DERİN ÖĞRENME KULLANARAK TÜRKÇE İÇİN BİR KELİME ANLAMI BELİRGİNLEŞTİRME UYGULAMASI

Tunç GÜMÜŞDAĞ
191402209

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans
Danışman: Dr. Öğr. Üyesi Volkan TUNALI

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Ocak, 2023

DERİN ÖĞRENME KULLANARAK TÜRKÇE İÇİN BİR KELİME ANLAMI BELİRGİNLEŞTİRME UYGULAMASI

Tunç GÜMÜŞDAĞ
191402209
ORCID: 0000-0002-9780-806X

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans
Danışman: Dr. Öğr. Üyesi Volkan TUNALI

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Ocak, 2023

JÜRİ VE ENSTİTÜ ONAYI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" ile bildirilen 6698 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

ETİK İLKE VE KURALLARA UYUM BEYANI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" ile bildirilen 6698 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

TEŞEKKÜR

Bu yüksek lisans tezini yazmama olanak sağlayan, destek olan ve yönlendiren değerli hocalarım Dr. Öğr. Üyesi Volkan Tunalı ve Dr. Öğr. Üyesi Mehmet Ali Aksoy Tüysüz olmak üzere, bu süreçte yanımda olan Bilgisayar Mühendisliği bölümü hocalarına ve maddi ve manevi olarak katkı sağlayan ve sağlamaya devam eden aileme teşekkür ederim.

Tunç GÜMÜŞDAĞ

Ocak, 2023

ÖZET

DERİN ÖĞRENME KULLANARAK TÜRKÇE İÇİN BİR KELİME ANLAMI BELİRGİNLEŞTİRME UYGULAMASI

Tunç Gümüşdağ
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Volkan Tunalı
Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü, 2023

Doğal dilleri anlamaya yönelik yapılan ilk sembolik yaklaşımlar insanlar tarafından yazılan kuralları kullanarak metnin anlamını yakalamaya çalışmaktadır. Ancak bu tür kural tabanlı sistemlerin kırılgan oldukları, tasarlandıkları belirli alanlarda sınırlı oldukları görülmüştür. Son 20 yılda, doğal dil işlemede istatiksel bir yaklaşım yaygınlaşmıştır. Bu sistem otomatik olarak matematiksel modeller kullanarak verilerden kurallar öğrenebilmektedir.

Bu tezde kelime anlamı belirginleştirme görevinde kullanılabilecek derin öğrenme çalışmaları ve algoritmaları incelenmiş ve bir Türkçe işaretlenmiş veri seti üzerinde deneyler yapılmıştır. Önceki birkaç çalışma ile karşılaştırma yapabilmek için SemEval-2007 çalıştayındaki Türkçe çalışmada kullanılan veriler kullanılmıştır.

Bu tez çalışması beş bölümden oluşmaktadır. İlk bölümde kelime anlamı belirginleştirmenin kısa bir özeti verilmektedir. Ardından tez kapsamında çözülmek istenilen problem, bu problemi çözmemizdeki amaç, problem hakkında yapılan varsayımlar ve sınırlıklar verilmektedir. İkinci bölümde ise Türkçe ve yabancı dillerdeki kelime anlamı belirginleştirme çalışmaları ile ilgili literatür çalışmalarına yer verilmektedir. Üçüncü bölümde, ilk olarak bu çalışmada kullanılan veya bu çalışmayla ilgili olan tanımlara yer verilmektedir. Devamında ise verilerin toplanmasına, analizine ve yorumlanmasına yer verilmektedir. Verilerin toplanması üzerine olan bölümde SemEval-2007 Türkçe çalışmasında kullanılan kelimelerin ve özelliklerin detaylarına yer verilmektedir. Verilerin çözümlenmesi ve yorumlanması kısmında ise ilk önce çalışmada

kullanılan kelimeler ve özelliklerin nasıl bir yapıda kullanıldığı ve geçirildikleri ön işlem aşamaları anlatılmaktadır. Ardından bu verilerin girdi olarak verileceği yapay sinir ağı modelleri ve bu modeller ile yapılan eğitim aşamaları ve bu aşamaların sonuçların analizi üzerinde detaylı anlatım yapılmaktadır. Son olarak, bu çalışmada sonuçlarının analizinde kullanılacak metrikler ve detayları anlatılmıştır. Dördüncü bölümde ise derin öğrenme modelleri kullanılarak alınan sonuçların ağırlıklı ve makro ortalamaları ile SemEval-2007 Türkçe verileriyle önceki çalışmaların elde ettiği metrik değerlerine yer verilmiş olup ardından bu bulgular yorumlanmıştır. Beşinci bölümde ise yapılan bu çalışma genel hatlarıyla özetlenmiş ve ileride yapılabilecek çalışmalar için öneriler verilmiştir.

Anahtar Sözcükler: Derin Öğrenme, Doğal Dil İşleme, Kelime Anlamı Belirginleştirme, SemEval-2007

ABSTRACT

A WORD SENSE DISAMBIGUATION APPLICATION ON TURKISH LANGUAGE USING DEEP LEARNING

Tunç Gümüşdağ
Master Thesis
Department of Computer Engineering
Computer Engineering Programme with Thesis
Thesis Advisor: Assist. Prof. Dr. Volkan Tunalı
Maltepe University Graduate School, 2023

Early symbolic approaches to understanding natural languages attempted to capture the meaning of text using rules is written by humans. However, such rule-based systems have proved to be fragile and limited in the specific areas for which they were designed. In the last 20 years, a statistical approach to natural language processing has become widespread. This system can automatically learn rules from data using mathematical models.

In this thesis, deep learning studies and algorithms that can be used in word sense disambiguation task are investigated and experiments are conducted on a labeled Turkish dataset. Data from the Turkish study in SemEval-2007 is used in order to make comparisons with several previous studies.

