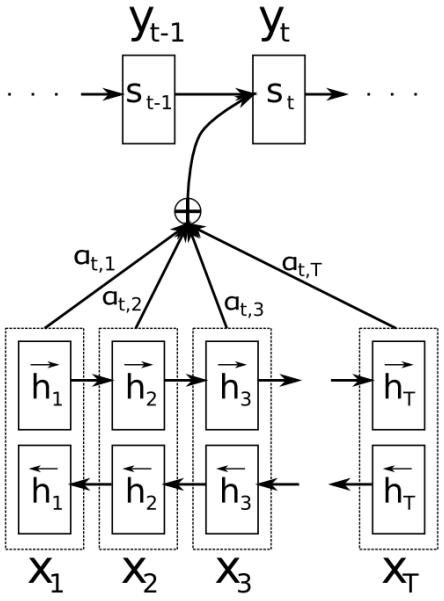
2.4. Dikkat Mekanizması

Son yıllarda Derin sinir ağlarında yaygın kullanılmaya başlayan dikkat mekanizması ilk olarak Bahdanau'nun 2014 yılındaki makine çevirisi üzerine yaptığı çalışmada kullanılmıştır [19]. Bu çalışmada mevcut makine çevirisi çalışmalarının kısıtlarını aşmak ve mevcut sonuçlardan daha iyi bir sonuç elde etmek için daha sonraları dikkat mekanizması ya da dikkat ağları olarak adlandırılacak bir çözüm ortaya atılmıştır.

2.4.1. Tanım

Makine çevirisi problemleri daha önce de bahsedildiği üzere genelde diziden diziye problemi ile çözülmeye çalışılmıştır. Bu modelde cümle uzunluğu ne olursa olsun, cümlenin sabit uzunluktaki bir vektöre sıkıştırılmasının özellikle uzun cümleler için çeviri kalitesini düşürmesi araştırmacıları farklı çözüm yolları bulmaya yönlendirmiştir. Dikkat mekanizmasının ilk olarak makine çevirisi konusunda yapılan çalışmalarda geliştirilmiş olmasından ötürü dikkat mekanizması da diziden diziye modelinin üzerine kurgulanmıştır.

Bahdanau ve diğ. (2014) [19] yapmış oldukları çalışmada cümleyi tek bir vektöre sıkıştırıp çözümleme aşamasında o vektörü kullanmak yerine çözümlemenin her adımında yeniden çevirisi yapılacak cümleye dönerek cümlenin farklı kelimelerine dikkat etmesini sağlamışlardır. Bahdanau'nun çalışmasından alınan model çizimine Şekil 2.10.'da yer verilmiştir.



ŞEKIL 2.10. Bahdanau'nun çalışmasındaki dikkat mekanizması [19]

Şekil 2.10.'da verilen dikkat mekanizmasında s_t çözümleyici tekrarlayan sinir ağının t adımındaki gizli durumunu, y_t değeri ise o adımdaki çıktıyı ifade etmektedir. Kodlayıcı tekrarlayan sinir ağına ait gizli durumlar ise h_j ile ifade edilmiştir. Gerçekleştirilen işlemler matematiksel olarak aşağıdaki gibi ifade edilir:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \tag{10}$$

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j \tag{11}$$

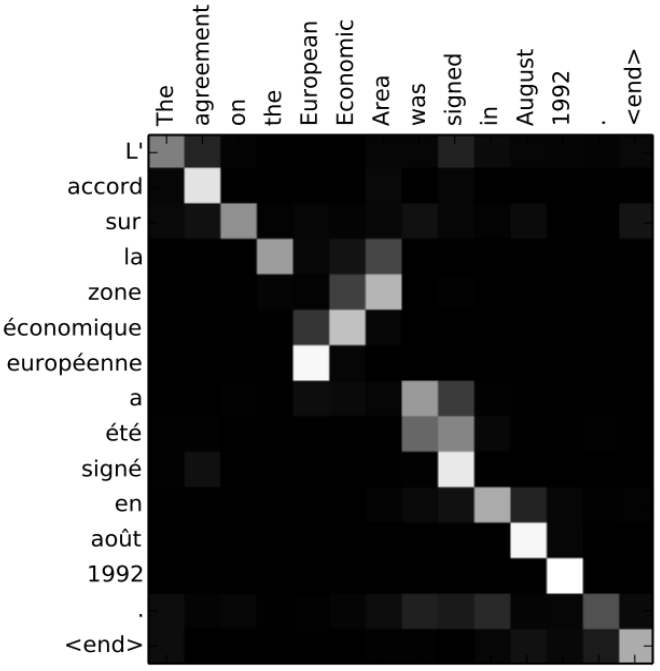
$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})} \tag{12}$$

$$e_{tj} = a(s_{t-1}, h_j) \tag{13}$$

e_{tj} değeri çıktı dizisinin t adımına ait gizli durum (s_{t-1}) ile girdi dizisinin j adımının gizli durumu (h_j) arasındaki eşleşmeyi ifade eder ve eşleşme skoru olarak adlandırılır. Formül (13) ile bulunan eşleşme skoru softmax katmanında geçirilerek α_{tj} ağırlık değeri bulunur. Bulunan bu ağırlık değeri ve kodlayıcı tekrarlayan sinir ağına ait gizli durumlar Formül (11)'deki gibi çarpılarak bağlam vektörü elde edilir. Dikkat mekanizması kullanılmayan diziden diziye modellerde çözümleyici tekrarlayan sinir ağının t adımındaki gizli durumu (s_t); bir önceki adımın gizli durumu (s_{t-1}) ve yine bir önceki adımın çıktısı (y_{t-1}) ile

bulunur. Dikkat mekanizması kullanan modellerde ise artık bu ikisinin yanı sıra dikkat mekanizması ile bulunmuş olan bağlam vektörü (c_t) de hesaba dahil edilir.

Formül (12) ile bulunmuş olan, softmax'den geçirilmiş eşleşme skorları kullanılarak dikkat mekanizması görselleştirilebilmektedir. Şekil 2.11.'de İngilizceden Fransızcaya çevrilmiş bir cümlede her kelimede önem verilen noktalar gösterilmiştir [19].



ŞEKIL 2.11. Bahdanau'nun çalışmasındaki dikkat mekanizması sonucu elde edilen örnek bir İngilizce cümlenin Fransızca karşılığı ve kelimelerin birbiri üzerindeki önem derecesi [19]

Dikkat mekanizmasının kullanılması ve makine çevirisi alanında iyi sonuçlar alınması ile birlikte hem makine çevirisi alanında [20] hem de dizi etiketlemeden [21] görüntüden metin çıkarımına [22] kadar diğer birçok alanda da dikkat ağı çalışmaları yapılmıştır. Bu çalışmalarda global dikkat ağı ve lokal dikkat ağı gibi farklı dikkat mekanizmaları da ortaya çıkmıştır [20]. Global dikkat mekanizmasında, kodlayıcı tekrarlayan sinir ağının tüm adımları hesaba katılarak bağlam vektörü çıkarılırken lokal dikkat mekanizmasında ise sadece belirli bir komşuluk uzaklığında olan düğümler bağlam vektörünün hesaplanmasında kullanılmıştır.

2.4.2. Basitleştirilmiş Dikkat Ağları

Makine çevirisi problemi dikkat ağları çıkmadan önce de diziden diziye modeliyle çözülmeye çalışıldığı için kısıtları aşmak için ortaya çıkan dikkat mekanizması da diziden diziye modelinin üzerine kurgulanmıştır. Ancak dikkat mekanizmasını kullanmak için bir diziden diziye model olması gibi bir zorunluluk yoktur. Dikkat mekanizmasının mantığı bir değeri bulmaya çalışırken kaynaktaki hangi noktalara daha çok dikkat edilmesine karar vermektir. Bu ihtiyaca yönelik daha basitleştirilmiş dikkat ağları da tasarlanmıştır. Bu tür dikkat ağları sadece bazı çalışmalarda basitleştirilmiş dikkat ağı olarak geçmekte [23], diğer çalışmalarda ise bir ayrım yapılmadan sadece dikkat ağı veya dikkat mekanizması ifadeleri kullanılmaktadır [24].

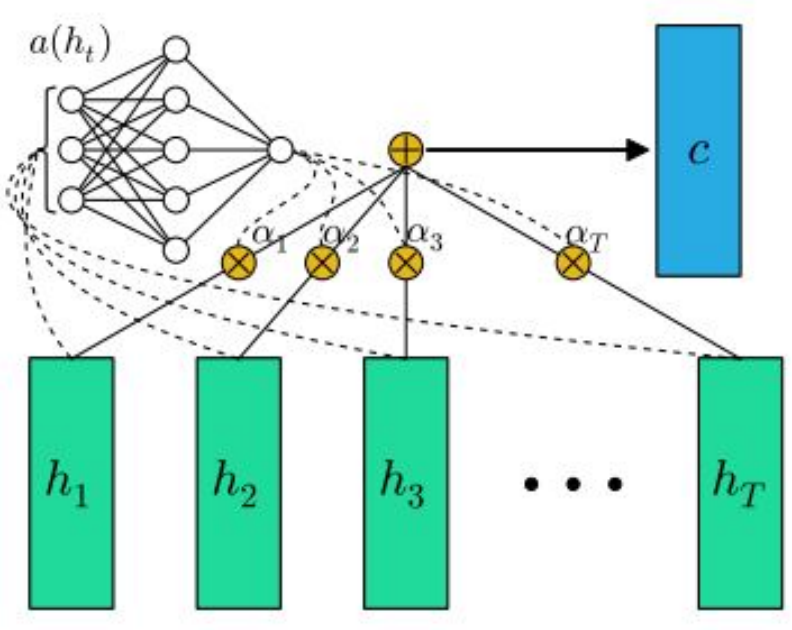
Basitleştirilmiş dikkat ağı yönteminde çok sayıda girdi dikkat mekanizmasına girdi olarak verilir. Ağın içerisinde bulunan öğrenilebilir parametreler ile ağırlıkları belirlenir. Daha sonrasında bunların ağırlıklı toplamları ile tek bir değer elde edilmiş olur. Yani bu modelle bulunmak istenen tek bir çıktıya karar verirken çok sayıda girdiden hangisi veya hangilerine daha fazla önem verilmesi, dikkat edilmesi bulunmaya çalışılır.

$$c = \sum_{t=1}^{T} \alpha_t h_t \tag{14}$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \tag{15}$$

$$e_t = a(h_t) \tag{16}$$

Basitleştirilmiş dikkat mekanizmasına ait formüller görüldüğü üzere daha önce verilen temel dikkat mekanizması formüllerinin (11)-(13) biraz yalınlaştırılmış halidir. Bunun sebebi temel dikkat mekanizmasında her çıktı adımı için tüm girdilere bakılırken burada ise tek bir sefer tüm girdilere (h_t) bakılıp bunların eşleşme skorları (e_t, α_t) ile çarpılıp toplanmasından tek bir bağlam değeri c elde edilmektedir. Bu modele ait örnek bir çizime Şekil 2.12.'de yer verilmiştir.



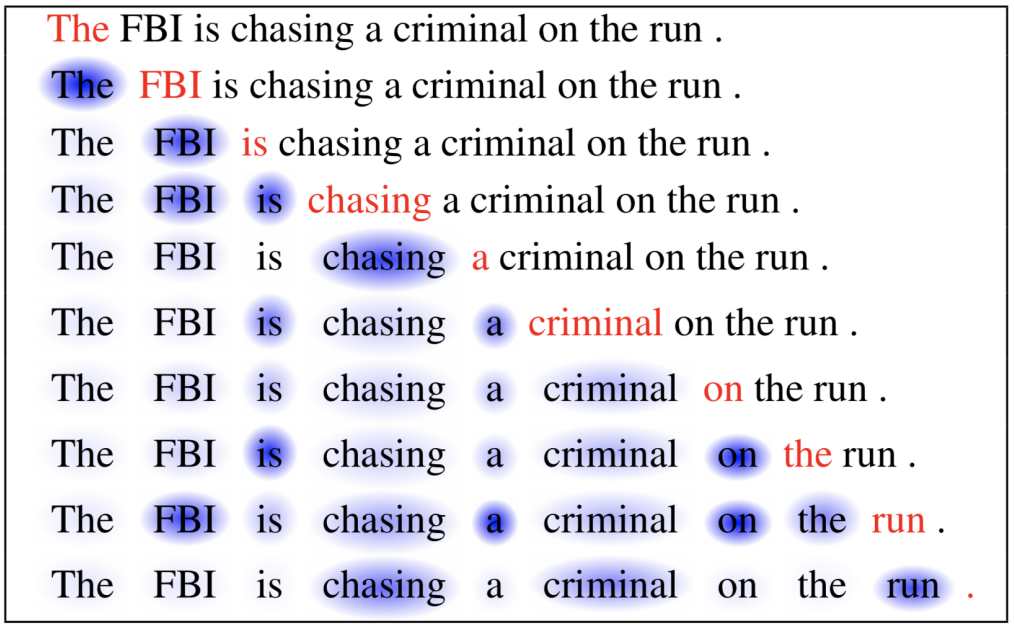
ŞEKIL 2.12. Basitleştirilmiş dikkat mekanizması örnek çizimi [23]

2.4.3. Öz Dikkat Modeli

Bir dizinin bir gösterimini hesaplamak için aynı dizinin farklı konumlardaki öğelerini ilişkilendiren dikkat mekanizmasına öz dikkat (self attention) veya iç dikkat (intra attention) modeli denilmektedir. Yani öz dikkat modeli (mekanizması) bize girdideki bir öğenin diğer öğelerle olan ilişkisini vermektedir. Doğal dil işleme problemleri özelinde öz dikkat mekanizmasını açıklama gerekirse bir cümledeki bir kelimenin yine aynı cümledeki diğer kelimelerle olan ilişkilerinin bulunabilmesini sağlayan bir yöntem olarak düşünülebilir.

Öz dikkat mekanizmasını yalın dikkat mekanizmasından ayıran en temel fark bir değerin hesaplanması sırasında kaynaktaki farklı bir diziye bakmak yerine yine aynı diziye bakarak o diziden hangi öğelerin mevcut hesaplanacak olan öğeyle daha fazla ilişkili olduğunun bulunmasıdır.

Öz dikkat mekanizması doğrudan bu isimle geçmese de ilk olarak Cheng ve diğ. (2016) [25]'un makine okuması üzerine yaptıkları çalışmada kullanılmıştır. Bu çalışmada mevcut kelimenin yine aynı cümledeki önceki kelimelerle olan ilişkisinin öğrenilmesine imkan sağlayacak bir dikkat ağı tasarlanmıştır. Bu çalışmada yer verilen örnek bir metnin ve kelimelerin önceki kelimelerle olan ilişkisi Şekil 2.13.'de gösterilmiştir.



ŞEKIL 2.13. Cheng [25]'in makine okuması çalışmasında mevcut kelimelerin önceki kelimelerle olan ilişki gösterimi

Dikkat ağı (self attention) ifadesinin ilk geçtiği çalışma Vaswani ve diğ. (2017) [26]'ne ait çalışmadır. Bu çalışmada makine çevirisi problemini çözmek için Transformer ağı adı verilen ve hiçbir tekrarlayan sinir ağı içermeyen bir kodlayıcı-çözümleyici modeli oluşturulmuş ve bu model tamamen öz dikkat mekanizması üzerine kurgulanmıştır. Tekrarlayan sinir ağlarında bir adımdaki çıktıyı hesaplamak için kendisinden önceki adımların da hesaplanması gerekmekteyken bu modelde bir tekrarlayan sinir ağı yer almadığı için işlemler paralel bir şekilde gerçekleştirilebilmektedir.

Öz dikkat mekanizması çıktılarını hesaplamak için öncelikle dizideki her değer için anahtar (key), değer (value) ve sorgu (query) vektörlerinin bulunması gerekmektedir. Bu vektörler dizideki her değerin yine isimleri anahtar (W^K), değer (W^V) ve sorgu (W^Q) olan ağırlık parametrelerinin çarpılması ile bulunacaktır. Bu vektörler bulunduktan sonra öz dikkat ağının çıktıları aşağıdaki gibi hesaplanmaktadır.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{17}$$

Bu formülde Q, K, V değerleri dizideki tüm öğelerin sorgu, anahtar ve değer vektörlerini tutan matrislerdir. d_k değeri ise anahtar vektörünün boyutunu ifade etmektedir. Bu bölme işlemi daha kararlı gradyanlara sahip olunmasını sağlamaktadır.