This thesis consists of five chapters. In the first chapter, a summary of word sense disambiguation is given. Then, the problem to be solved within the scope of the thesis, the purpose of solving this problem, the assumptions made about the problem and the limitations are given. In the second chapter, literature studies on word sense disambiguation studies in Turkish and foreign languages are given. In the third section, firstly, the definitions used in this study or related to this study are given. Afterwards, data collection, analysis and interpretation are given. In the section on data collection, details of the words and features used in the SemEval-2007 Turkish study are given. In the data analysis and interpretation section, firstly, the structure of the words and features used in the study and the pre-processing stages they have undergone are explained. Then,

the artificial neural network models to which these data will be given as input, the training stages with these models and the analysis of the results of these stages are explained in detail. Finally, the metrics to be used in the analysis of the results of this study and their details are explained. In the fourth section, the weighted and macro averages of the results obtained using deep learning models and the metric values obtained by previous studies with the SemEval-2007 Turkish data are given and then these findings are interpreted. In the fifth section, this study is summarized in general terms and suggestions for future studies are given.

Keywords: Deep Learning, Natural Language Processing, Word Sense Disambiguation, BERT, SemEval-2007

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

.

KISALTMALAR

ANN : Artificial Neural Network (Yapay Sinir Ağı)

BERT: Bidirectional Encoder Representations from Transformers (Transformatörlerden Çift Yönlü Kodlayıcı Temsilleri)

BiLSTM : Bidirectional Long Short-Term Memory (Çift Yönlü Uzun Kısa Vadeli Bellek)

BiGRU : Bidirectional Gated Recurrent Network (Çift Yönlü Geçitli Yineleme Birimi)

CEC : Constant Error Carousel (Sabit Hata Karuseli)

CG : Coarse-Grained (Kaba Ayrım)

CNN : Convolutional Neural Network (Evrişimli Sinir Ağı)

FG : Fine-Grained (İnce Ayrım)

FNN : Feedforward Neural Network (İleri Beslemeli Sinir Ağı)

GloVe : Global Vectors (Global Vektörler)

GRU : Gated Recurrent Unit (Geçitli Yineleme Birimi)

LSTM : Long Short-Term Memory (Uzun Kısa Vadeli Bellek)

MLP : Multilayer Perceptron (Çok Katmanlı Algılayıcı)

NLP : Natural Language Processing (Doğal Dil İşleme)

RNN : Recurrent Neural Network (Özyinelemeli Sinir Ağı)

RvNN : Recursive Neural Network (Tekrarlayan Sinir Ağı)

WEKA : Waikato Environment for Knowledge Analysis (Bilgi Analizi

için Waikato Ortamı)

WSD **:** Word Sense Disambiguation (Kelime Anlamı Belirginleştirme)

1. GİRİŞ

Dil bilimleri günlük hayatta yapılan sohbetlerde, karşılaşılan metinlerde ve diğer medyalarda yapılan çeşitli dil bilimsel gözlemleri açıklamayı ve nitelendirmeyi amaçlamakta olup doğası gereği dili edinmek, üretmek ve anlamak gibi insanların bilişsel algılarıyla alakalıdır. Dildeki dilbilimsel yapıları anlamak bu sürecin bir parçasıdır, ve bu aşamayı ele almak için dilsel ifadelerin yapılandırılmasında belli kurallar olduğu öne sürülmüştür (Manning ve Schütze, 1999). 1960'lı yılların başında bu tür sembolik yaklaşımlar ile doğal dil işlemede yapay zekâ tabanlı yöntemler kullanılmaya başlamıştır. 1980'li yıllarda ise bilgisayarlar ve elektronik kaynaklardaki gelişmeler ile istatistiksel yöntemler de kullanılmaya başlanmıştır (Orhan, 2006).

Doğal dil işleme kapsamındaki ana konular konuşma sentezi, konuşma tanıma, doğal dil anlama, doğal dil üretme ve makine çevirisi başlıkları altında özetlenebilir. Doğal dil işleme uygulamaları ise ana ve ara uygulamalar olarak ikiye ayrılabilmektedir. Ana uygulamalar makine öğrenmesi, otomatik özetleme ve bilgi çıkarımı gibi bütün olarak bir işlemi yapan sistemlerdir. Ara uygulamaların ise tümceyi öğelerine ayırma, çözümleme, biçimbilimsel analiz ve kelime anlamı belirginleştirme gibi ana uygulamalara yarar sağlayan belli görevleri vardır (Orhan, 2006).

Bu ara uygulamalardan kelime anlamı belirginleştirme, 1950'li yıllarda ilgi çekmeye başlayan bir doğal dil problemidir. Bu uygulama, belirli bir bağlamdaki çok anlamlı bir kelimenin ima edilen anlamını tanımlamaya çalışmaktadır (Arukgoda, Bandara, Bashani, Gamage ve Wimalasuriya, 2014).

Kelime anlamı belirginleştirme uygulamalarında karşılaşılan birçok problem vardır:

- Kelimeler cümle içinde kullanıldığı noktadan bilgi edindiği için incelenmesi gereken pencere belirsiz olabilmektedir.
- Hedef kelimenin kaç farklı anlamı olabileceğini saptamak oldukça zor olabilmektedir. Kelimenin anlam sayısı uygulamadan uygulamaya veya kişiden kişiye farklılık gösterebilmektedir.

- Yapılan farklı çalışmaların birbiriyle karşılaştırılması kolay olmayabilir. Kullanılan metinler çalışmadan çalışmaya farklılık gösterebilir. Kimi metinlerde çok özel ve belirli konular vardır ve kullanılan kelime anlamları sınırlı olabilmektedir (Orhan, 2006).

Türkçe dilinde kelime anlamı belirginleştirme üzerine yapılan çalışmalar var olan dilbilimsel kaynağın az olması gibi sebepler dolayısıyla göreceli olarak azdır. Farklı diller üzerinde yapılan uluslararası Senseval çalıştayının dördüncüsü olan SemEval çalışması 2007 yılında yapılmıştır. Bu çalıştayda sözcüksel örnekler alanında Türkçe bir çalışma da yer almıştır. Bu çalışmada veriler üzerinde makine öğrenmesi uygulamıştır, ve alınan sonuçlar çalıştaydaki diğer sonuçlar ile karşılaştırılmıştır (Tüysüz ve Güvenoğlu, 2014). Gelecek yıllarda SemEval'de kullanılan Türkçe veriler kullanılarak *Waikato Environment for Knowledge Analysis* (WEKA) [Bilgi Analizi için Waikato Ortamı] yazılımıyla farklı makine öğrenme algoritmaları da denenmiştir.