Normalde öz dikkat mekanizmasında dikkat çıktıları oluşturulurken girdi olan dizideki tüm değerler hesaba dahil edilmektedir. Ancak bazı durumlarda tüm değerler yerine sadece mevcut öğeden önceki değerlerin ya da sadece belirli bir komşuluktaki değerlerin hesaba katılması istenmektedir. Bu tarz durumlarda Formül (17)'deki $softmax$ işleminden önce bir de maskeleme işlemi yapılarak sadece ilgili değerlerin hesaba dahil edilmesi sağlanmaktadır.

2.5. Koşullu Rastgele Alan

Dizi etiketleme problemleri gibi birçok uygulamada, birbirine bağımlı birden fazla değişkenin yani girdi olarak verilen özellikler ile rastgele değişken vektörlerinin tahmin edilmesi istenmektedir. Koşullu ratgele alan (CRF) yöntemi bu tarz problemler için yaygın olarak kullanılmaktadır [27].

Koşullu rastgele alan (CRF), bir girdi dizisine karşılık gelecek en muhtemel etiket dizisini tahmin etmek için kullanılan bir modeldir. Sıradan bir sınıflandırıcı komşu değerlere bakmadan tek bir değere bakarak bir etiketi tahmin etmekteyken CRF ise komşu değerlere de bakmakta yani bağlamı da hesaba katmaktadır.

2.5.1. Tanım

Koşullu Rastgele Alan, diziler ve ağaçlar gibi yapılandırılmış verilerin etiketlenmesi ve bölümlendirilmesi için kullanılan, sınıflandırma ve grafiksel modellemenin avantajlarını birleştiren olasılıksal bir sınıflandırma yöntemidir.

Bayes ağları, yapay sinir ağları, Markov modelleri gibi grafiksel modeller birbirine bağımlı çıktı değişkenlerini temsil etmek için kullanılabilmektedir. Bu grafiksel modeller girdiler ve çıktılar üzerinde birleşik olasılık dağılımını modellemeye (joint probability distribution) çalışmaktadırlar. Bu yöntemlerin bazı avantajları olsa da değişkenlerin arasında karmaşık bağımlılıkların olması bu değişkenler üzerinde bir olasılık dağılımı oluşturmayı zorlaştırmaktadır. Bu kısıtlar koşullu rastgele alan kullanılarak aşılabilmektedir.

Bu yöntemde temel fikir, verilen bir girdi dizisine ait etiket dizileri üzerinde ortak bir dağılım yerine koşullu olasılık dağılımını (conditional probability distribution) tanımlamaktır.

CRF; cümlenin kelimeleri gibi bir girdi dizisini ($x = (x_1, \ldots, x_m)$) alarak o kelimelere ait varlık ismi etiketleri gibi bir çıktı dizisini ($s = (s_1, \ldots, s_m)$) üretmektedir. Verilen bir girdi dizisine ait çıktı dizisinin koşullu olasılığı $p(s_1, \ldots, s_m | x_1, \ldots, x_m)$; tüm girdi dizisini çıktı dizisine eşleyen d boyutlu bir özellik vektörü $\Phi(x_1, \ldots, x_m, s_1, \ldots, s_m) \in \mathbb{R}^d$ ile bulunmaktadır. Girdi dizisinden çıktı dizisinin hesaplanması için kullanılacak koşullu olasılık Formül (18) ile modellenmektedir:

$$p(s|x; w) = \frac{\exp(w \cdot \Phi(x, s))}{\sum_{s'} \exp(w \cdot \Phi(x, s'))} \tag{18}$$

3. LİTERATÜR İNCELEMESİ

Doğal dil işleme alanındaki çalışmalarda son yıllara kadar istatistiksel yöntemler daha yaygın kullanılıyordu. Bu istatistiksel yöntemler manuel olarak çıkarılmış özelliklerin kullanılmasını ve çalışılan problemin türüne göre probleme özgü bir takım kaynakların kullanılmasını gerektiriyordu. Örneğin varlık ismi tanıma problemi üzerinde çalışılıyorsa özel isim sözlüğü, coğrafi isim sözlüğü gibi kaynaklar kullanılabiliyor ya da kelimelerin büyük küçük harf içerip içermemesi, rakam içerip içermemesi gibi manuel çıkarılabilecek özelliklere ihtiyaç duyulabiliyordu. Bu sebeple derin öğrenme yöntemleri kullanılmaya başlayana kadar dizi etiketleme problemlerini bir bütün olarak ele alan çalışmalardan ziyade sadece belirli bir problemi çözmeye odaklanmış olan çalışmalar daha yaygındı. Derin öğrenme yöntemlerinin dizi etiketleme alanında kullanılmaya başlaması ile ek kaynaklar kullanmayan ve birden fazla dizi etiketleme problemini aynı anda çözmeye çalışan çalışmalar yaygınlaştı. Ancak şu an Türkçede yapılmış çalışmalar arasında dizi etiketleme problemlerini bir bütün olarak çözmeye çalışan fazla çalışma bulunmamaktadır.

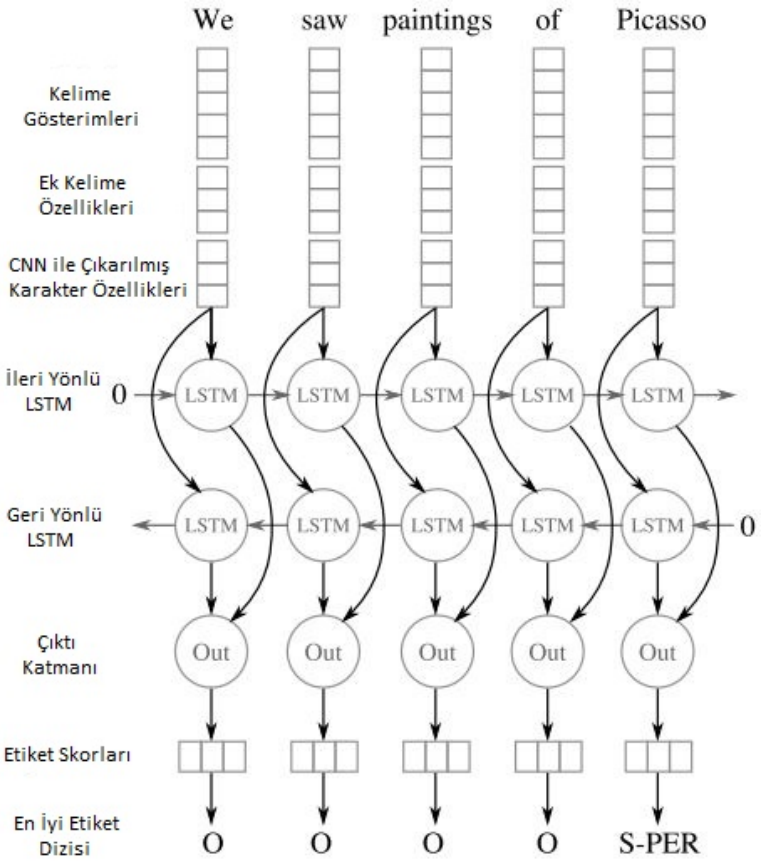
Bu bölümde dizi etiketleme problemleri üzerinde daha önce yapılmış çalışmalara değinilmiştir. Bu amaçla hem belli bir dizi etiketleme problemi üzerine yapılmış çalışmalar hem de birden fazla dizi etiketleme problemi için ortak çözüm içeren çalışmalar incelenmiştir. Öncelikle Türkçeden bağımsız olarak dizi etiketleme problemleri ile ilgili son yıllarda yapılmış güncel çalışmalar ve geliştirilen modeller anlatılacak, sonrasında Türkçe için yapılmış çalışmalara değinilecektir. Ayrıca bizim önerdiğimiz modelde önemli bir yer tutan dikkat mekanizması ile ilgili çalışmalardan da ayrıca bahsedilecektir.

3.1. Dizi Etiketleme Alanındaki Güncel Yöntemler

Derin öğrenme yöntemlerinin kullanılmaya başlamasından önce dizi etiketleme problemlerinde Saklı Markov Modeli (Hidden Markov Model - HMM), Maksimum Entropi Markov Modeli (MEMM), Koşullu Rastgele Alan (CRF) gibi istatistiksel yöntemler yaygın olarak kullanılmıştır [2–4].

İstatistiksel yöntemlerden sonra yapay sinir ağlarının kullanıldığı kimi çalışmalar yapılmıştır. Bu çalışmalardan birisinde **Collobert (2011)** kelime vektörleri kullanan basit ama etkili bir yapay sinir ağı önermiştir [5]. Bu modeldeki bazı özellikler manuel olarak çıkarılıyorsa da çoğu özellik kelime vektörleri ile öğrenilecek şekilde tasarlanmıştır.

Son zamanlarda, tekrarlayan sinir ağları (RNN) ile uzun kısa süreli bellek ağı (LSTM), kapılı tekrarlayan hücre (GRU) kullanan mimariler dizisel verileri modellemede büyük başarı göstermiştir. Doğal dil işleme problemlerinde cümlelerdeki kelimeler doğal olarak sadece kendisinden önceki kelimelere değil kendisinden sonra gelen kelimelerle de bağımlı olduklarından iki yönlü tekrarlayan sinir ağı ve türevlerini kullanmak daha anlamlı sonuçlar elde edilmesini sağlamıştır.



ŞEKIL 3.1. Çift yönlü LSTM + CNN Modeli [16]

Yine bazı doğal dil işleme problemlerinde karakter seviyesi özellikleri çıkarmak için evrişimli yapay sinir ağları (CNN) kullanılmıştır. Bu yaklaşım dizi etiketleme problemlerinde

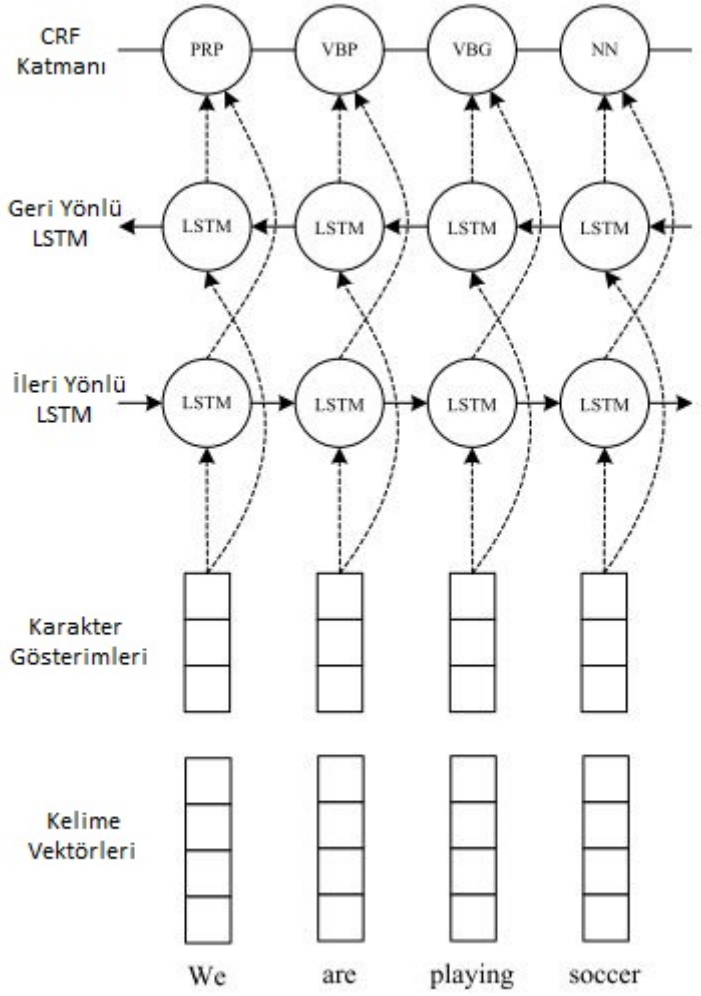
de kullanılmış ve başarılı sonuçlar alınmıştır [28, 29]. Bazı çalışmalar ise karakter seviyesindeki bu özellikleri evrişimli sinir ağı yerine tekrarlayan sinir ağı veya türevleri ile çıkarmıştır [30–32].

Chiu & Nichols (2016) [16] varlık ismi tanıma problemine yeni bir çözüm getirmek için çift yönlü uzun kısa süreli bellek ağını (LSTM) kullanan bir model tasarlamıştır. Bu çift yönlü LSTM ağı, kelime vektörü ve evrişimli sinir ağıyla elde edilmiş karakter özelliklerini içeren karakter vektörünü girdi olarak almaktadır. Bunlara ek olarak manuel olarak çıkarılmış büyük-küçük harf ve rakam içerip içermeme gibi bazı ek özellikler de girdilere eklenmiştir. LSTM çıktısı $softmax$ katmanından geçirilerek istenen etiketler elde edilmiştir. Bu modele ait çizime Şekil 3.1.'de yer verilmiştir.

Huang ve diğ. (2015) [33] de çalışmalarında benzer bir model kullanmıştır. Yine modelin merkezinde çift yönlü uzun kısa süreli bellek ağı bulunmaktadır. Girdi olarak sadece kelime vektörü kullanılmış, karakter özelliklerini içeren herhangi bir karakter vektörü kullanılmamıştır. Modelin önceki modelden bir diğer farkı da LSTM çıktısının koşullu rastgele alandan (CRF) geçirilmesi olarak gözükmektedir. Böylece etiketler arasındaki ilişki CRF yardımıyla modellenmektedir. Ayrıca bu çalışmada sadece varlık ismi tanıma problemi değil genel olarak dizi etiketleme problemleri üzerinde çalışılmıştır.

Ma & Hovy (2016) [17] ise dizi etiketleme problemleri için oluşturdukları modelde Chiu & Nichols [16] çalışması ile benzer şekilde girdi olarak hem kelime vektörlerini hem de CNN ile elde edilmiş karakter vektörlerini kullanmıştır. Çift yönlü uzun kısa süreli bellek ağının çıktısını ise Huang ve diğ. [33] çalışması ile benzer şekilde koşullu rastgele alandan geçirmiştir. Bu model önceki modellerden farklı olarak manuel olarak çıkarılmış hiç bir özellik veya probleme özgü hiç bir kaynak kullanmamıştır. Bu çalışmanın modeline ait çizim Şekil 3.2.'de verilmiştir.

Yang ve diğ. (2016) [30] yapmış oldukları dizi etiketleme çalışmasında yine merkezinde çift yönlü tekrarlayan sinir ağı olan, girdi olarak kelime vektörlerinin dışında karakter vektörleri de alan, çıktısı CRF katmanına verilen modeli ufak değişiklikler yaparak kullanmıştır. Bu çalışmadaki en temel fark girdileri uzun kısa süreli bellek ağı yerine tekrarlayan sinir ağının bir başka türevi olan kapılı tekrarlayan hücreden (GRU) geçirmiş olmalarıdır.



ŞEKIL 3.2. Çift yönlü LSTM + CNN + CRF Modeli [17]

Benzer şekilde karakter vektörleri de evrişimli sinir ağı yerine kapılı tekrarlayan hücreden geçirilerek oluşturulmuştur.

Lample ve diğ. (2016) [34] sadece varlık ismi tanıma problemi üzerine yapmış oldukları bu çalışmada ortaya koydukları modelin Ma & Hovy (2016) [17]'in çalışmasından tek farkı karakter vektörlerini evrişimli sinir ağı yerine uzun kısa süreli bellek modeli kullanarak üretmeleri olmuştur.

Zhang ve diğ. (2018) [35] önceki çalışmalardan biraz daha farklı bir model ortaya koyarak ve ne girdi olarak karakter vektörlerini kullanmış ne de çıktıyı koşullu rastgele alandan geçirmiştir. Bunun yerine uzun kısa süreli bellek ağının çıktılarını etiket katmanı dedikleri ikinci bir katmandaki uzun kısa süreli bellek ağından geçirerek etiketleri bulmaya

çalışmıştır.

Yang ve diğ. (2018) [18] dizi etiketleme alanında yapılmış tüm bu bahsettiğimiz çalışmaları yeniden kodlayarak karşılaştırmalarını yapmıştır. Bu bağlamda yaptıkları analizler kelime seviyesinde uzun kısa süreli bellek (LSTM) ağı kullanılmasının, karakter vektörünün evrişimli sinir ağı ile oluşturulup kelime vektörüne eklenmesinin ve uzun kısa süreli bellek ağının çıktısının koşullu rastgele alandan geçirilmesinin en iyi sonuçları verdiğini göstermiştir.

Varlık ismi tanıma problemi bu bölümde şu ana kadar anlatılan tüm yayınlarda ortak çalışılan problemdir. Bu sebeple tüm bu çalışmaların sonuçlarını kıyaslamak için bu çalışmalardaki varlık ismi tanıma problemine ait F1 skorları Tablo 3.1.'de verilmiştir. Tüm bu çalışmalarda ortak kullanılan veri kümesi CoNLL-2003 veri kümesidir. Ayrıca çalışmalardaki ek kaynak kullanımı ile elde edilen sonuçlara yer verilmemiştir.

TABLO 3.1. Varlık ismi tanıma problemi üzerine yapılmış literatürdeki güncel çalışmaların CoNLL-2003 veri kümesi üzerindeki sonuçlarının karşılaştırılması

Model	F1
Chiu & Nichols (2016)	90,91
Huang ve diğ. (2015)	90,10
Ma & Hovy (2016)	91,21
Yang ve diğ. (2016)	90,96
Lample ve diğ. (2016)	90,94
Zhang ve diğ. (2018)	91,22

3.2. Dikkat Mekanizması Kullanan Çalışmalar

Daha önce Alan Bilgisi bölümünde bahsedildiği üzere dikkat mekanizması ilk olarak 2014 yılında Bahdanau [19]'nun makine çevirisi alanında yaptığı bir çalışmada yer bulmuştur. Daha sonrasında ufak değişikliklerle birçok farklı dikkat ağı modeli oluşturulmuş ve bu modeller farklı alanlarda da kullanılmaya başlanmıştır. Bu bölümde bu dikkat ağı modellerinden, modellerin farklılıklarından ve bu modelleri kullanan çalışmalardan bahsedilecektir.

Bahdanau ve diğ. (2014) [19] çalışmasında makine çevirisi yaparken cümlenin uzunluğu ne olursa olsun bunu sabit uzunlukta bir vektöre kodlayıp sıkıştırmanın özellikle

uzun cümlelerde çeviri başarısını ciddi bir şekilde düşürdüğünü, bunun yerine çözümleme yapılırken her adımda çevirisi yapılacak cümleye dönülerek nereye daha fazla dikkat edilmesi, önem gösterilmesi gerektiğine bakacak bir çözümün sonuçları önemli ölçüde etkilediğini ortaya koymuştur. Makine çevirisine ekledikleri dikkat mekanizmasının sonuçları nasıl etkilediğini görmek için dikkat mekanizması içeren ve içermeyen iki farklı model oluşturulmuş ve aynı veri kümesi üzerinde BLEU skorları üzerinden karşılaştırma yapılmıştır. 30 kelimeye kadar olan cümleler için dikkat mekanizması eklenmesi BLEU skorunu 13,93'ten 21,50'ye çıkarırken 50 kelimeye kadar olan cümlelerdeki BLEU skoru 17,82'den 26,75'e çıkmıştır.

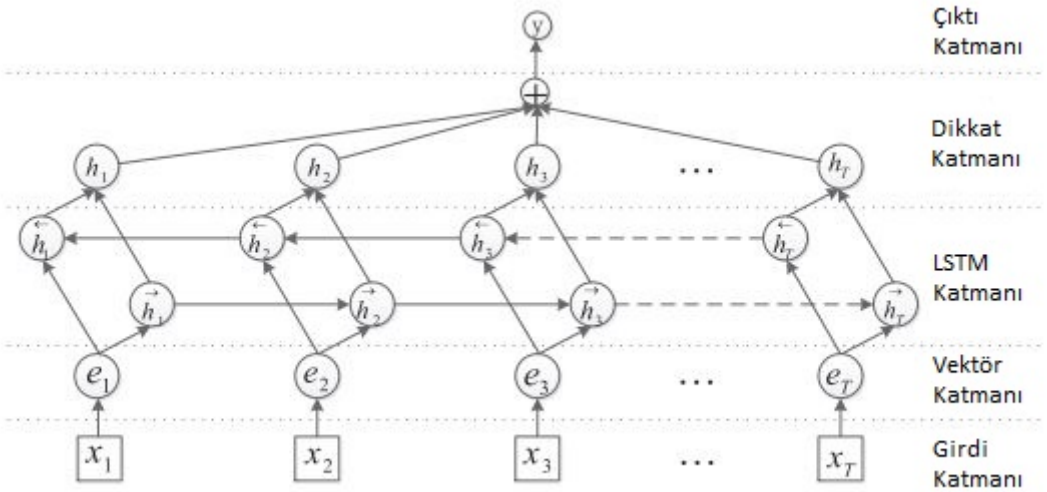
Luong ve diğ. (2015) [20] çalışmasında global ve lokal olmak üzere iki farklı dikkat mekanizması modeli ortaya koymuştur. Global dikkat mekanizması, çözümleme anında çevirisi yapılacak olan cümlenin tüm kelimelerini hesaplamaya dahil ederken, lokal dikkat mekanizması ise sadece belirli bir komşuluktaki kelimeleri hesaplamaya dahil etmektedir. Önerilen global dikkat mekanizması aslında Bahdanau ve diğ. (2014) [19] çalışmasındaki dikkat mekanizmasına çok benzemekle birlikte birkaç ufak farklılık içermektedir:

- Bahdanau'nun çalışmasında iki yönlü uzun kısa süreli bellek ağı (LSTM) kullanılırken bu çalışmada tek yönlü uzun kısa süreli bellek ağı kullanılmıştır.
- Luong'un çalışmasında daha basit bir hesaplama yolu kullanılmıştır. Bahdanau bir t anındaki çıktıyı hesaplamak için bir önceki adımın gizli durumunu (h_{t-1}) dikkat mekanizmasından geçirip (a_t) bulunan bağlam değerini (c_t) çözümleme işleminin t adımına tekrardan girdi olarak eklerken ($h_{t-1} \rightarrow a_t \rightarrow c_t \rightarrow h_t$); Luong ise çözümleme işleminin t adımındaki gizli durumu (h_t) ürettikten sonra bunu dikkat mekanizmasından geçirerek doğrudan sonucu üretmektedir ($h_t \rightarrow a_t \rightarrow c_t \rightarrow \widehat{h}_t$).

Li ve diğ. (2018) [21] dikkat mekanizmasını önceki çalışmalardan farklı olarak dizi etiketleme problemi için uygulamıştır. Oluşturulan modelde Luong ve diğ. (2015) [20] çalışmasındakine benzer bir şekilde hesaplama işlemi için ($h_t \rightarrow a_t \rightarrow c_t \rightarrow \widehat{h}_t$) yolu kullanılmıştır. Ancak $\widehat{h}_t$ değerini hesaplarken; Luong'un çalışmasında, dikkat mekanizması ile elde edilen bağlam vektörü (c_t) çözümleme işleminin t adımının çıktısı ile birleştirilerek elde ediliyorken, bu çalışmada ise bağlam vektörü kodlama işleminin t adımının çıktısı ile

birleştirilerek elde edilmiştir. Yazarlar, dizi etiketleme probleminde çıktının büyük ölçüde kodlama adımının çıktısı ile ilişkili olmasından ötürü böyle bir yol izlediklerini belirtmişlerdir.

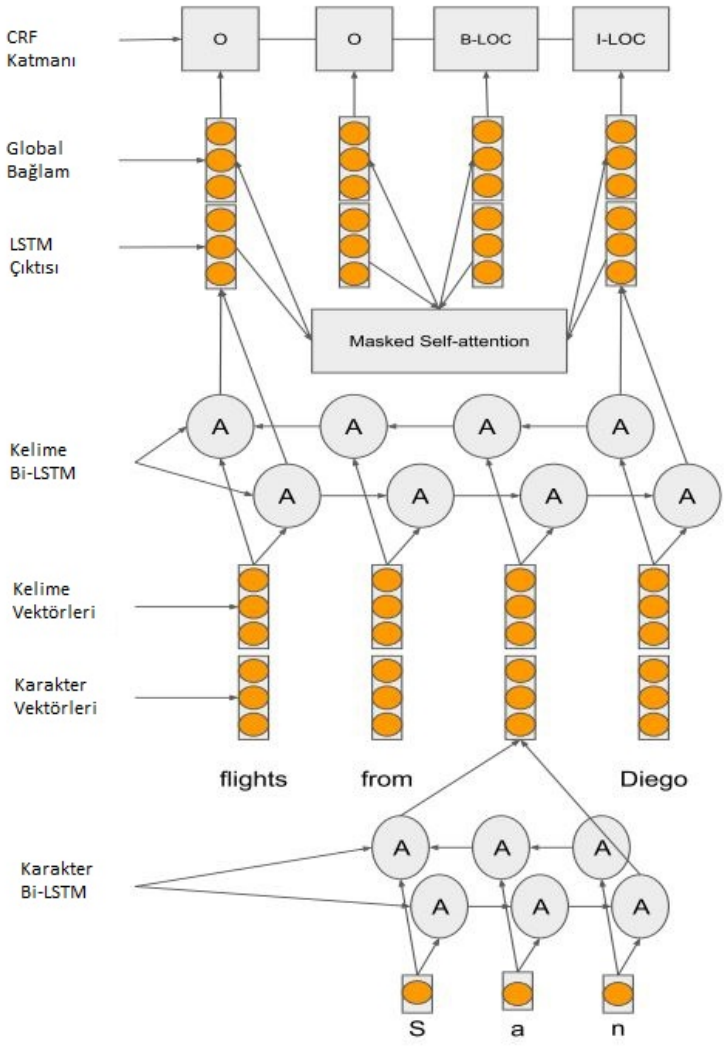
Raffel & Ellis (2015) [23] önceki çalışmalardan farklı olarak basitleştirilmiş dikkat mekanizması kullanmışlardır. Hatta tasarladıkları modelde bir tekrarlayan sinir ağı veya türevi bir ağ dahi bulunmamaktadır. İleriye doğru sinir ağını basitleştirilmiş dikkat mekanizması ile birlikte kullanmışlardır. Yine benzer bir çalışmada **Zhou ve diğ. (2016)** [24] de basitleştirilmiş dikkat mekanizmasını ilişki sınıflandırma problemi için kullanmışlardır. Burada yapılan işlem tekrarlayan sinir ağının çıktılarını basitleştirilmiş dikkat ağından geçirerek ilişki sınıfının bulunmasında hangi kelimelerin daha anlamlı olduğunu bulmaktır. Bu çalışmanın modeline ait çizime Şekil 3.3.'de yer verilmiştir.



ŞEKIL 3.3. Basitleştirilmiş Dikkat Mekanizması kullanan bir ilişki sınıflandırma modeli [24]

Yang ve diğ. (2016) [36] basitleştirilmiş dikkat mekanizmasını hiyerarşik olarak kullanmışlar ve buna da hiyerarşik dikkat ağı adını vermişlerdir. Doküman sınıflandırma problemi için geliştirilmiş olan bu modelde öncelikle kelime vektörleri çift yönlü uzun kısa süreli bellek ağına geçirilip çıktısı da kelime düzeyindeki basitleştirilmiş dikkat ağına verilmiştir. Böylece cümle içerisindeki hangi kelimeye daha çok dikkat edilmesi gerektiği öğrenilmiştir. Cümlelere ait bu dikkat ağı ile oluşturulan bağlam vektörleri tıpkı kelimelerde olduğu gibi önce çift yönlü uzun kısa süreli bellek ağından sonra bunun çıktısı da

cümle seviyesindeki basitleştirilmiş dikkat ağıdan geçirilmiştir. Bu sayede dökümanın sınıflandırılmasında hangi cümlelerin daha anlamlı olduğuna dikkat edilmesi sağlanmıştır.



ŞEKIL 3.4. Öz dikkat ağı kullanan dizi etiketleme modeli [31]

Dizi etiketleme alanında yapılan ve dikkat mekanizması kullanan güncel çalışmalardan birisi de **Xie ve diğ. (2018)** [31]'nin yapmış olduğu çalışmadır. Bu çalışmanın en önemli farkı öz dikkat (self attention) modelini kullanıyor olmasıdır. Çift yönlü uzun kısa süreli bellek ağı ile çıkarılmış karakter vektörleri kelime vektörleri ile birleştirildikten sonra yine çift yönlü uzun kısa süreli belleğe girdi olarak verilmekte, bu katmanın çıktısı da koşullu rastgele alandan geçirilmektedir. Bu haliyle model daha önce 3.1. bölümünde anlatılan güncel dizi etiketleme çözümleri benzerdir. Bu modelin farklılığı LSTM çıktısının CRF'e geçilmeden önce öz dikkat mekanizmasından geçirilmesidir. Öz dikkat mekanizmasının

çıktısı yine LSTM çıktısı ile birleştirilerek CRF katmanına verilmektedir. Bu çalışmanın modeline ait çizime Şekil 3.4.'de yer verilmiştir.

3.3. Türkçede Dizi Etiketleme Çalışmaları

Türkçe dili için yapılan çalışmalara baktığımızda genel olarak dizi etiketleme yerine daha ziyade sadece varlık ismi tanıma problemi üzerine çalışmalar yapıldığı görülmüştür. Ayrıca birkaç çalışma hariç derin öğrenme yöntemlerini kullanan çalışma bulunamamıştır. Çalışmaların çok güncel olmamasından ötürü daha ziyade istatistiksel yöntemlerin kullanıldığı görülmüştür. Burada varlık ismi tanıma ve sözcük türü işaretleme konusunda yapılmış ve bizim de karşılaştırma yaparken kullandığımız çalışmalara kısaca değinilecektir.

Şeker ve Eryiğit (2012) [27] Türkçe dilinin morfolojik özelliklerini koşullu rastgele alan (CRF) ile beraber kullanarak varlık ismi tanıma problemine çözüm getirmişlerdir. Sonuçları iyileştirmek için coğrafi sözlük de kullanılmıştır. Yazıldığı dönem için mevcut çözümleri geçip kullanılan veri kümesi [37] için 91,94'lük F1 skoruna ulaşılmıştır.

Varlık ismi tanıma üzerine yapılan bir başka çalışmada **Güneş ve Tantuğ (2018)** [38] çok katmanlı ve iki yönlü uzun kısa süreli bellek ağını kullanmışlardır. Farklı katman sayıları ile farklı deneyler yapılmış ve genel olarak katman sayısını artırdıkça daha iyi sonuçlar elde edildiğini tespit etmişlerdir. Yazım özellikleri ve biçim bilim özellikleri dahil edildiğinde kullanılan veri kümesinde [37] F1 skorunun 93,69'a kadar çıktığı belirtilmektedir. Yazım ve biçim bilim özellikleri kullanılmadığında ise F1 skorunun 91,59'a gerilediği belirtilmiştir.

Güngör ve diğ.'nin 2018 yılında yapmış oldukları çalışma [32] dizi etiketlemede yaygın kullanılan, kelime ve karakter vektörlerini LSTM katmanına parametre olarak verip bu katmanın çıktısını da CRF katmanından geçirerek sonucu bulmaya çalışan derin ağ modelini Türkçe için kullanan ilk çalışma olarak görülmektedir. Bu çalışmada karakterlerin özellikleri çift yönlü uzun kısa süreli bellek ağı ile çıkarılmaktadır. Daha sonra bu vektörler dağılımsal kelime vektörleri ile birleştirilerek yine kelime seviyesindeki çift yönlü

uzun kısa süreli bellek ağına verilmekte, bunun çıktısı ise koşullu rastgele alandan geçirilmektedir. Bu çalışma herhangi bir ek özellik ve probleme özgü kaynak kullanmamış olmasına rağmen Türkçe için yapılan diğer çalışmalardan daha iyi bir sonuç elde etmiş ve kullanılan veri kümesinde [37] 93,37'lik F1 skoruna ulaşmıştır.