Kelime anlamı belirginleştirmede geleneksel makine öğrenmesi yöntemlerine kıyasla son yıllarda popülerleşen derin öğrenme algoritmaları dilsel iletişim ve bilgisayarlı görü alanlarında büyük başarı göstermiştir (S ve Raju, 2022).

1.1 Problem

Kelime anlamı belirsizliği problemine örnek olarak: *Geldi*, *Göze geldi* ve *Göz göze geldi* cümlelerinde geçen *geldi* kelimesi, her bir cümlede farklı anlamlarda kullanılmıştır. Yalnızca *geldi* kelimesini duyan veya okuyan bir insan için bu kelime genellikle *birinin veya bir şeyin bir yere ulaştığı* anlamına gelmektedir. Ancak *geldi* kelimesi *göze geldi* şeklinde kullanıldığında *başa kötü bir şeyin gelmesi, nazar değmesi* olarak yorumlanmaktadır ve son olarak *göz göze geldi* cümlesinde ise *geldi* kelimesinin *biriyle karşılaşma* anlamında kullanıldığı anlaşılacaktır (Orhan, 2006).

Bu çalışmanın üzerinde duracağı problem, *geldi* kelimesi gibi bir kelimenin bağlama göre hangi anlama geldiğini makinenin anlamasını sağlamaktır.

1.2 Amaç

Bu tez çalışması ile:

- Türkçe dilinde kelimelerin cümle içindeki anlamını belirginleştirmede derin öğrenme algoritmalarının etkililiğini deneysel olarak gözlemlemek,
- Bu tezde bahsi geçen SemEval-2007'de yapılan Türkçe çalışma ve bu Türkçe çalışma baz alınarak yapılan makine öğrenmesi çalışmalarını derin öğrenme algoritmalarıyla alınan sonuçlar ile karşılaştırarak analiz etmek,
- Günümüzde popülerleşen ve başarı gösteren derin öğrenme konusu üzerine bir çalışma yaparak literatüre katkıda bulunmak istenmiştir.

1.3 Varsayımlar

Kullanılan Türkçe verilerde eksiklikler, hatalı yazımlar olmadığı veya kullanılan algoritmaların cümleler üzerinde yaptığı ön işleme adımında kelimelerin anlamlarını yitirmediği varsayılmaktadır.

1.4 Sınırlıklar

Derin öğrenme algoritmaları daha fazla veri ile daha başarılı olabildiği için kullanılan örnek cümle sayısı ve kelime hazinesinin büyüklüğü sonuçlarda alınacak başarıyı etkilemektedir.

Veri miktarının az olduğu durumlarda, kullanılan kelimelerin eğitimde sık olarak kullanılmaması ve kelimenin çok kısa veya çok uzun olması, bu kelimenin kullanılan derin öğrenme algoritmalarında aldığı sonuçları olumsuz etkilediği için önem arz etmektedir.

2. LİTERATÜR ARAŞTIRMASI

2.1 Türkçe Kelime Anlamı Belirginleştirme

Bu bölümde kelime anlamı belirginleştirme alanında Türkçe dili üzerinde yapılmış çalışmalara yer verilmektedir.

Orhan'ın (2006) yaptığı çalışmada Türkçe anlam belirsizliği olan cümlelerde kelime anlamı belirginleştirme için kullanılabilecek kullanımı kolay algoritmalar ve özellikler üzerinde araştırma yapılmaktadır. Anlam belirsizliği olabilecek cümleler, anlam sınıfları ve algoritmalarda kullanılacak elle işaretleme önceden yapılmıştır. Aynı zamanda, bu hususla yakından ilişkili olan Senseval projesi de incelenmiştir (Orhan, 2006).

Orhan, Çelik ve Neslihan (2007), SemEval-2007 çalıştayında Türkçe dili üzerinde bir çalışma yapmışlardır. Bu çalışmada semantikler incelenmiş, Türkçe dilinde yapılacak kelime anlamı belirginleştirme uygulamaları için etiketlenmiş test ve eğitim veri setleri çıkarılmıştır. İstatiksel sonuç sunan basit bir sistem kullanılmış olup tam anlamı bulan *fine-grained* (FG) yöntemi ve alternatifleri içeren bir seti de kontrol eden *coarse-grained* (CG) yöntemi olarak iki ayrı kelime anlamı tespiti yaklaşımı kullanılmıştır. Değerlendirme yapılırken FG anlamları doğru sonuç elde edildiğinde bir puan alırken, CG anlamları her doğru olabilecek alternatif için puan almıştır. Alınan sonuçlarda FG için 0,13 duyarlılık (*precision*) ve 0,46 geri çağırım (*recall*) skoru; CG için 0,59 duyarlılık ve 0,46 geri çağırım skoru alınmıştır (Orhan vd., 2007).

Tüysüz ve Güvenoğlu (2014), SemEval çalıştayındaki Türkçe kelimelere makine öğrenmesi uygulandığında daha iyi sonuçlar alınıp alınamayacağını incelemiştir. Bu çalışmada Naive Bayes ve Karar Ağacı makine öğrenme algoritmaları ile WEKA yazılımı kullanılarak makine eğitilmiş ve alınan sonuçların SemEval çalıştayından daha iyi sonuç verip vermediği incelenmiştir. Bu çalışmada SemEval çalıştayında kullanılan FG değerleri ince ayrım olarak, CG değerleri ise kaba ayrım şeklinde belirtilmektedir. Çalışmada sadece ince ayrım değerleri incelenmiş olup çalışma sonucunda Naive Bayes algoritmasından ortalama 0,546 duyarlılık ve 0,584 geri çağırım skoru elde olunurken,

Karar Ağacı algoritması kullanıldığında 0,484 duyarlılık ve 0,577 geri çağırım skoru alınmaktadır (Tüysüz ve Güvenoğlu, 2014).

Aktaş'ın (2021) çalışmasında, Tüysüz ve Güvenoğlu'nun (2014) yaptığı çalışma üzerine olup, çalışmada SemEval çalıştayındaki Türkçe kelimeleri farklı makine öğrenmesi algoritmaları uygulanmasıyla ince ayrım tespit etmede daha iyi sonuçlar alınabilip alınamayacağı incelenmiştir. Yaptığı çalışmada Aktaş makineyi SemEval verileri ile K En Yakın Komşuluk, Destekçi Vektör Makineleri ve Rastgele Orman (*Random Forest*) makine öğrenme algoritmaları ile WEKA yazılımı kullanarak eğitmiştir. Çalışma sonucunda K En Yakın Komşuluk algoritması ortalama 0,369 duyarlılık ve 0,770 geri çağırım skoru vermiş, Destekçi Vektör Makineleri algoritması 0,133 duyarlılık ve 0,490 geri çağırım skoru sunmuş, Rastgele Orman algoritması ise 0,426 duyarlılık ve 0,990 geri çağırım skoru ile sonuçlanmıştır (Aktaş, 2021).

2.2 Yabancı Dillerde Kelime Anlamı Belirginleştirme

Türkçe dili dışındaki dillerde de kelime anlamı belirsizliği önemli bir problem olup literatürde bu diller için yapılmış çeşitli çalışmalara rastlanmaktadır.

Sun, Lv, Wang ve Wang (2017), Çince dilinde kelime anlamı belirginleştirme için yeni bir yöntem denemişlerdir: Çince'de de aynı şekilde yazılan kelimeler farklı anlamlara gelebildiği için yaptıkları çalışmada araştırmacıların hedef kelimenin gelebileceği anlamları alıp bunları eş anlamları ile değiştirdikleri görülmüştür. Örneğin, *zu* kelimesi hem "ölüm" hem de "asker" anlamına gelmektedir; makineye doğru anlamı öğretmek için *zu* kelimesi, ölüm kelimesi ile eşanlamlı *si wang* değiştirilirken, asker kelimesi eşanlamlısı olan *shibing* kelimesi ile değiştirilmiş, makine iki farklı anlam ile denenmiş ve hangi kelimenin doğru olduğunun bulunması sağlanmak istenmiştir. Yapılan deneyler sonucu, *Recurrent Neural Network* [Özyinelemeli Sinir Ağı] (RNN) modelinde 0,64 doğruluk (*accuracy*) oranı, *Gated Recurrent Unit* [Geçitli Yineleme Birimi] (GRU) modelinde 0,5 doğruluk oranı, ve *Long Short-Term Memory* [Uzun Kısa-Vadeli Bellek] (LSTM) bellek ağı modelinde 0,78 doğruluk oranı elde edilmiştir (Sun vd., 2017).

Gumizawa ve Yamamoto (2018), Japonca dilinde kelime anlamı belirginleştirme için Hiragana-Kanji dönüştürme yöntemi kullanmışlardır. Bu yöntem kelimeleri öznitelik

olarak seçmede kelime gömme (*word embedding*) ve noktasal karşılıklı bilgi (*point-wise mutual information* - PMI) katsayısı kullanmaktadır. Çalışmanın diğer kelime gömme metotlarından daha iyi doğruluk sonucu aldığı gözlemlenmiştir. Çalışmada Word2Vec kütüphanesinde *skip-gram* modeli ve hiyerarşik *softmax* fonksiyonu kullanılmıştır. Bu çalışmada eğitim ve test veri setleri olarak BCCWJ korporası ve Wikipedia kullanılmıştır. BCCWJ eğitim veri setiyle eğitildiğinde BCCWJ ve Wikipedia test veri setlerinde sırasıyla %93,37 ve %92,08 doğruluk oranı, Wikipedia ile eğitildiğinde ise sırasıyla aynı test veri setleriyle %90,40 ve %95,38 doğruluk oranı, hem Wikipedia hem de BCCWJ ile eğitildiğinde ise sırasıyla %92,91 ve %94,62 doğruluk oranı alındığı görülmüştür (Gumizawa ve Yamamoto, 2018).

Nithyanandan ve C (2019) yaptıkları çalışmada kelime anlamı belirginleştirmede çift yönlü modelleri tek yönlü modeller ile karşılaştırmışlardır. Karşılaştırmada LSTM ve GRU modellerini kullanmışlardır. Tensorflow ve Keras kütüphaneleri ile modelleri oluşturmuşlar, kelime gömme için ise GloVe modeli kullanmışlardır. Yapılan çalışma sonucunda BiLSTM + GloVe modeli 0,889 F1 skoru, BiGRU + GloVe modeli 0,874 F1 skoru, LSTM + GloVe modeli 0,839 F1 skoru, GRU + GloVe modeli ise 0,834 F1 skoru almıştır. Çalışma sonucunda çift yönlü modellerin tek yönlü modellerden daha isabetli sonuçlar aldığı, BiLSTM modelinin BiGRU modeline göre eğitim sırasında daha fazla vakit aldığı ve daha yüksek doğruluk oranına sahip olduğu gözlemlenmiştir (Nithyanandan ve C, 2019).

Singh ve Kumar (2020), Pencap dilinde kelime anlamı belirginleştirme için derin öğrenme teknikleri üzerinde çalışmalar yapmışlardır. Derin öğrenme modeli olarak çok katmanlı algılayıcı (*multilayer perceptron* - MLP) ve LSTM modelleri kullanılmıştır ve toplam 50 epok çalıştırılmıştır. Yapılan çalışma sonucu MLP modelinin doğruluk oranı ortalama %97,06, LSTM modelinin doğruluk oranı ise %91,7 olarak görülmüştür (Singh ve Kumar, 2020).

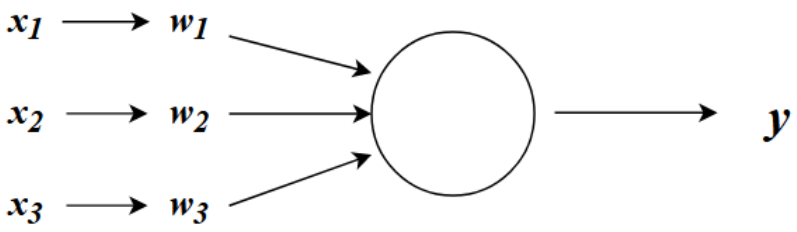
3. YÖNTEM

3.1 Derin Öğrenme

Derin öğrenme yapay zekâ alanının alt konularından biri olan makine öğrenmesinin özel bir halidir. Bu öğrenme çeşidi modeli verilen verilerden çıkarmaktadır. Derin öğrenme ile yapılan yapay sinir ağları yapı olarak giriş katmanı, en az iki gizli katman ve bir çıkış katmanından oluşmaktadır (Moposita, Trojman, Crupi, Lanuzza ve Vladimirescu, 2022).