Yine **Güngör ve diğ.**'nin 2019 yılında yayınladıkları başka bir çalışmada [39] ise Türkçenin morfolojik yapısının zenginliği göz önünde bulundurulmuştur. Bundan dolayı kelime ve karakter vektörlerine ilaveten morfem vektörleri de hesaba dahil edilmiştir. Morfem vektörleri çift yönlü uzun kısa süreli bellek ağı kullanılarak çıkarılmıştır. Bu çalışmada 92,93'lük F1 skoruna ulaşılmıştır.

Türkçede sözcük türü işaretleme problemi üzerinde varlık ismi tanıma problemine nazaran daha az çalışılmıştır. Mevcut çalışmalara baktığımızda **Dinçer ve diğ. (2008)** [40] çalışması istatistiksel bir yöntem olan HMM kullanarak kelimelerin sözcük türünü bulmaktadır. METU-Sabancı Türkçe Treebank [1] veri kümesini kullanan bu çalışmada %88,90 doğruluk elde edilmiştir. Yine aynı veri kümesini kullanan bir başka çalışmada ise **Bahçevan ve diğ. (2018)** [41] tekrarlayan sinir ağını ve bunun türevlerinden uzun kısa süreli bellek ağını kullanarak sözcük türü işaretleme gerçekleştirmişlerdir. Bu çalışma da bir öncekinden çok fazla bir iyileşme sağlayamamış ve %89,00'lık bir doğruluk elde etmiştir.

4. ÖNERİLEN MODEL

Dizi etiketleme problemlerinde Bölüm 3.'te de bahsedildiği üzere tekrarlayan sinir ağları (RNN), uzun kısa süreli bellek ağları (LSTM) veya kapılı tekrarlayan hücre ağlarının (GRU) merkezde olduğu ve bu katmanların çıktısının da koşullu rastgele alandan (CRF) geçirildiği derin ağ modelleri son yıllarda daha yaygın olarak tercih edilmektedir. Bu tez kapsamında bu ağlardan uzun kısa süreli bellek ağı tercih edildiğinden bundan sonra bu derin ağ modeli için kısaca LSTM-CRF ağ modeli ifadesi kullanılacaktır.

LSTM-CRF ağ modeli bu çalışmada dayanak noktası olarak alınmış, sonrasında eklenen karakter ve morfem vektörleri ile model daha da geliştirilmiştir. Karakter vektörleri evrişimli sinir ağ modelleri kullanılarak elde edilirken morfem vektörleri ise dikkat mekanizmasının da yer aldığı bir tekrarlayan sinir ağ modeli ile elde edilmiştir. Oluşturulan bu karakter ve morfem vektörleri daha önceden eğitilmiş olan kelime vektörleri ile birleştirilerek çift yönlü LSTM-CRF modelinde eğitilmiştir.

Ayrıca modeldeki çift yönlü uzun kısa süreli bellek ağının (LSTM) çıktıları koşullu rastgele alandan (CRF) geçirilmeden önce öz dikkat ağından geçirilerek daha gelişmiş modeller de oluşturulmuş ve bu modellerin elde ettikleri başarılar karşılaştırılarak en ideal çözüm bulunmaya çalışılmıştır. Oluşturulan modeller dizi etiketleme problemlerinden varlık ismi tanıma ve sözcük türü işaretleme problemleri üzerinde denenmiş ve Türkçe için yapılmış mevcut çalışmalardan daha iyi sonuçlar elde edilmiştir.

Bu bölümde öncelikle model girdileri olan kelime, karakter ve morfem vektörlerinden ve morfem vektörlerinin çıkarılmasında kullanılan dikkat mekanizmasından bahsedilecektir. Daha sonrasında oluşturulan çift yönlü LSTM-CRF ağı açıklanacaktır. Son olarak ise modelin geliştirilmesine ve kodlanmasına ait geliştirme detayları açıklanacaktır.

4.1. Model Girdileri

Kelimeler üzerinde matematiksel işlemler yapabilmek için öncelikle kelimelerin matematiksel olarak ifade edilmeleri gerekmektedir. Bu amaçla kelime vektörleri kullanılmaktadır. Word2Vec [6], GloVe [8] gibi kelime vektörlerinin çıkarımında yaygın kullanılan

yöntemler kelimeleri bir bütün olarak ele aldığından kelimenin karakter ve morfem bilgilerinden yararlanılamamaktadır. Bu amaçla önerilen modelde sadece kelime vektörleri değil aynı zamanda karakter ve morfem seviyesindeki kelime özelliklerini içeren vektörler de modele girdi olarak dahil edilmiştir.

4.1.1. Kelime Vektörleri

İlk olarak Mikolov ve diğ. (2013) [6] tarafından önerilen bu yöntem sayesinde kelimeler sık vektörler olarak tutulmaktadır. Anlamsal olarak yakın kelimelerin vektör uzayında birbirlerine yakın vektörler ile gösterilebildiği bu yöntem CBOW ve Skip-gram adı verilen iki farklı algoritma ile geliştirilebilmektedir.

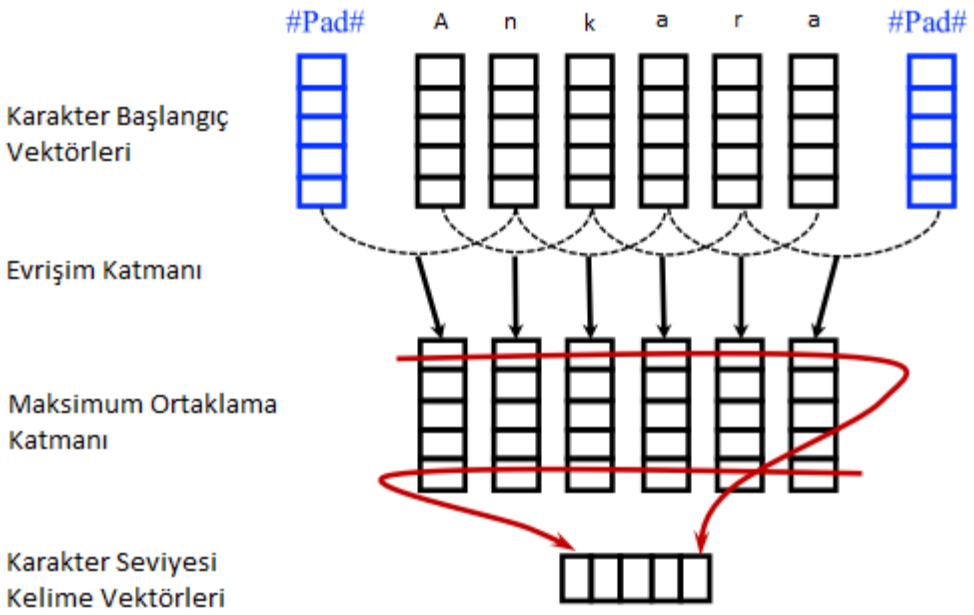
Bu tez kapsamında Skip-gram algoritması ile oluşturulmuş, önceden eğitilmiş bir Word2Vec kelime vektör kümesi kullanılmıştır [42]. Her bir kelimenin 300 boyutundaki vektörlerle ifade edildiği bu vektör kümesinde 1.238.485 kelimeye ait önceden eğitilmiş kelime vektörü bulunmaktadır. Oluşturulacak olan modelde kelime vektörleri E (embeddings) sembolü ile gösterilecektir.

Word2Vec dışında yine yaygın olarak kullanılan bir başka kelime vektörü çıkarım yöntemi olan FastText [9] de LSTM-CRF modeli ile kullanılmıştır. Ancak FastText ile oluşturulmuş vektörlerin daha düşük sonuçlar vermesinden dolayı modelde yapılan daha sonraki geliştirmeler sadece Word2Vec yöntemi ile test edilmiştir.

4.1.2. Karakter Vektörleri

Türkçe gibi sondan eklemeli dillerde kelimenin bir bütün olarak modellenmesi seyreklik problemini de beraberinde getirmektedir. Bundan ötürü, kelimeyi bir bütün olarak ele almaktansa karakterleri üzerinden bir kelimeyi ifade etmek kelime hakkında daha detaylı bilgi vermekte ve seyreklik problemini de azaltmaktadır. Bu amaçla özellikle Türkçe gibi dillerde yapılan doğal dil işleme çalışmalarında önceden eğitilmiş Word2Vec [6] gibi kelime vektörlerinin dışında karakter seviyesindeki kelime özelliklerini içeren ikinci bir

vektör kullanılmaktadır. Bu tez çalışmasında bu vektörlere kısaca karakter vektörleri denilecek ve K sembolü ile gösterilecektir.



ŞEKIL 4.1. Evirişimli sinir ağı kullanarak karakter vektörlerinin çıkarımı [18]

Karakter vektörlerinin oluşturulması için öncelikle karakter başlangıç vektörleri (character embeddings) kullanılmaktadır. Bu vektörler rastgele ve tek düze bir şekilde -0,5 ile +0,5 arasındaki değerlerle başlatılmış 30 boyutundaki vektörlerdir. Bir kelimedeki her bir karakter için ayrı karakter başlangıç vektörleri olmuş olacaktır. Bu vektörler karakter seviyesindeki kelime özelliklerinin çıkarılması için kullanılacak olan evrişimli sinir ağına girdi olacaklardır.

Oluşturulan ve ilk değerleri ile başlatılan karakter başlangıç vektörleri Şekil 4.1.'deki gibi öncelikle evrişim katmanından geçirilmiştir. Bu evrişim katmanında 3 boyutunda bir filtre (pencere) kullanılmış ve atlama adım aralığı 1 olarak seçilmiştir. Bu katmanın çıktısı ise maksimum ortaklama katmanından geçirilerek her kelime için 30 boyutunda tek bir vektör elde edilmiştir. Bu elde edilen vektör, modelin karakter seviyesi kelime vektörü ya da kısaca karakter vektörü olarak adlandırılacaktır.

Oluşturulan 30 boyutundaki bu karakter vektörleri (K) önceden eğitilmiş olan 300 boyutundaki kelime vektörleri (E) ile uç uca eklenerek her kelimenin 330 boyutunda temsil

edildiği yeni bir kelime gösterimi ($E \oplus K$) elde edilmektedir. LSTM-CRF modeline artık girdi olarak bu yeni kelime gösterimi verilecektir.

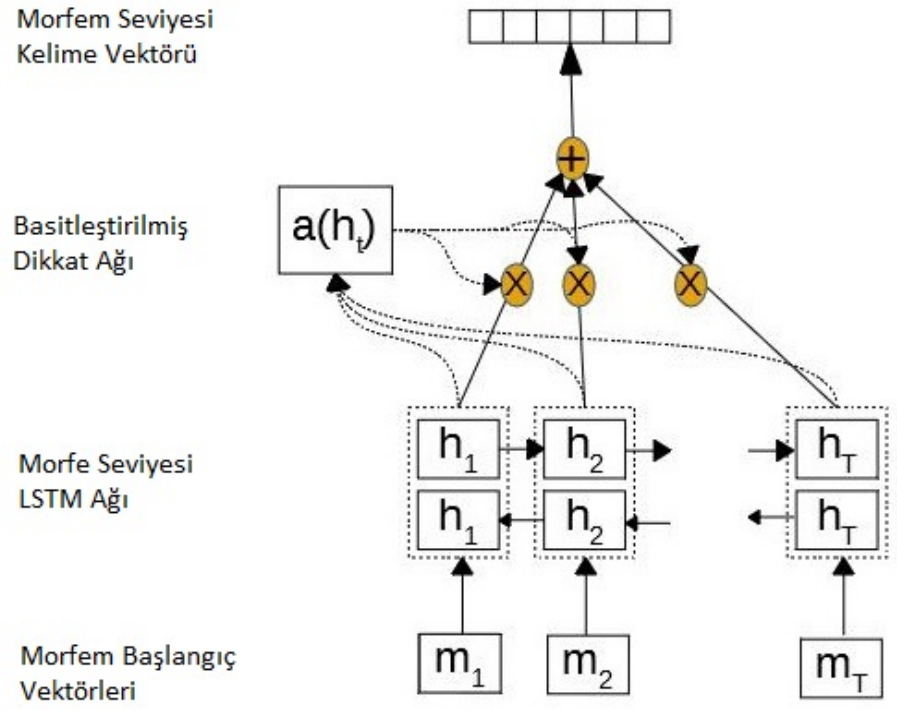
4.1.3. Morfem Vektörleri

Türkçenin sondan eklemeli yapısından ötürü bir kökten onlarca farklı kelime türetilebilmektedir. Türkçe için kelime vektörlerinin oluşturulacağı veri kümesinde her olası kelimenin yer alması zor olduğundan çalışılan veri kümesinde sözlük harici kelimelerle ya da nadir geçen kelimelerle sıkça karşılaşılabilmektedir. Sözlük harici kelimelere rastgele değerlerin atanması ve nadir geçen kelime vektörlerinin iyi eğitilmemiş olması sebebiyle İngilizce gibi dillerde denenip iyi sonuçlar vermiş modeller Türkçe gibi sondan eklemeli diller için beklenenden daha düşük sonuçlar verebilmektedir.

Sondan eklemeli dillerdeki bu seyreklik kaynaklı problemi çözmek için bir önceki kısımda anlatılan karakter vektörleri kullanılabileceği gibi kelimelerin morfem özelliklerinin kullanılması da bir çözüm olarak değerlendirilmektedir. Nitekim yapılan çalışmalar da kelime morfem bilgilerinin oluşturulan modellere dahil edilmesinin Türkçe için sonuçları iyileştirdiğini göstermiştir [43, 44].

Bu tez kapsamında önerilen modelde de morfem seviyesi kelime özelliklerinin kullanılabilmesi için morfem vektörleri oluşturulmuş ve LSTM-CRF modeline dahil edilmiştir. LSTM-CRF modeli, morfem düzeyi ve kelime düzeyi olmak üzere iki katmanlı hiyerarşik bir mimariye çevrilmiştir. Morfem düzeyindeki çift yönlü uzun kısa süreli bellek ağı (LSTM) ve basitleştirilmiş dikkat mekanizması kullanılarak morfem vektörleri adı verilecek olan ve kelimelerin morfem özelliklerini tutan vektörler çıkarılmakta, çıkarılan bu vektörler kelime düzeyindeki çift yönlü LSTM katmanına girdi olarak verilmektedir. Önerilen modelde morfem vektörleri M sembolü ile gösterilecektir. Şekil 4.2.'de modelin morfem düzeyindeki bileşenleri gösterilmiştir.

Morfem vektörlerinin çıkarılması için öncelikle kelimelerin morfem etiketlerinin çıkarılmış ve bu morfem etiketlerinin sıcak kodlanmış(one-hot) vektörlerine çevrilmiş olması gerekmektedir. Bu katmanda kullanılan çift yönlü uzun kısa süreli bellek ağının (LSTM) durum boyutu 30 olarak belirlenmiştir. Yani hem ileri yönlü hem de geri yönlü LSTM



ŞEKIL 4.2. LSTM ve dikkat mekanizması kullanarak morfem vektörlerinin çıkarımı

ağları girdi olarak morfem etiketlerinin sıcak kodlanmış vektörlerini almakta, çıktı olarak ise 30 uzunluğunda vektörler üretmektedirler. Bu ileri ve geri yönlü ağların çıktıları uç uca birleştirilerek 60 boyutunda yeni bir vektör elde edilmektedir. Elde edilen bu vektör kelimenin morfem vektörüdür.