3.2 Yapay Sinir Ağ Nöronları

Yapay sinir ağlarındaki her nöron, canlıların beynindeki nöronların çalışma yöntemini taklit etmektedir (Kim, 2017).



Şekil 1. Algılayıcı Nöron Modeli (Kim, 2017)

Örneğin, Şekil 1'deki nöron üç girdi almaktadır. Şekildeki x_1, x_2 *ve* x_3 değerleri girdi sinyallerini, ardından gelen w_1, w_2 *ve* w_3 değerleri ise sinaptik ağırlıkların sinyallerini belirtmektedir. Girdiler, ağırlıklar ve önyargı (bias) değeri kullanılarak ağırlıklı toplam hesaplanabilir, alınan sonuç belirli bir değerin altında veya üstünde olmasına göre sınıflandırılabilir (Kim, 2017).

$$\text{Ağırlıklı toplam} = w_1x_1 + w_2x_2 + w_3x_3 + b \tag{3.1}$$

3.2.1 Algılayıcı nöronlar

Algılayıcı nöronlar (p*erceptrons*) girdi olarak ikili kod değerleri almaktadır ve cevap olarak ikili kod değeri vermektedir (Nielsen, 2019).

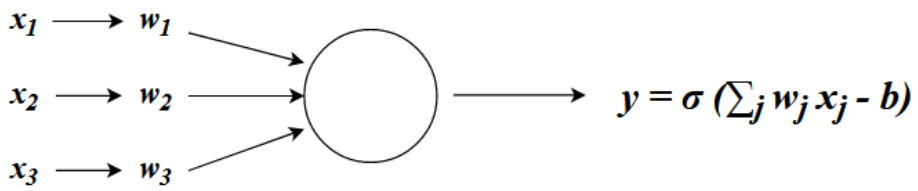
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Şekil 2. Algılayıcı Nöron Çıktı Hesaplaması (Nielsen, 2019)

Algılayıcı nöronların ateşlenme koşulları Şekil 2'deki gibidir. Bu koşullar için gerekli olan faktörler *x1, x2 ve x3* olarak düşünüldüğünde, bu girdilerin çıktı için önemini ifade eden gerçek sayılar ise *w1, w2 ve w3* ağırlıklarıyla belirlenebilir. Algılayıcı nöronlarda çıktı hesaplaması, ağırlıklı toplamın sonucuna bağlı olduğundan, en yüksek ağırlığı olan veya sınır değerinden (threshold) daha büyük bir ağırlık değeri olan bir faktör, nöronun ateşlenmesinde daha büyük bir rol oynamaktadır. Örneğin, bir algılayıcı nöronun ağırlıklarını *w1* = 5, *w2* = 1 ve *w3* = 1 ve sınır değeri 4 olarak varsayıldığında, *x1* faktörü sağlanmadığında bu nöronun ateşlenmesi imkânsız olarak kabul edilebilir. Başka bir örnekte *x1* faktörüne "havanın güneşli olması" denildiğinde, bu nöron havanın güneşli olmadığı senaryolarda tetiklenmeyecektir: Çünkü *w1* değeri sınır değerinden daha yüksek ağırlığa sahip olacaktır; bu da nöronun tetiklenmesi için gerekli bir koşuldur (Nielsen, 2019).

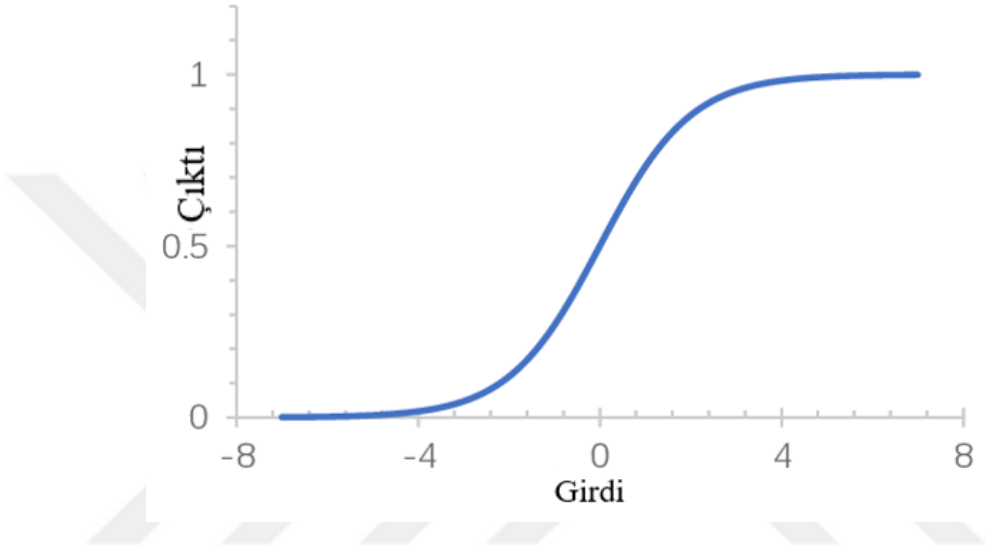
3.2.2 Sigmoid nöronlar

Şekil 3'te de görüldüğü gibi sigmoid nöronlar, algılayıcı nöronlar ile benzer yapıya sahiptir (Nielsen, 2019).



Şekil 3. Sigmoid Nöron Modeli (Nielsen, 2019)

Sigmoid nöronlar ağırlık değeri alırken algılayıcı nöronların aksine sigmoid fonksiyonunu kullanır. Bu sebeple sigmoid nöronlarda çıktı sadece 0 veya 1 ile kısıtlanmamakta, 0 ile 1 arasında herhangi bir sayı da olabilmektedir. Alınan ağırlıklı ortalama ve önyargı değerinin sigmoid fonksiyonu sonucu $\sigma\ (\Sigma_j\ w_j\ x_j\ -\ b)$ denklemi kullanılarak hesaplanmaktadır (Nielsen, 2019). Sigmoid fonksiyonu çizildiğinde Şekil 4'teki gibi görünmektedir.