Uzun kısa süreli bellek ağı ile elde edilen 60 uzunluğundaki bu morfem vektörleri (M), karakter vektörlerine (K) benzer şekilde önceden eğitilmiş olan 300 boyutundaki kelime vektörleri (E) ile uç uca eklenerek her kelimeyi 360 boyutunda temsil eden yeni bir kelime gösterimi ($E \oplus M$) elde edilmektedir. Ayrıca daha önceden oluşturulmuş olan karakter vektörleri (K) de eklendiğinde 390 boyutunda yeni bir vektör ($E \oplus M \oplus K$) elde edilebilmektedir. Bu son vektör kelimenin hem karakter hem de morfem bilgilerini içermektedir.

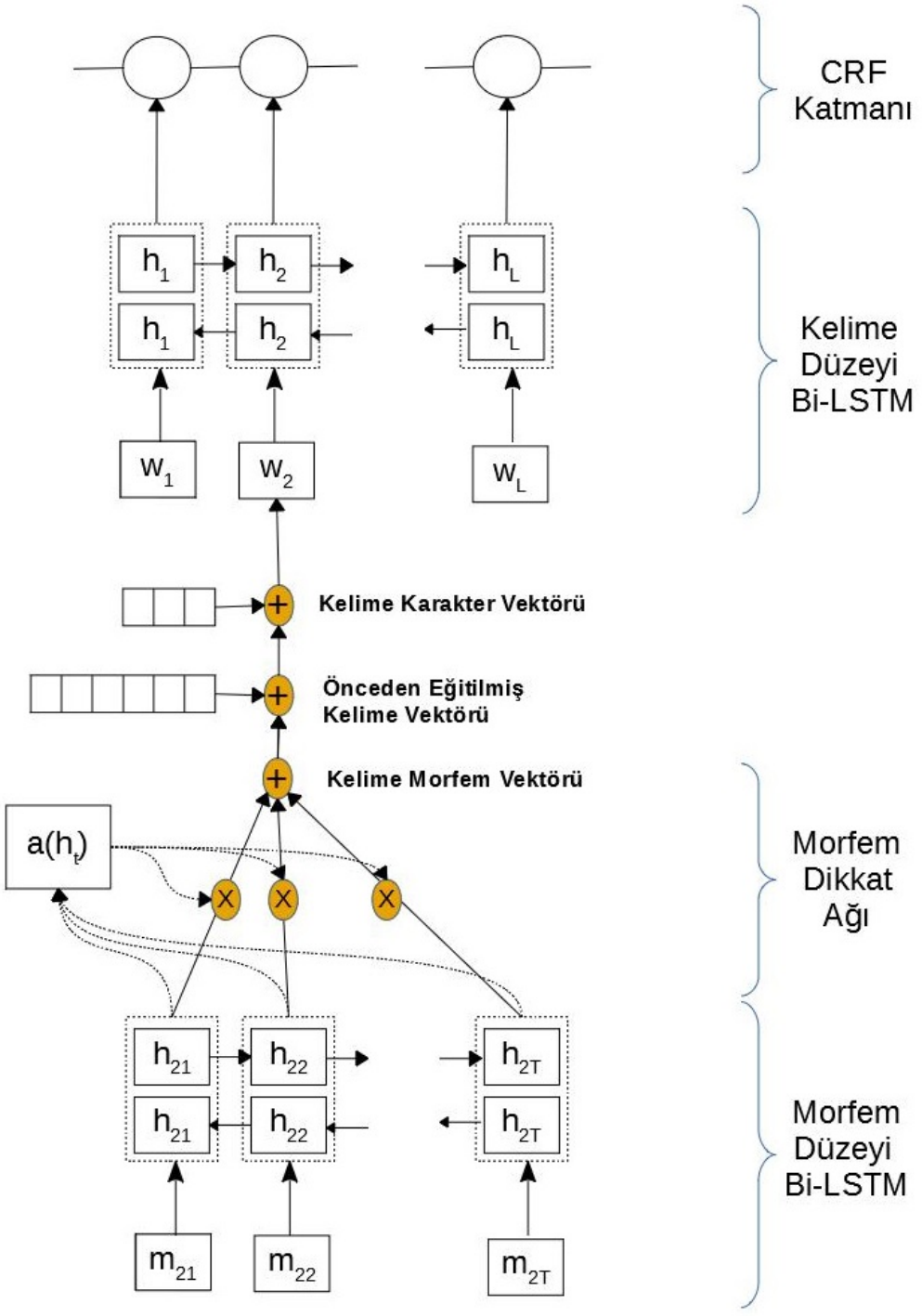
Morfem Vektörlerinin Çıkarımında Basitleştirilmiş Dikkat Ağı Kullanımı

Morfem seviyesindeki uzun kısa süreli bellek ağının sonucunda üretilen vektörün oluşturulması sırasında herhangi bir morfem diğerlerine göre daha fazla anlam içermemekte yani her morfem eşit öneme sahip olmaktadır. Ancak bazı morfemler bulundukları kelime için daha fazla anlam içerebilmektedirler. Dolayısıyla bu gibi durumlarda morfem vektörleri bulunurken kelime için daha anlamlı olan morfemlerin diğer morfemlerden daha büyük bir ağırlığa sahip olması gerekmektedir. Bölüm 2.4.'de teknik detayı verilmiş olan dikkat ağları tam da bu sebeple kullanılmaktadır. Burada da morfemlere farklı ağırlıklar vermek, belli morfemlere daha fazla dikkat etmek maksadıyla basitleştirilmiş dikkat ağı kullanılmıştır.

Şekil 4.2.'de gösterildiği gibi morfem düzeyindeki uzun kısa süreli bellek ağının (LSTM) her bir adımının çıktısı (h_t) bir dikkat fonksiyonundan (a) geçirilerek morfem etiketlerinin ağırlıkları (α_t) bulunmaktadır. Bulunan bu ağırlıklar yine LSTM adımlarının çıktısı ile çarpıldıktan sonra toplanarak, yani ağırlıklı toplamı hesaplanarak bağlam değeri (c) elde edilmektedir. Elde edilen "bağlam değeri aslında dikkat mekanizması kullanan modelin morfem vektörünü ifade etmektedir. Yani bir üst katmanda kelime ve karakter vektörleri ile birleştirilecek olan vektör doğrudan bu dikkat mekanizmasının çıktısıdır. Dikkat ağından geçirilerek elde edilen morfemler $A(M)$ şeklinde gösterilecektir. M morfem vektörlerini $A()$ ise dikkat fonksiyonunu ifade etmektedir.

4.2. LSTM-CRF Modeli

Uzun kısa süreli bellek ağı (LSTM) bir diziyi ($x_1 \dots x_t$) girdi olarak alıp dizinin her bir elemanı üzerinde aynı işlemi gerçekleştirmekte ve her adımda bir çıktı üretmektedir. Dizi etiketleme problemlerinde girdi olarak kullanılacak olan dizi, kelime vektörleridir ($w_1 \dots w_t$). Bu tez kapsamında önerilecek olan modelde ise LSTM katmanına sadece önceden eğitilmiş olan kelime vektörleri değil aynı zamanda karakter ve morfem vektörleri de girdi olarak verilmektedir. Ancak bu 3 vektör ayrı ayrı değil birleştirilerek tek bir vektör haline getirildikten sonra LSTM katmanına verilmektedir.



ŞEKIL 4.3. Önerilen modelin genel görünümü

Tez kapsamında önerilen ilk model Şekil 4.3.'de verilmiştir. Burada morfem vektörlerinin çıkarımı, karakter ve kelime vektörleri ile birleştirilerek LSTM katmanının girdisi olacak olan $(w_1 \dots w_t)$ değerlerinin oluşturulması ve LSTM katmanının çıktısının koşullu

rastgele alan (CRF) katmanına verilmesi gösterilmiştir.

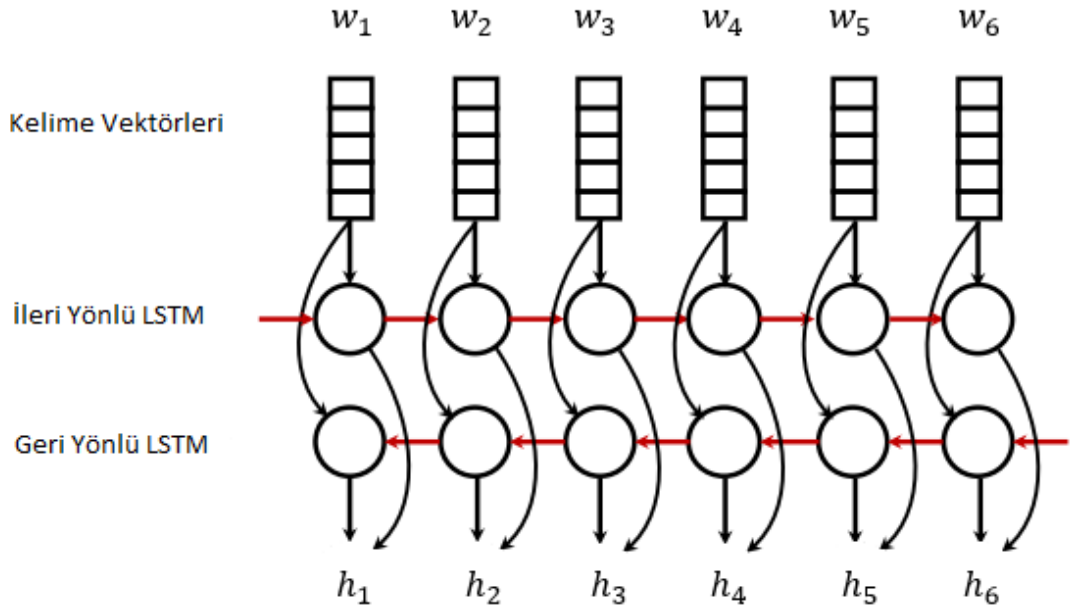
İlk model grubu 5 farklı modelden oluşmakta ve birbirlerinden aldıkları girdiler yönünden ayrışmaktadırlar:

- **Model 1 (LSTM-CRF + Word2Vec)**: Dizi etiketleme literatüründe yaygın kullanılan ve bu tez çalışmasında da dayanak noktası olarak alınan bu modelde uzun kısa süreli bellek ağının (LSTM) tek girdisi önceden eğitilmiş kelime vektörleridir.
- **Model 2 (LSTM-CRF + Word2Vec + CharVec)**: Model 1'den farklı olarak LSTM katmanı girdisine karakter vektörleri de eklenmiş olan modeldir.
- **Model 3 (LSTM-CRF + Word2Vec + MorphVec)**: Model 1'den farklı olarak LSTM katmanı girdisine dikkat mekanizması kullanılmadan üretilen morfem vektörleri de eklenmiş olan modeldir.
- **Model 4 (LSTM-CRF + Word2Vec + MorphAttVec)**: Model 3'den farklı olarak morfem vektörleri dikkat mekanizması kullanılarak elde edilmiş olan modeldir.
- **Model 5 (LSTM-CRF + Word2Vec + CharVec + MorphAttVec)**: Model 4'den farklı olarak LSTM katmanı girdisine karakter vektörleri de eklenmiş olan modeldir.

Tüm bu modellerde kullanılan uzun kısa süreli bellek ağının (LSTM) durum boyutu 200 olarak belirlenmiştir. Ayrıca yine tüm modellerde çift yönlü LSTM tercih edildiğinden LSTM'in her bir adımında ileriye doğru ve geriye doğru LSTM'lerden gelen vektörler birleştirilmiş ve 400 boyutundaki vektörler üretilmiştir. Bu birleştirme işlemi Şekil 4.4.'de gösterilmiştir.

4.2.1. Öz Dikkat Ağı

Bu tez çalışmasında önerilen modelde morfem vektörlerinin çıkarımı sırasında kullanılan basitleştirilmiş dikkat mekanizması dışında modeli daha da geliştirmek için bir başka

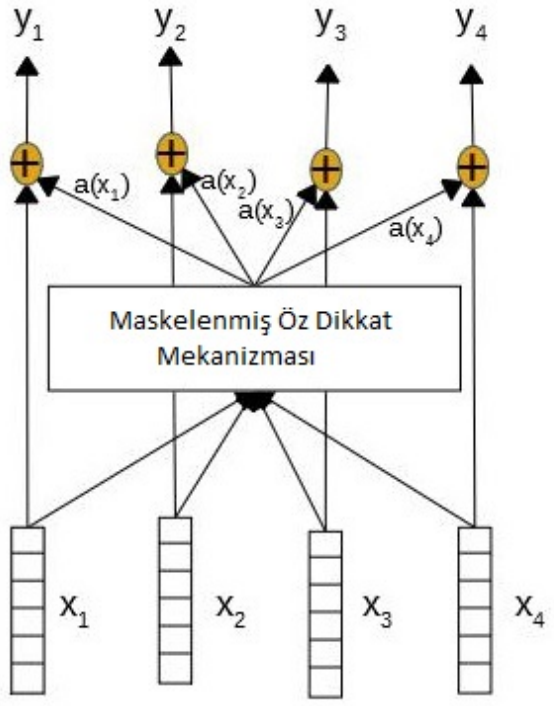


ŞEKIL 4.4. İleri ve geri yönlü LSTM ağlarının birleştirilmesi

dikkat ağı daha kullanılmıştır. Öz dikkat ağı mimarisinde olan bu dikkat mekanizması kelime düzeyindeki uzun kısa süreli bellek ağı çıktılarını kullanacak ve koşullu rastgele alan katmanına girdi olacak şekilde kurgulanmıştır.

Şekil 4.5.'de model görünümü verilmiş olan öz dikkat mekanizması uzun kısa süreli bellek ağı (LSTM) adımlarının çıktılarını $(x_1 \dots x_t)$ girdi olarak alıp bu vektörleri teknik detayı Bölüm 2.4.3.'de verilen bir a fonksiyonundan geçirerek öz dikkat mekanizmasının çıktılarını $(a(x_1) \dots a(x_t))$ üretmektedir. Koşullu rastgele alan (CRF) katmanı daha önce doğrudan kelime düzeyindeki LSTM'in çıktılarını kulllanırken öz dikkat mekanizmasının eklenmesinden sonra bu katmanın çıktıları ile LSTM çıktılarının artık bağlantı (residual connection) yöntemi ile birleştirilmesi ile elde edilen vektörleri $(y_1 \dots y_t)$ kullanmaktadır. Burada bahsedilmiş olan artık bağlantı yöntemi bir katmanın girdileri ile çıktılarının birleştirilmesini ifade etmektedir.