Şekil 4. Sigmoid Fonksiyonu (Xing ve Wu, 2020)

3.3 Yapay Sinir Ağı Modeli

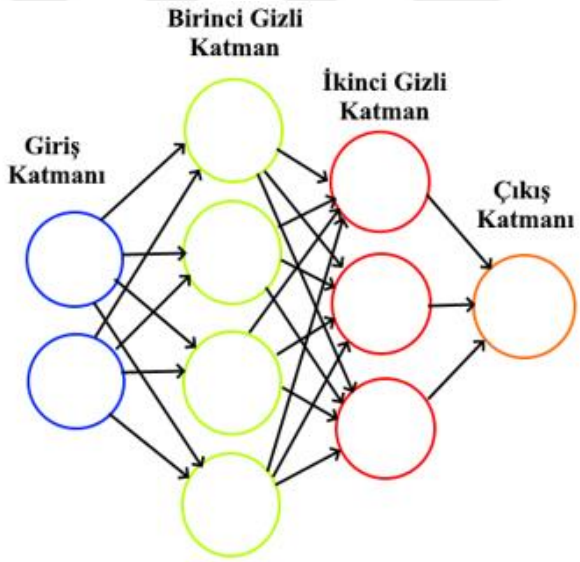
Yapay sinir ağı (*artificial neural network* - ANN) modelleri, yapay sinir ağ nöronlarından oluşan bir ağdır. Bu nöronlar birçok farklı şekilde bağlanabilir ve farklı tipte sinir ağları elde edilebilir. Bu yapılardaki nöron gruplarına katman denmektedir. En sık kullanılan yapılarda giriş katmanı, gizli katmanlar ve çıkış katmanı bulunmaktadır. Giriş katmanları sadece giriş sinyallerini iletmekte kullanılmaktadır. Bu katmanlarda ağırlıklı toplam veya aktivasyon fonksiyonu hesaplaması yapılamamaktadır. Çıkış katmanları sinir ağ modellerinin sonucunu içeren katmanlardır. Giriş ve çıkış katmanları arasındaki katmanlara sinir ağının dışından erişilemedikleri için gizli katmanlar denmektedir (Kim, 2017).

Yapay sinir ağı modelleri zamanla basit yapılardan daha kompleks yapılara dönüşmeye başlamıştır. İlk başlarda sinir ağları sadece bir giriş ve çıkış katmanından oluşmuştur, ve

bunlar tek katmanlı sinir ağları olarak adlandırılmıştır: Bu sinir ağlarına gizli katman eklendiğinde çok katmanlı sinir ağları oluşmaktadır. Dolayısıyla, çok katmanlı sinir ağları giriş katmanı, bir veya birden fazla gizli katman ve çıkış katmanından oluşmaktadır. Sadece bir gizli katmana sahip olan ağlara sığ sinir ağları, birden fazla gizli katman içeren ağlara ise derin sinir ağları denmektedir. Pratikte kullanılan modern sinir ağlarının çoğu derin sinir ağları kullanmaktadır (Kim, 2017).

3.3.1 Çok katmanlı algılayıcı

Çok katmanlı algılayıcılar (*multilayer perceptrons* - MLP), MLP algoritması olarak da bilinen bir ileri beslemeli yapay sinir ağıdır. MLP, aktivasyon fonksiyonlarını kullanarak lineer olmayan tahminler yapmaktadır. Yapı olarak giriş katmanı, gizli katmanlar ve çıkış katmanından oluşmaktadır (Sridevi, Kularni ve Maria, 2019). Bu yapıya örnek bir grafik Şekil 5'te verilmektedir.

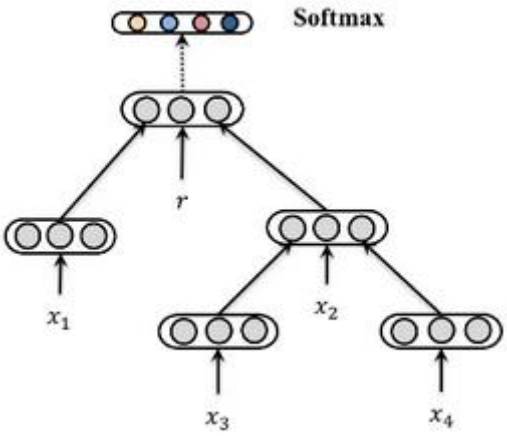


Şekil 5. İki Gizli Katmanı ve Bir Çıkış Katmanı olan MLP Nöronları

3.3.2 Tekrarlayan sinir ağları

Tekrarlayan sinir ağları (*recursive neural network* - RvNN), ayrıştırma ağacı (*parse tree*) gibi yapılı girdileri kullanarak aşağıdan yukarıya (*bottom-up*) her ilişkiyi hesaplamaktadır. Bu sinir ağı türü ayrıştırma, duygu analizi, yazısal gerektirme (*textual entailment*), segmentasyon ve eşanlatım tespiti (*paraphrase identification*) gibi birçok

çalışmaya katkı sağlamıştır (Li, Li ve Hovy, 2014). Bu sinir ağı modeline örnek bir grafik Şekil 6'da verilmektedir.



Şekil 6. Aşağıdan Yukarıya Tekrarlayan Sinir Ağı Modeli (Ma, Gao, Joty ve Wong, 2022)

3.3.3 Özyinelemeli sinir ağları

Özyinelemeli sinir ağlarının (*recurrent neural network* - RNN) döngüsel bağlantılar kurması, bu ağın yapılarını dizi işlemede güçlü kılmaktadır. Derin sinir ağları sadece belli bir pencere için geçici modelleme yapmaktadır. Buna karşın RNN, önceki döngülerin aktivasyon değerlerini güncel adıma girdi olarak kullanarak yapılacak tahminleri etkileyebilmektedir (Sak, Senior ve Beaufays, 2014).