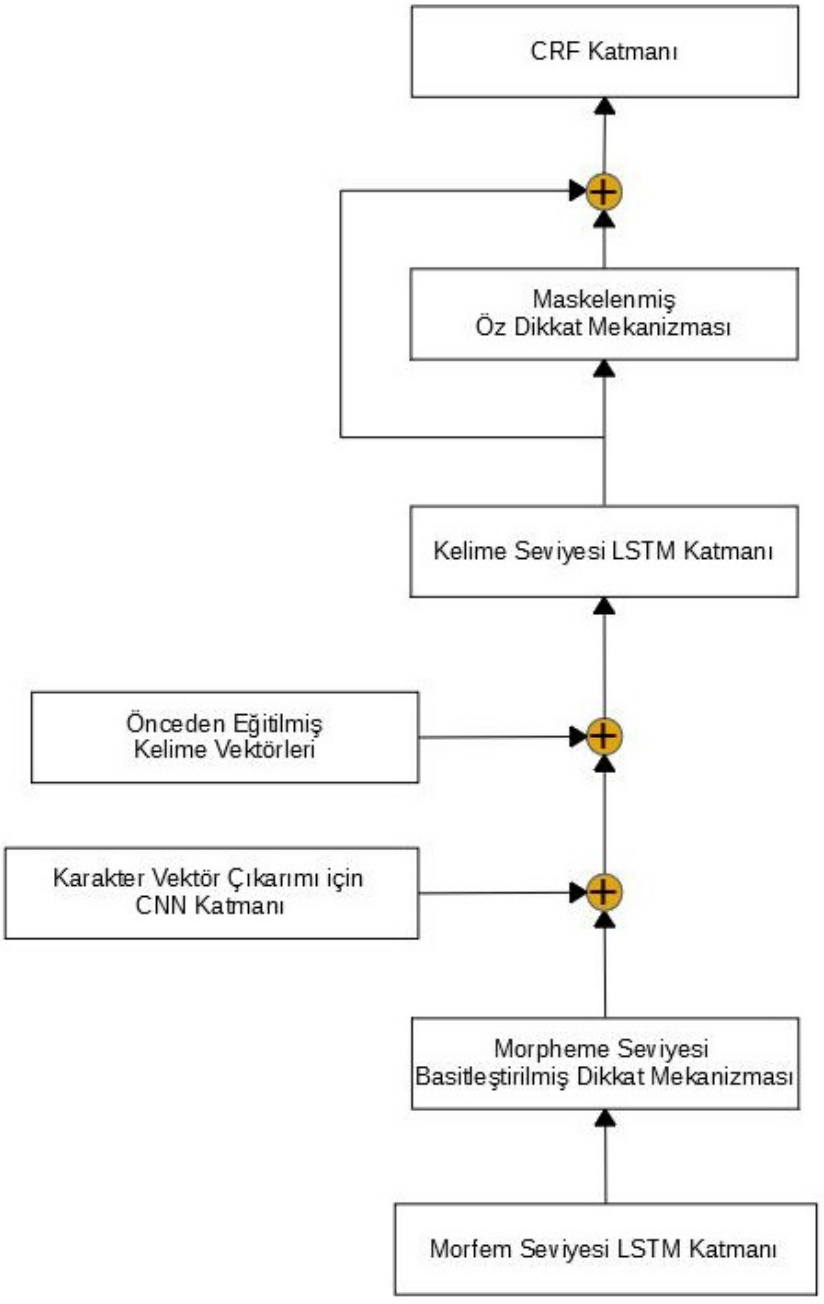
Morfem seviyesindeki kelime vektörlerinin çıkarımı sırasında kullanılan basitleştirilmiş dikkat mekanizmasında amaç bir kelimenin etiketinin bulunması sırasında o kelimenin hangi morfeminin daha belirleyici olduğunu bulmak ve o morfeme daha fazla dikkat etmek iken, LSTM çıktılarını kullanan öz dikkat ağında ise amaç bir kelimenin etiketinin bulunması sırasında diğer kelimelerin etkisini ve diğer kelimelerden hangisine daha fazla



ŞEKIL 4.5. Kelime ve morfem seviyesindeki LSTM çıktılarına uygulanacak öz dikkat mekanizması

dikkat edilmesi gerektiğini bulmaktır. Ancak dizi etiketleme problemlerinde her kelimenin etiketi doğal olarak en çok o kelimeye bağımlı olduğundan diğer kelimelerin etkisinin kaybolmaması için öz dikkat mekanizmasında her kelime, kendisi maskelenecek şekilde dikkat mekanizmasından geçirilmektedir. Böylece öz dikkat mekanizmasından sadece diğer kelimelere ait bilgiler gelmektedir. Ancak bu mekanizmanın çıktısı zaten artık bağlantı (residual connection) yöntemi sayesinde girdisi ile toplandığından ağın çıktısı kelimenin kendisine ait bilgileri de içermektedir.

Öz dikkat ağı katmanı eklendikten sonra elde edilen modelin tüm katmanları içeren son hali Şekil 4.6.'de verilmiştir. Bu model; uzun kısa süreli bellek ağları, evrişimli sinir ağı, farklı mimari kullanan dikkat ağları ve koşullu rastgele alan (CRF) gibi birçok farklı detayda katman içerdiğinden model gösteriminde kolaylık sağlaması için Şekil 4.1., Şekil 4.3. ve Şekil 4.5.'deki model katmanları bileşenlere çevrilerek gösterilmiştir. Ayrıca önerilen modellerin tüm varyasyonları aslında bu şekildeki bazı bileşenlerin eklenmesi veya çıkarılması ile oluşturulmuştur.



ŞEKIL 4.6. Önerilen modelin öz dikkat mekanizmaları ile genişletilmiş versiyonu

Bir önceki kısımda önerilen 5 modele öz dikkat ağı katmanı da eklenerek bu modeller biraz daha geliştirilmiş ve yine 5 modelden oluşan ikinci bir model grubu oluşturulmuştur:

- **Model 6 (LSTM-CRF-Öz Dikkat + Word2Vec)**: Kelime ve morfem vektörlerini içermeyen ve temel alınan LSTM-CRF modeline (Model 1) kelime düzeyinde öz dikkat makanizması eklenmesiyle oluşturulan modeldir. Yani bu model Şekil

4.6.'deki bileşenlerden sadece Kelime Vektörü, LSTM, Öz Dikkat ve CRF bileşenlerini içermektedir.

- **Model 7 (LSTM-CRF-Öz Dikkat + Word2Vec + CharVec)**: Model 6'dan farklı olarak LSTM katmanı girdisine karakter vektörleri eklenmiş, Model 2'den farklı olarak kelime düzeyinde öz dikkat makanizması kullanılmış olan modeldir.

- **Model 8 (LSTM-CRF-Öz Dikkat + Word2Vec + MorphVec)**: Model 6'dan farklı olarak LSTM katmanı girdisine dikkat mekanizması kullanılmadan üretilen morfem vektörleri eklenmiş, Model 3'den farklı olarak kelime düzeyinde öz dikkat makanizması kullanılmış olan modeldir.

- **Model 9 (LSTM-CRF-Öz Dikkat + Word2Vec + MorphAttVec)**: Model 8'den farklı olarak morfem vektörleri dikkat mekanizması kullanılarak oluşturulmuş, Model 4'den farklı olarak kelime düzeyinde öz dikkat makanizması kullanılmış olan modeldir.

- **Model 10 (LSTM-CRF-Öz Dikkat + Word2Vec + CharVec + MorphAttVec)**: Model 9'dan farklı olarak LSTM katmanı girdisine karakter vektörleri de eklenmiş, Model 5'den farklı olarak kelime düzeyinde öz dikkat makanizması kullanılmış olan modeldir.

4.3. Geliştirme Detayları

Bu tez kapsamındaki çalışmalar Python dili 3.7 versiyonu ve derin öğrenme kütüphanelerinden Keras [45] kullanılarak geliştirilmiştir. Keras 2.2 sürümü kullanılmıştır. Ayrıca veri kümelerindeki kelimelerin morfemleri ve morfem etiketleri Zemberek kütüphanesi [46] kullanılarak elde edilmiştir.

5. DENEYLER & SONUÇLAR

5.1. Veri Kümeleri

Dizi etiketleme problemleri üzerine yapılmış bu tez çalışmasında varlık ismi tanıma ve sözcük türü işaretleme problemleri üzerine çalışılmış ve önerilen modeller bu problemler üstünde denenmiştir. Bu amaçla önerilen modeller varlık ismi etiketleri ve sözcük türü etiketleri içeren iki farklı veri kümesi üzerinde denenmiştir.

Türkçe için varlık isimlerini tanıma problemi (NER) üzerine yapılan çalışmaların birçoğunda aynı veri kümesi [37] kullanılmaktadır. Bu çalışmada da önerilen modellerin diğer çalışmalarla karşılaştırılabilmesi için aynı veri kümesi tercih edilmiştir. Gazete makalelerinden derlenmiş olan bu veri kümesindeki varlık isimleri ENAMEX tipinde işaretlenmiştir. Veri kümesinde yaklaşık olarak 500 bin kelime, 37 bin kadar da varlık ismi bulunmaktadır. Bu veri kümesinde bulunan varlık isimlerinin tiplerine göre dağılımı Tablo 5.1.'de verilmiştir.

TABLO 5.1. Kullanılan varlık ismi veri kümesindeki [37] varlık isimlerinin varlık tipine göre dağılımı

Varlık Tipi	Varlık Sayısı
Kişi	16.297
Konum	10.889
Organizasyon	10.033
Toplam	**37.219**

Bir diğer problem olan sözcük türü işaretleme için ise, METU-Sabancı Türkçe Treebank [1] kullanılmıştır. Bu veri kümesinde 5635 cümle, 56.424 tane sözcük türü işaretlenmiş kelime bulunmaktadır.

Her iki veri kümesi de 10 parçaya ayrılmış, bu parçalardan birisi doğrulama (validation), birisi test için kullanılırken geri kalanlar ise eğitim (training) amaçlı kullanılmıştır. Geliştirilen modeller her seferinde bu parçalar bir kaydırılarak 10 sefer çalıştırılmış böylece her seferinde farklı bir veri kümesi üzerinde çalışması sağlanmıştır. Bu yönteme 10 parçalı çapraz doğrulama (10-fold cross validation) adı verilir.

Modellerde kullanılan kelime vektörleri model içerisinde eğitilmemiş olup, daha önceden Skip-gram algoritması ile eğitilmiş hazır vektörlerdir [42]. Bu hazır vektör kümesinde 1.238.485 kelimeye ait vektör bulunmaktadır. Varlık ismi tanıma problemi için kullanılan veri kümesinde [37] yer alan toplam 70.530 adet farklı kelimenin 65.983 tanesinin, sözcük türü işaretleme problemi için kullanılan veri kümesinde [1] yer alan toplam 19.125 adet farklı kelimenin 18.072 tanesinin kelime vektörleri önceden eğitilmiş kelime vektör kümesinde yer almaktadır. Vektör kümesinde bulunmayan sözlük harici kelimeler (out of vocabulary words) içinse değerleri -0,25 ile +0,25 arasında olan rastgele tek düze vektörler oluşturulmuştur.

Veri kümeleri herhangi bir ön işlem (preprocessing) yapılmadan kullanılmıştır. Ancak veri kümelerinde morfem etiketleri bulunmadığından modeller çalıştırılmadan önce veri kümelerindeki tüm kelimelerin morfem etiketleri Zemberek [46] kütüphanesi kullanılarak elde edilmiştir.

5.2. Deneyler

Bu bölümde öncelikle geliştirilen modelin çalıştırılması sırasında tercih edilen parametrelerden ve sonuçları iyileştirmek ve düzenlileştirmek (regularization) için kullanılan yöntemlerden bahsedilecektir. Sonrasında sonuçları karşılaştırmak için tercih edilen değerlendirme yöntemleri açıklanacaktır. Son olarak üzerinde çalışılan her iki problem için elde edilmiş sonuçlar paylaşılacaktır.

5.2.1. Deney Parametreleri ve Eğitim

Modelin girdilerini oluşturan kelime, karakter ve morfem vektörleri farklı boyutlarda tutulmaktadır. Kelime vektörü olarak Skip-gram algoritması kullanılarak daha önceden eğitilmiş, her kelimenin 300 boyutundaki vektörlerle temsil edildiği, hazır bir Word2Vec vektör kümesi [42] kullanılmıştır. Bu çalışmada kısaca karakter vektörleri olarak adlandırılan, kelimelerin karakter özelliklerini temsil eden ve evrişimli sinir ağı (CNN) ile çıkarılmış olan vektörlerin boyutu 30 olarak belirlenmiştir. Kelimelerin morfem özelliklerini tutan,

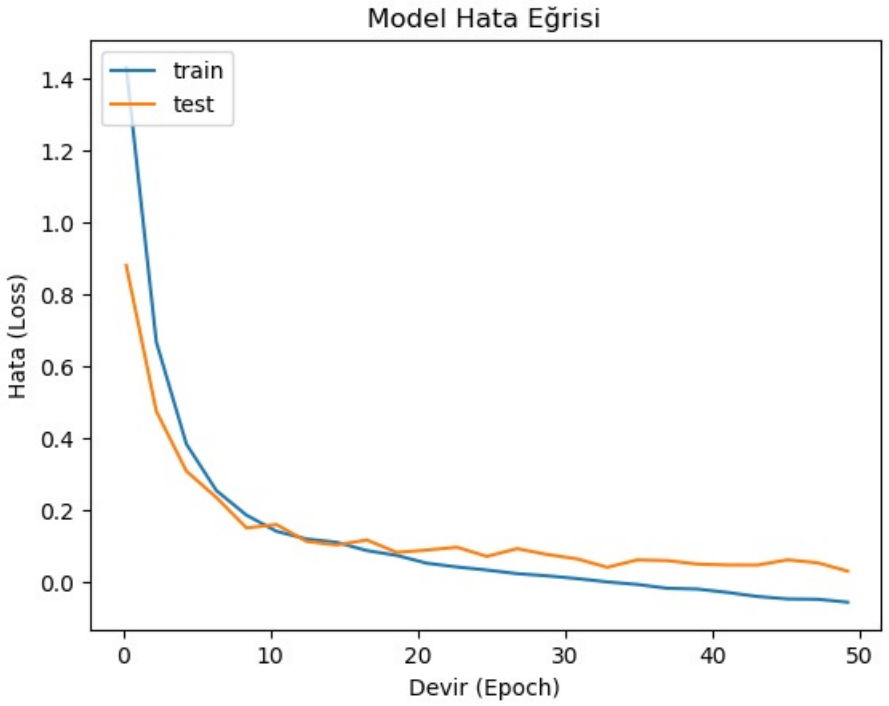
modeldeki morfem katmanının çıktısı olan morfem vektörlerinin boyutu ise 60 olarak belirlenmiştir. Böylece, kelime vektörleri, karakter vektörleri ve morfem vektörleri uç uca eklenerek 390 boyutlu bir vektör elde edilmiştir.

Modelde kelime düzeyi ve morfem düzeyi olmak üzere iki farklı seviyede uzun kısa süreli bellek ağı (LSTM) kullanılmıştır. Kelime düzeyindeki LSTM katmanında durum boyutu 200 olarak belirlenirken morfem düzeyinde ise durum boyutu 30 olarak seçilmiştir. Her iki katmanda da çift yönlü LSTM kullanılmasından ve her iki yönün çıktılarının uç uca eklenerek sonucun bulunmasından dolayı bu katmanların çıktısından 400 ve 60 boyutlu vektörler elde edilmiştir. Modelde karakter vektörlerinin çıkarılması ise evrişimli sinir ağı kullanılarak gerçekleştirilmiştir. Bu katmanda filtre boyutu 3 olarak seçilirken, atlama adım aralığı ise 1 olarak belirlenmiştir.

Makine öğrenmesi ve derin öğrenme modelleri geliştirilirken modelin eğitim verisinde çok iyi sonuçlar verirken deneme yapılan farklı bir veri kümesi üzerinde çok daha düşük sonuçlar verebildiği durumlar olmaktadır. Bunun en temel nedenlerinden birisi aşırı uyum (overfitting) ya da eğitim verisini ezberleme olarak açıklanan durumdur. Yani model eğtim veri kümesini çok iyi öğrenmiş, ezberlemiş olsa da öğrenilen bilgiler diğer veri kümelerini de yorumlamaya yetecek bir genelliğe sahip değildir. Bu sorunu çözmek için çeşitli düzenlileştirme teknikleri (regularization) uygulanmaktadır. Bu tez çalışmasındaki birçok katmanda da bir düzenlileştirme tekniği olan seyreltme (dropout) yöntemi kullanılmıştır.

Kelime düzeyindeki uzun kısa süreli bellek ağının (LSTM) girdilerine (X_t) 0.5, LSTM hücreleri arasında transfer edilen gizli durum (h_t) ve hücre durumu (C_t) değerlerine 0.25 oranında seyreltme uygulanmıştır. Morfem düzeyindeki LSTM ağının ise hem girdilerine hem de hücreler arasında transfer edilen gizli durum ve hücre durumu değerlerine 0.4 oranında seyreltme uygulanmıştır. Karakter vektörlerinin çıkarımında kullanılan evrişimli sinir ağının da hem girdilerine hem de çıktıların 0.5 oranında seyreltme uygulanmıştır. Seyreltme uygulanan son bir nokta da öz dikkat ağının çıktısıdır. Burada kullanılan seyreltme oranı ise 0.1'dir. Bu değerler bir dizi deney ile elde edilmiş sonuçlara göre belirlenmiştir.

Önerilen modelle ilgili son değinilmesi gereken parametre öz dikkat ağı maskeleme ya da pencere boyutudur. Bu parametre sayesinde bir cümleye öz dikkat mekanizması uygulanırken sadece belirli bir komşuluktaki kelimelere bakılması sağlanmaktadır. Bu modelde bu değer 10 olarak belirlenmiştir. Yani öz dikkat ağı katmanında her kelime için sadece kendisinden önce ve sonra gelen 5 kelime anlam ifade etmektedir.