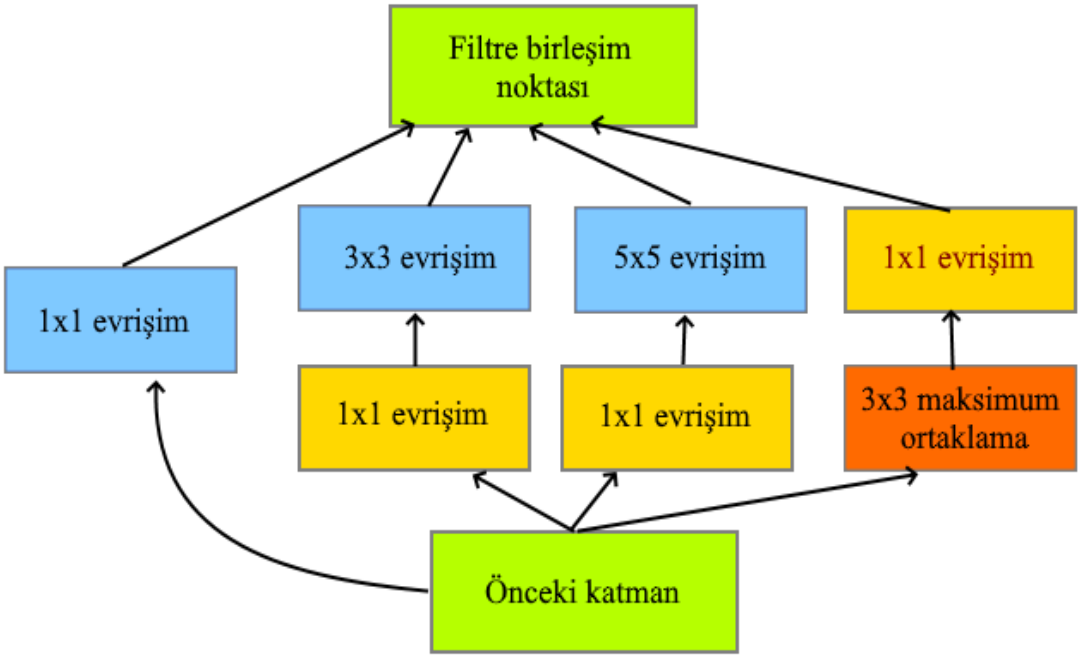
RNN'nin sınırlarından bir tanesi ise uzun ve karışık geçici dizilerde zorlanmasıdır (Williams ve Zipser, 1989). RNN'nin istenilen öğrenimi sağlaması için hangi eski girdileri tutması gerektiğini anlaması gerekmektedir: Ancak gradyan kullanan öğrenme metotları çok uzun zaman aldığı için (Hochreiter ve Schmidhuber, 1997), geriye yayılım (*back-propagation*) yöntemlerinde eskiye gönderilen hata sinyalleri kaybolmakta veya yok olmaya başlamaktadır (Hochreiter, 1998).

3.3.4 Evrişimli sinir ağları

Evrişimli sinir ağları (*Convolutional neural network* - CNN), MLP yapısının özel bir türüdür. Sınıflandırmanın yanı sıra nesne tanıma, takip etme, doğal dil işlemede anlam ayrıştırma, cümle modelleme ve tahmin etme gibi problemlerde de kullanılmıştır.

Bilgisayarlı görü uygulamalarında başarılı sonuçlar verdiği için CNN bu alanda daha çok kullanılmaktadır (Silver vd., 2016).

LeNet-5 çalışmasından sonra CNN mimarileri genel olarak standart bir yapıya sahip olmuşlardır. Katman olarak, üst üste konulmuş evrişimli katmanlar, gerekli durumlarda normalleştirme (*normalization*) ve havuzlandırma (*pooling*) katmanları, ve bir veya birden fazla tamamen bağlı (*fully-connected*) katmanlar kullanılmaktadır (Szegedy vd., 2015).



Şekil 7. GoogLe-Net Inception Modülü (Szegedy vd., 2015)

Şekil 7'deki GoogLe-Net Inception modülündeki 1x1 evrişim katmanları, asıl işlevlerinin yanı sıra modelin darboğaz yapmasını engellemekte de kullanılmaktadır (Szegedy vd., 2015).

3.3.5 Uzun kısa vadeli bellek ağları

Uzun kısa vadeli bellek ağları (*long short-term memory* - LSTM), giriş katmanı, unutma katmanı ve çıkış katmanı olarak üç katmandan oluşmaktadır: Bu mimariler klasik sinir ağlarından farklı olarak hem ileriye, hem de geriye yönelik besleme yapabilmektedir (Hochreiter ve Schmidhuber, 1997).

LSTM ağları, RNN mimarisindeki gradyan patlaması ve yok olması problemlerinden dolayı dizi işleme özelliği olan sinir ağ uygulamalarında hatalı sonuçlar verebildiği için bu probleme çözüm olarak geliştirilmiştir. LSTM, RNN'de gradyanların geriye yayılım hatalarını engellemek için sabit hata döngüsü (*constant error carousel* - CEC) denilen gradyan bazlı bir algoritma kullanmaktadır (Hochreiter ve Schmidhuber, 1997). Modern LSTM yapıları gözetleme bağlantıları da (*peephole*) içermektedir (Sak vd., 2014).

3.4 Doğal Dil İşleme

Doğal dil işlemenin (*natural language processing* - NLP) ana amacı dilleri inceleyip anlamaktır. NLP sistemleri bir yazının yapısını en az seviyede de olsa "Kim kime ne yaptı?" sorusunu yanıtlayabilecek kadar anlamalıdır. Bundan dolayı NLP araştırmalarında birçok görev (*task*) arasından birine odaklanıp cümlenin yapısının sadece bir kısmını anlama üzerine çalışma yapılmaktadır (Manning ve Schütze, 1999).

NLP alt başlıklar halinde birden fazla belirsizliği konu almaktadır. Örneğin: cümle parçacığı içinde kelime anlamı etiketleme (*part-of-speech tagging*; Manning ve Schütze, 1999), varlık ismi tanıma (*named entity recognition*; Borthwick, Sterling, Agichterin ve Grishman, 1998), dil modelleme (*language modelling*; Bengio, Ducharme, Vincent ve Jauvin, 2003) ve, bu tezin ana konusu olan, kelime anlamı belirginleştirme (*word sense disambiguation*) gibi alt konulara sahiptir (Manning ve Schütze, 1999).