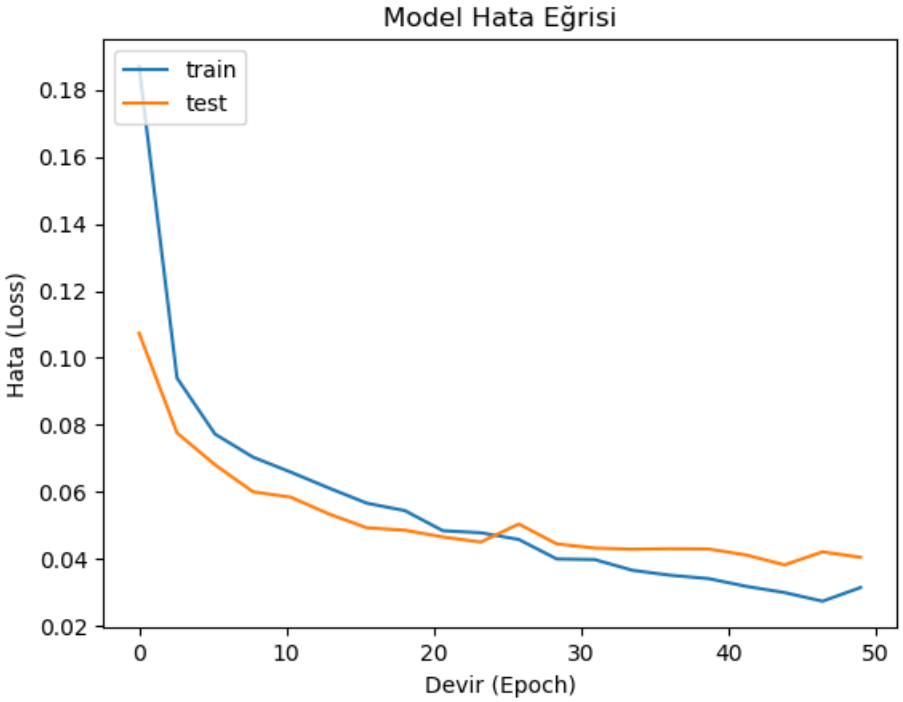


ŞEKIL 5.1. Sözcük türü işaretleme problemi için modelin hata eğrisi

Modellerin optimizasyonu için rmsprop yöntemi kullanılmıştır. Ayrıca eğitim esnasında her bir model için derlemin üzerinden 50 kez (epoch) geçilmiştir. Modelin üzerinden her geçişten sonra kayıp (losss) değerleri kullanılarak hesaplanan hata eğrileri Şekil 5.1. ve Şekil 5.2.'de verilmiştir.

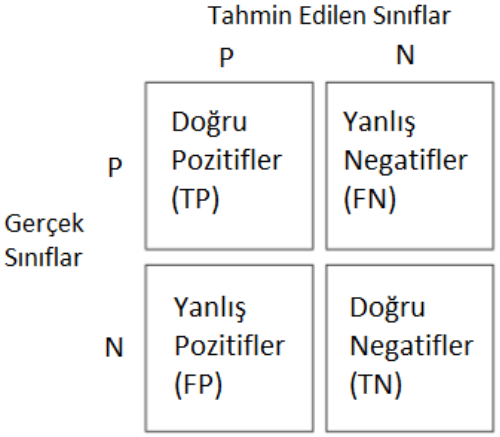
5.2.2. Değerlendirme

Bu tez kapsamında çalışılan dizi etiketleme problemleri her ne kadar birebir aynı modeli, aynı parametreleri kullanıyor olsalar da bu iki problemin değerlendirilme yöntemleri birbirinden farklıdır.



ŞEKIL 5.2. Varlık ismi tanıma problemi için modelin hata eğrisi

Model, varlık ismi tanıma problemi kapsamında değerlendirilirken tercih edilen yöntem $F1$ skorudur. Bu değer duyarlılık (recall) ve kesinlik (precision) değerleri kullanılarak bulunmaktadır. Tüm bu değerler Şekil 5.3.'de verilen doğruluk matrisindeki değerler kullanılarak hesaplanmaktadır:



ŞEKIL 5.3. Modellerin değerlendirilmesinde kullanılan kesinlik, duyarlılık ve F1 skorunun bulunmasında kullanılan doğruluk matrisi

- Doğru Pozitif (TP): Bulunması gerekip model tarafından da doğru bulunan değerlerdir.
- Yanlış Negatif (FN): Bulunması gerekirken model tarafından bulunamayan değerlerdir.
- Yanlış Pozitif (FP): Bulunmaması gerekirken model tarafından hatalı olarak bulunan değerlerdir.
- Doğru Negatif (TN): Bulunmaması gerekip model tarafından da bulunamayan değerlerdir.

Duyarlılık değeri bulunması gereken değerlerin ne kadarının bulunduğunu; kesinlik değeri ise bulunan değerlerin ne kadarının gerçekten bulunması gerektiğini ifade etmektedir. Bu iki skor kullanılarak bulunan F1 skoru şu şekilde hesaplanmaktadır:

$$duyarlilik = \frac{TP}{TP + FN} \tag{19}$$

$$kesinlik = \frac{TP}{TP + FP} \tag{20}$$

$$F1 = 2 * \frac{duyarlilik * kesinlik}{duyarlilik + kesinlik} \tag{21}$$

Sözcük türü işaretleme probleminde, problemin doğası gereği kelime için bulunan etiket ya doğrudur ya da yanlıştır. Yani bulunması gereken ya da bulunmaması gereken gibi kavramlar olmadığından yanlış pozitif (FP) ve yanlış nagatif (FN) değerleri aynıdır. Dolayısı ile bu problem için modeli değerlendirirken sadece doğruluk değerine bakmak yeterli olmaktadır. Doğru bulunan kelime etiketi sayısının derlemdeki tüm kelime sayısına oranı bu doğruluk değerini vermektedir.

5.2.3. Deney Sonuçları

Bu tez kapsamındaki önerilen modelin (Şekil 4.6.) Bölüm 4.2.'de de anlatıldığı gibi farklı versiyonları oluşturulmuştur. Model, öz dikkat ağının kullanılıp kullanılmamasına göre iki ayrı gruba ayrılırken, girdi olarak alınan vektörlere göre ise her grupta beşer farklı modele ayrılmaktadır. Tüm bu modeller bu ayrımlarına göre Tablo 5.2.'de gösterilmiştir. Bu

tablodaki CharVec ifadesi evrişimli sinir ağı ile elde edilen karakter vektörlerini, MorphVec ifadesi çift yönlü uzun kısa süreli bellek ağı ile çıkarılmış olan morfem vektörlerini, MorphAttVec ifadesi ise basitleştirilmiş dikkat mekanizması da kullanılarak çıkarılmış morfem vektörlerini ifade etmektedir.

TABLO 5.2. Tez çalışmasında önerilen tüm modellerin mimari farklılıkları

	Model Grubu	
Girdi Vektörleri	LSTM-CRF	LSTM-CRF-Öz Dikkat
Word2Vec	Model 1	Model 6
Word2Vec + CharVec	Model 2	Model 7
Word2Vec + MorphVec	Model 3	Model 8
Word2Vec + MorphAttVec	Model 4	Model 9
Word2Vec + CharVec + MorphAttVec	Model 5	Model 10

Varlık ismi tanıma problemi için modelleri kıyaslarken F1 skoruna bakmak sadece doğruluk değerine bakmaktan daha sağlıklı bir karşılaştırma yapılmasını sağladığı için önerilen modeller F1 skorları ile karşılaştırılmıştır. Öz dikkat ağı katmanını içermeyen modellerden (Model 1-5) varlık ismi tanıma problemi için kullanılan veri kümesi [37] üzerinde elde edilen kesinlik (precision), duyarlılık (recall) ve F1 skorları Tablo 5.3.'de, öz dikkat ağı katmanını da içeren modellerden (Model 6-10) aynı veri kümesi üzerinde elde edilen kesinlik, duyarlılık ve F1 skorları ise Tablo 5.4.'de verilmiştir.

TABLO 5.3. Öz dikkat ağı içermeyen modellerin (Model 1-5) varlık ismi tanıma problemi için kullanılan veri kümesi [37] üzerinde elde ettiği kesinlik, duyarlılık ve F1 değerleri

Girdi Vektörleri	LSTM-CRF		
	Kesinlik (%)	Duyarlılık (%)	F1 (%)
Word2Vec	92,58	93,49	93,03
Word2Vec + CharVec	92,14	94,35	93,23
Word2Vec + MorphVec	92,45	92,47	93,35
Word2Vec + MorphAttVec	93,93	93,43	93,68
Word2Vec + CharVec + MorphAttVec	93,50	93,92	**93,71**

Hem Tablo 5.3. hem de Tablo 5.4.'de görüldüğü üzere model girdilerinde karakter ve morfem vektörleri kullanmak, temel alınan ve sadece kelime vektörleri kullanan modellere (Model 1 ve Model 6) göre sonuçları artırmıştır. Morfem vektörü kullanmanın etkisi karakter vektörü kullanmaya göre daha fazla olmuştur. Özellikle basitleştirilmiş dikkat

TABLO 5.4. Öz dikkat ağı içeren modellerin (Model 6-10) varlık ismi tanıma problemi için kullanılan veri kümesi [37] üzerinde elde ettiği kesinlik, duyarlılık ve F1 değerleri

Girdi Vektörleri	LSTM-CRF-Öz Dikkat		
	Kesinlik (%)	Duyarlılık (%)	F1 (%)
Word2Vec	93,19	93,31	93,25
Word2Vec + CharVec	92,89	93,95	93,42
Word2Vec + MorphVec	92,62	94,61	93,61
Word2Vec + MorphAttVec	93,97	93,42	93,69
Word2Vec + CharVec + MorphAttVec	94,68	93,37	**94,02**

ağından geçirilerek elde edilen morfem vektörleri daha iyi sonuçlar vermiştir. Her iki model grubu için de en iyi sonuçlar hem karakter hem de dikkat ağından geçirilmiş morfem vektörlerinin kullanıldığı modeller (Model 5 ve Model 10) olmuştur.

Öz dikkat katmanı içeren ve içermeyen bu iki model grubu kıyaslandığında öz dikkat ağı kullanmanın tüm farklı girdiye sahip modeller için sonuçları iyileştirdiği görülmektedir. Öz dikkat ağı katmanının eklenmesi tüm model versiyonları için sonuçlarda önemli bir iyileşme sağlamıştır.

Önerilen modellerden %94,02 ile en yüksek sonuç veren model (Model 10) daha önce Türkçe için yapılmış diğer çalışmalarla kıyaslanmış ve bu sonucun önceki çalışmalardan daha yüksek olduğu görülmüştür. Diğer çalışmalarla yapılan karşılaştırmalar Tablo 5.5.'de verilmiştir. Bu modeller arasından bu tezde önerilen modellere en yakın sonuçlar Güngör ve diğ. [32, 39]'nin çalışmaları olarak gözükmektedir. Bu çalışmalarda da bu tezde önerilen modele benzer bir şekilde morfem özellikleri modele dahil edilmiştir. Bizim çalışmamızdaki modelin önceki çalışmalardan en önemli farkı hem morfem seviyesinde hem de kelime seviyesinde farklı dikkat ağı mimarileri kullanmasıdır. Ayrıca bu tezde önerilen modeller hiçbir ek özellik ya da kaynak kullanmazken Güneş ve Tantuğ (2018) [38] gibi önceki kimi çalışmalarda büyük/küçük harf, noktalama işareti gibi yazım özellikleri de modele eklenmiştir.

Tez kapsamında çalışılan diğer bir problem olan sözcük türü işaretleme için de aynı şekilde önerilen tüm modeller denenmiştir. Varlık ismi tanıma probleminden farklı olarak önerilen modeller bu problem için doğruluk (accuracy) değeri kullanılarak karşılaştırılmıştır. Önerilen 10 modelin tamamına ait doğruluk değerleri Tablo 5.6.'da verilmiştir.

TABLO 5.5. Önerilen en iyi modelin (Model 10) literatürdeki Türkçe için yapılmış olan varlık ismi tanıma çalışmaları ile kıyaslanması

Model	F1 Skoru (%)
Şeker ve Eryiğit (2012) [27]	91,94
Demir ve Özgür (2014) [47]	91,85
Kuru ve diğ. (2016) [48]	91,30
Güneş ve Tantuğ (2018) [38]	93,69
Güngör ve diğ. (2018) [32]	93,37
Güngör ve diğ. (2019) [39]	92,93
Önerilen Model (Model 10)	**94,02**

TABLO 5.6. Önerilen tüm modellerden (Model 1-10) sözcük türü işaretleme problemi için kullanılan veri kümesi [1] üzerinde elde edilen doğruluk (accuracy) değerleri

Girdi Vektörleri	LSTM-CRF	LSTM-CRF-Öz Dikkat
	Doğruluk (%)	Doğruluk (%)
Word2Vec	92,57	93,62
Word2Vec + CharVec	93,85	93,85
Word2Vec + MorphVec	95,56	95,69
Word2Vec + MorphAttVec	**95,71**	**95,77**
Word2Vec + CharVec + MorphAttVec	95,68	95,71

Varlık ismi tanıma probleminde olduğu gibi sözcük türü işaretleme probleminde de karakter ve morfem vektörlerinin eklenmesi sonuçlardaki doğruluğu artırmıştır. Hatta sözcük türü işaretleme için bu artış daha fazla olmuştur. Burada varlık ismi tanıma probleminden farklı olarak ilginç bir şekilde hem karakter hem de morfem vektörleri eklenmiş olan modeller (Model 5 ve Model 10) sadece dikkat ağından geçirilerek oluşturulan morfem vektörlerini girdi alan modellerden (Model 4 ve Model 9) ufak bir farkla da olsa daha kötü sonuçlar vermiştir. Bu da karakter vektörlerinin varlık ismi tanıma problemi için daha anlamlı olduğu sonucunu ortaya koymaktadır. Varlık isimlerindeki kelimelerin ilk harflerinin genelde büyük harfle başlıyor oluşu, evrişimli sinir ağının büyük harf gibi bilgileri de kodlayarak karakter vektörlerini çıkarıyor olması bunun en önemli sebebi olarak değerlendirilmiştir.

Sözcük türü işaretleme deneylerinde öz dikkat ağı katmanının eklenmesi, sadece kelime vektörünü girdi olarak alan model (Model 1) hariç sonuçları beklendiği kadar çok artırmamıştır. Kelimelerin sözcük türlerinin, kelimelerin varlık isim etiketlerine nazaran diğer kelimelerle daha az ilişkisinin olmasının böyle bir sonuca neden olduğu düşünülmüştür.

TABLO 5.7. Önerilen en iyi modelin (Model 9) literatürdeki Türkçe için yapılmış olan sözcük türü işaretleme çalışmaları ile kıyaslanması

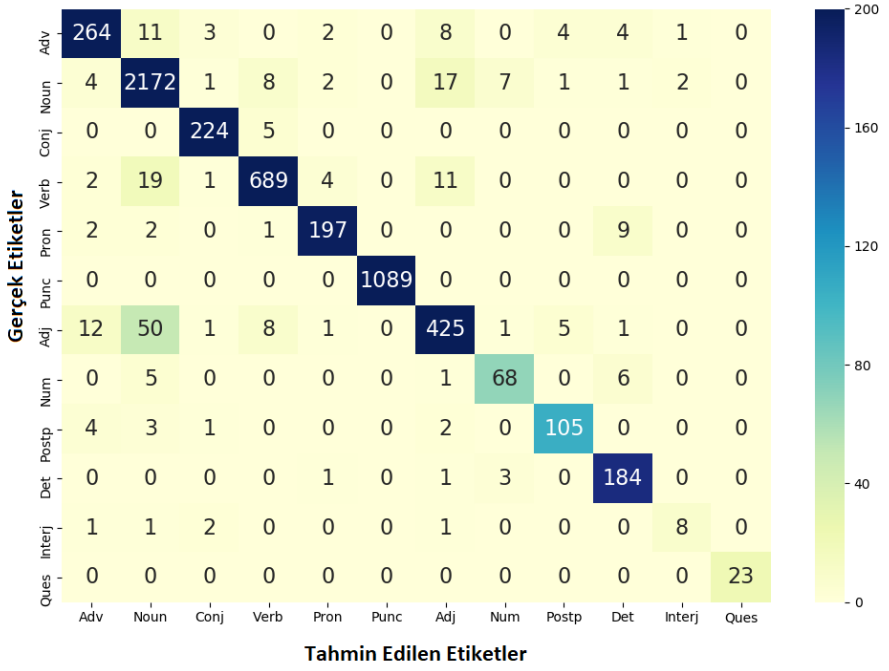
Model	Doğruluk (%)
Dinçer ve diğ. (2008) [40]	88,90
Can ve diğ. (2016) [49]	91,05
Bahçevan ve diğ. (2018) [41]	89,00
Önerilen Model (Model 9)	**95,71**

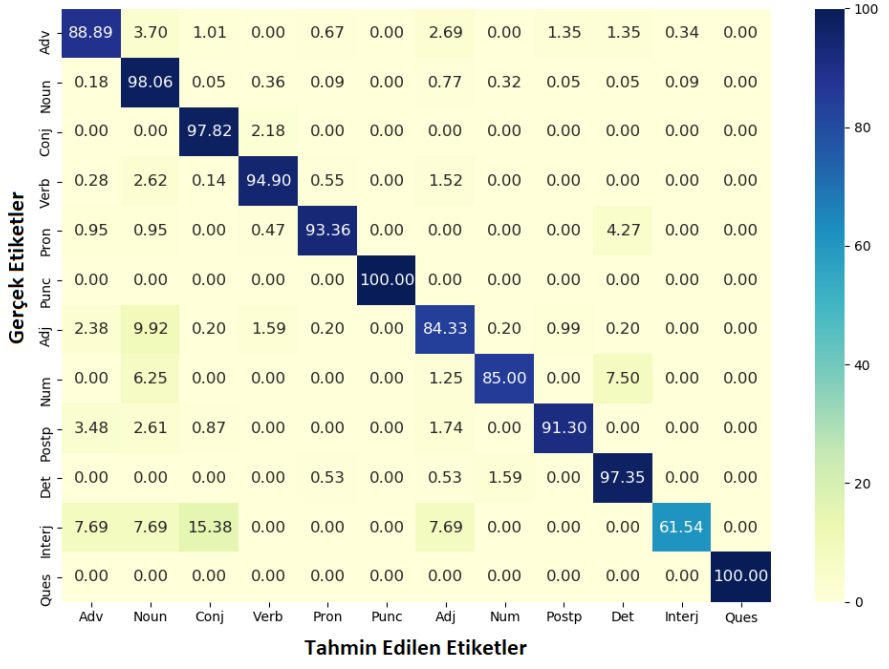
Sözcük türü işaretleme problemi için elde edilen en iyi doğruluk değerine (%95,77) sahip model (Model 9) literatürde aynı veri kümesini kullanarak geliştirilmiş diğer çalışmaların sonuçları ile kıyaslanmıştır. Bu çalışmaların sonuçları Tablo 5.7.'de verilmiştir. Morfem tabanlı olarak önerdiğimiz model ile bu çalışmalar arasında ciddi bir başarı farkı görülmektedir. Bu da morfem bilgisinin sözcük türü işaretlemede önemli bir katkısı olduğunu göstermektedir.

Çalışılan her iki problem için de önerilen en iyi model ile bulunan etiketlerin ne kadarının doğru ne kadarının yanlış bulunduğunu ve hataların daha çok nerelerde olduğunu anlamak için karmaşıklık matrisleri (confusion matrix) çıkarılmıştır. Şekil 5.4.'de sözcük türü işaretleme problemi için oluşturulmuş karmaşıklık matrislerine yer verilmiştir. Bu matrislerden ilkindeki değerler test veri kümesindeki her bir etiketin model ile bulunan etiketlere göre sayısal dağılımını ifade ederken ikinci matristeki değerler ise bu dağılımı yüzdelik olarak göstermektedir.

Sözcük türü işaretleme problemi için oluşturulan karmaşıklık matrislerine bakıldığında sadece çok az verisi olan etiket tiplerinde sapmalar olduğu diğer etiket tiplerinde ise başarının yüksek olduğu görülmektedir. Ayrıca noktalama işaretleri ve soru ekleri gibi hep belirli bir karakter kümesinin kullanıldığı öğelerdeki başarı ise %100 olarak saptanmıştır. Kelimelerin karakter özelliklerini içeren karakter vektörlerinin bu tip etiketlerde belirleyici olduğu görülmüştür.

Varlık isimi tanıma problemi için oluşturulmuş karmaşıklık matrislerine de Şekil 5.5.'de yer verilmiştir. Görüldüğü üzere bu problem türünde daha sık görülen hatalar varlık isimlerinin yanlış etiketlenmesi değil bu varlıkların hiç bulunamamasıdır. Varlık isimlerin yanlış etiketlenmesindeki en büyük hata %2,50 ile yer adlarının kurum adı olarak etiketlenmiş olmasıdır. Varlık isimlerin bulunamamasında ise hata oranı %6'lara yaklaşmaktadır.





ŞEKIL 5.4. Sözük türü işaretleme problemi için oluşturulan karmaşıklık matrisleri - İlk grafik her bir etiketin model ile bulunan etiketlere göre dağılımını, ikinci grafik ise bu dağılımın yüzdelik değerlerini göstermektedir.

ŞEKIL 5.5. Varlık ismi tanıma problemi için oluşturulan karmaşıklık matrisleri - İlk grafik her bir etiketin model ile bulunan etiketlere göre dağılımını, ikinci grafik ise bu dağılımın yüzdelik değerlerini göstermektedir.

6. SONUÇ

6.1. Sonuç

Türkçe gibi sondan eklemeli dillerde aynı kökten birden fazla kelime türetilebilmesi doğal dil işleme alanındaki çalışmalarda seyreklik sorununa neden olmaktadır. Bu seyreklikten dolayı veri kümelerinde fazla sayıda nadir geçen veya sözlük harici kelimeye denk gelinmektedir. Nadir geçen kelimelerin vektörlerinin genelde iyi eğitilmemiş olması ve sözlük harici kelimelerin vektörlerinin rastgele oluşturulmasından dolayı doğal dil işleme alanında sondan eklemeli diller üzerinde yapılan çalışmalar diğer dillerde olduğu kadar başarıya ulaşamamaktadır. Bu tezde bu sorunu çözmek için kelimelerin karakterlerinden ve morfemlerinden çıkarılacak bilgilerin de modele dahil edilmesi önerilmiştir.

Bu tez çalımasında Türkçe için dizi etiketleme problemlerinden olan varlık ismi tanıma ve sözcük türü işaretleme problemleri üzerinde çalışılmıştır. Bu kelime dizilerini etiketleme problemleri için bir derin öğrenme modeli önerilmiştir. Bu amaçla, dizi etiketleme literatüründe yaygın kullanılan LSTM-CRF katmanlarından oluşan derin öğrenme mimarisi kullanılmıştır. Literatürdeki kimi çalışmalar gibi kelimelerin karakter özelliklerini içeren karakter vektörleri ve kelimelerin morfem özelliklerini içeren morfem vektörleri de LSTM-CRF modelinde kullanılmıştır. LSTM-CRF modelindeki LSTM katmanına ek olarak morfem vektörleri de LSTM ağı ile çıkarıldığı ve bu katmanın çıktıları diğer vektörlerle de toplandıktan sonra kelime düzeyindeki LSTM ağının girdilerini oluşturdukları için önerilen model hiyerarşik bir mimariye de sahiptir.

Bu çalışmanın diğer çalışmalardan en temel farklarından birisi morfem vektörlerinin çıkarımı için morfem düzeyindeki LSTM çıktılarını basitleştirilmiş dikkat ağı mekanizmasından geçirmesidir. Yine kelime düzeyindeki LSTM çıktıları da bir başka dikkat ağı mimarisi olan öz dikkat mekanizmasından geçirilmektedir. Kullanılan dikkat ağları sayesinde bir kelimenin etiketinin bulunmasında o kelimenin hangi morfemlerinin daha önemli olduğu ve belirli bir komşulukta olmak kaydıyla diğer kelimelerin mevcut kelimenin etiketinde ne kadar öneme sahip olduğu bulunmaktadır.

Bu farkları da göz önünde bulundurarak bu tez çalışmasının katkıları şu şekilde ifade edilebilir:

- Başta İngilizce olmak üzere diğer dillerde uygulanan derin öğrenme tabanlı dizi etiketleme modellerini Türkçenin morfolojik yapısına göre uyarlamak
- Morfem seviyesinde kullanılan dikkat ağı ile morfem vektörlerinin kelimelerin morfem bilgisi hakkında daha fazla detay içermesini sağlayarak modellerde daha yüksek başarı elde etmek
- Kelime düzeyinde kullanılan öz dikkat mekanizması ile kelime etiketlerinin bulunmasında belirli bir komşuluktaki kelimelerden de yararlanılması

Önerilen model sözcük türü işaretleme ve varlık ismi tanıma olmak üzere iki farklı dizi etiketleme problemi için test edilmiştir. Her iki problem için de önceki çalışmalara göre daha iyi sonuçlar elde edilmiştir. Varlık ismi tanımada kelime, karakter ve morfem vektörlerinin bir arada kullanılmasıyla en iyi sonuçlar elde edilirken, sözcük türü işaretleme probleminde en iyi sonuçlar kelime ve morfem vektörleri kullanıldığında alınmıştır.

6.2. Gelecek Çalışmalar

Son aylarda doğal dil işleme konusunda birçok büyük gelişme yaşanmıştır. Bu gelişmelerin en önemlileri arasında ELMo [50] ve BERT [51] gibi çalışmalar bulunmaktadır.

Bu yöntemler her kelimeyi tek bir vektörle göstermek yerine kelimenin vektörünü içinde bulunduğu bağlama göre oluşturmaktadır. Yani Word2Vec gibi yöntemlerden farklı olarak yazılışı aynı anlamı farklı kelimeleri tek bir vektör olarak tutmamakta, bulunduğu bağlama göre dinamik olarak oluşturulabilmektedir.

ELMo, BERT gibi bağlamı da dikkate alan bu güncel ve yeni vektörel yöntemleri dizi etiketleme problemlerine Türkçenin morfolojik yapısını da göz önünde bulundurarak uygulamak, yapılabilecek olan gelecek çalışmaları arasında değerlendirilebilir.

KAYNAKLAR

[1] K. Oflazer, B. Say, D.Z. Hakkani-Tür, and G. Tür. Building a turkish treebank. In *Treebanks*, pages 261–277. Springer, **2003**.

[2] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning*, pages 147–155. Association for Computational Linguistics, **2009**.

[3] G. Luo, X. Huang, C. Lin, and Z. Nie. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888. **2015**.

[4] J. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. **2001**.

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, **2011**.

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

[7] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. **2013**.

[8] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. **2014**.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, **2017**.

[10] A. Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Erişim tarihi: 2019-06-01.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, **1997**.

[12] C. Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs`. Erişim tarihi: 2019-06-01.

[13] I. Sutskever, O. Vinyals, and Q.V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112. **2014**.

[14] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, **2014**.

[15] C. Dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78. **2014**.

[16] J.P.C. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370, **2016**.

[17] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, **2016**.

[18] J. Yang, S. Liang, and Y. Zhang. Design challenges and misconceptions in neural sequence labeling. *arXiv preprint arXiv:1806.04470*, **2018**.

[19] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, **2014**.

[20] M. Luong, H. Pham, and C.D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, **2015**.

[21] B. Li, T. Liu, Z. Zhao, and X. Du. Attention-based recurrent neural network for sequence labeling. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 340–348. Springer, **2018**.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, **2015**.

[23] C. Raffel and D.P.W. Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*, **2015**.

[24] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212. **2016**.

[25] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, **2016**.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. **2017**.

[27] G.A. Şeker and G. Eryiğit. Initial explorations on using crfs for turkish named entity recognition. *Proceedings of COLING 2012*, pages 2459–2474, **2012**.

[28] C.N. Dos Santos and V. Guimaraes. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*, **2015**.

[29] M. Labeau, K. Löser, and A. Allauzen. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237. **2015**.

[30] Z. Yang, R. Salakhutdinov, and W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, **2016**.

[31] J. Xie, Z. Yang, G. Neubig, N. A Smith, and J. Carbonell. Neural cross-lingual named entity recognition with minimal resources. *arXiv preprint arXiv:1808.09861*, **2018**.

[32] O. Güngör, S. Üsküdarlı, and T. Güngör. Recurrent neural networks for turkish named entity recognition. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, **2018**.

[33] Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, **2015**.

[34] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, **2016**.

[35] Y. Zhang, H. Chen, Y. Zhao, Q. Liu, and D. Yin. Learning tag dependencies for sequence tagging. In *IJCAI*, pages 4581–4587. **2018**.

[36] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. **2016**.

[37] G. Tür, D. Hakkani-Tür, and K. Oflazer. A statistical information extraction system for turkish. *Natural Language Engineering*, 9(2):181–210, **2003**.

[38] A. Güneş and A.C. TantuĞ. Turkish named entity recognition with deep learning. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, **2018**.

[39] O. Güngör, T. Güngör, and S. Üsküdarli. The effect of morphology in named entity recognition with sequence tagging. *Natural Language Engineering*, 25(1):147–169, **2019**.

[40] T. Dincer, B. Karaoglan, and T. Kisla. A suffix based part-of-speech tagger for turkish. In *Fifth International Conference on Information Technology: New Generations (itng 2008)*, pages 680–685. IEEE, **2008**.

[41] C.A. Bahcevan, E. Kutlu, and T. Yildiz. Deep neural network architecture for part-of-speech tagging for turkish language. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pages 235–238. IEEE, **2018**.

[42] O. Güngör and E. Yıldız. Linguistic features in turkish word representations. In *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, **2017**.

[43] A. Üstün, M. Kurfalı, and B. Can. Characters or morphemes: How to represent words? In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 144–153. **2018**.

[44] R. Yeniterzi. Exploiting morphology in turkish named entity recognition system. In *Proceedings of the ACL 2011 Student Session*, pages 105–110. Association for Computational Linguistics, **2011**.

[45] F. Chollet. Keras. `https://github.com/keras-team/keras`, **2015**.

[46] A.A. Akin. zemberek-nlp. `https://github.com/ahmetaa/zemberek-nlp`. Erişim tarihi: 2019-06-08.

[47] H. Demir and A. Özgür. Improving named entity recognition for morphologically rich languages using word embeddings. In *2014 13th International Conference on Machine Learning and Applications*, pages 117–122. IEEE, **2014**.

[48] O. Kuru, O.A. Can, and D. Yuret. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921. **2016**.

[49] B. Can, A. Üstün, and M. Kurfalı. Turkish pos tagging by reducing sparsity with morpheme tags in small datasets. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 320–331. Springer, **2016**.

[50] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, **2018**.

[51] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, **2018**.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı :	YASİN EŞREF
Doğum Yeri :	Denizli,Türkiye
Medeni Hali :	Evli
E-posta :	yasinesref@gmail.com
Adres :	Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi
	Beytepe-ANKARA

Eğitim

Lisans :	Bilgisayar Mühendisliği Böl., Hacettepe Üniversitesi, Türkiye, 2011
Master :	Bilgisayar Mühendisliği Böl., Hacettepe Üniversitesi, Türkiye, 2019

Yabancı Dil

İngilizce

İş Tecrübesi

Tübitak-BİLGEM-YTE (Ağustos,2011 - Halen)

Yazılım ve Veritabanı Uzmanı

Deneyim Alanları

Makine Öğrenmesi

Derin Öğrenme

Doğal Dil İşleme

Yayınlar

"Morfem Düzeyinde Dikkat Mekanizması Kullanarak Türkçe Dizi Etiketleme",
Sinyal İşleme ve İletişim Uygulamaları (SİU) Kurultayı, 2019,
Yasin Eşref, Burcu Can Buğlalılar