3.4.1 Kelime anlamı belirginleştirme

Kelime anlamı belirginleştirme (*word sense disambiguation* - WSD), bir cümle içerisinde kullanılan bir kelimenin hangi anlamına göre kullanıldığına karar vermeye dayanmaktadır. Burada ele alınan problem birçok kelimenin birden fazla anlamı olabilmesinden kaynaklanır. Herhangi bir bağlam olmadığı durumlarda kelime anlamı belirsizliği oluşmaktadır (Manning ve Schütze, 1999).

Tablo 1. Yüz Kelimesinin Türk Dil Kurumu Sözlüğüne Göre İki Ayrı Anlamı (Türk Dil Kurumu, 2022).

Türkçe	İngilizce	Sözlük Tanımı
yüz 1	surface	Yüzey
yüz 2	face	Başta, alın, göz, burun, ağız, yanak ve çenenin bulunduğu ön bölüm, sima, çehre, surat

Tablo 2. Yüz Kelimesinin Türk Dil Kurumu Sözlüğünden İki Ayrı Örneği (Türk Dil Kurumu, 2022).

Türkçe	Örnek Cümle
yüz 1	Suyun yüzünde
yüz 2	Bir güzel çocuk yüzüyle gülümsüyor.

Örneğin Tablo 1'de yüz kelimesinin iki ayrı anlamı, İngilizce diline çevirisi ve bu iki anlamın sözlük tanımı yazılmıştır. Yüz kelimesinin birinci anlamı *surface* [yüzey], ikinci anlamı ise *face* [yüz] olarak çevrilmiştir. Çevirilerinin farklı olması iki ayrı cümle anlamı olduğunun göstergesidir. Tablo 2'deki iki örnekte de yüz kelimesinin anlamının belirtilmesi için bir doğal dil işleme yazılımı tarafından kelime anlamı belirsizliğinin ortadan kaldırılması gerekmektedir.

3.5 Veriler ve Toplanması

Bu çalışmada, SemEval-2007 Türkçe çalışmasında da kullanılan anlamları işaretlenmiş 10 isim, 10 fiil ve 6 diğer (zarf, sıfat vb.) olarak sınıflandırılan kelimeler kullanılmıştır.

İsim grubunda **ara, baş, el, göz, kız, ön, sıra, üst, yan** ve **yol** kelimeleri kullanılmış olup eğitim için ortalama 92 örnek, test için ortalama 31 örnek olarak toplamda ortalama 123 örnek cümle kullanılmıştır.

Fiil grubunda **al, bak, çalış, çık, geç, gel, gir, git, gör** ve **konuş** kelimeleri kullanılmıştır ve eğitim için ortalama 263,3 örnek, test için ortalama 99,8 örnek olarak tamamıyla ortalama 363,1 örnek cümle kullanılmıştır.

Diğer grubunda **büyük, doğru, küçük, öyle, son** ve **tek** kelimeleri kullanılmış olup eğitim için ortalama 66,7 örnek, test için ortalama 21,5 örnek olarak toplamda ortalama 88,2 örnek cümle kullanılmıştır.

SemEval-2007 Türkçe çalışmasındaki veri grupları ve özellikleri hakkında detaylı bilgiler Tablo 3 ile Tablo 6 arasında verilmektedir.

Tablo 3. SemEval-1 Turkish Lexical Task İsim Grubundaki Veriler (Orhan vd., 2007)

Kelimeler	**İngilizce Anlamları**	**Anlam**	**Eğitim örnek sayısı**	**Test örnek sayısı**
İsim				
ara	distance, break, interval	7	192	63
baş	head, leader, beginning, top	5	68	22
el	hand, stranger, country	3	113	38
göz	eye, glance, division	3	92	27
kız	girl, virgin, daughter, angry	2	96	21
ön	front, foreground, face	5	72	23
sıra	queue, order, turn, desk	7	85	28
üst	upper side, outside, clothing	7	69	23
yan	side, direction, askew, burn	5	65	31
yol	way, road, path, method	6	68	29

Tablo 4. SemEval-1 Turkish Lexical Task Fiil Grubundaki Veriler (Orhan vd., 2007)

Kelimeler	**İngilizce Anlamları**	**Anlam**	**Eğitim örnek sayısı**	**Test örnek sayısı**
Fiil				
al	take, get, red	24	963	125
bak	look, examine	4	207	85
çalış	work, study, start	4	103	61
çık	climb, leave, increase	6	138	87
geç	pass, happen, late	11	164	90
gel	come, arrive, fit, seem	20	346	215
gir	enter, fit begin, penetrate	6	163	84
git	go, leave, pass, last	13	214	120
gör	see, understand, consider	5	206	68
konuş	talk, speak	6	129	63

Tablo 5. SemEval-1 Turkish Lexical Task Diğer Grubundaki Veriler (Orhan vd., 2007)

Kelimeler	**İngilizce Anlamları**	**Anlam**	**Eğitim örnek sayısı**	**Test örnek sayısı**
Diğer				
büyük	big, extensive, important	6	97	26
doğru	straight, true, accurate	6	81	38
küçük	little, small, young, kid	4	45	14
öyle	such, so that	4	51	23
son	last, recent, final	2	86	18
tek	single, unique, alone	2	40	10

Tablo 6. Verilerdeki Kısaltmalar ve Özellikleri (Zeynep Aktaş, 2021)

Özellikler	**Kısaltmalar**
Dosya	Dosya
ÖnceKök	Ök
Tür	Tür
ÖnceTür	Öt
ÖnceTüretme	Ötr
ÖnceHalEki	Öhe
Öncеİyelik	Öiy
Öncеİlişki	Öiş
HedefKök	Hk
HedefTür	Ht
HedefTüretme	Htr
HedefHalEki	Hhe
Hedefİyelik	Hiy
Hedefİlişki	Hiş
SonraKök	Sk
SonraTür	St
SonraTüretme	Str
SonraHalEki	She
Sonraİyelik	Siy
Sonraİlişki	Siş

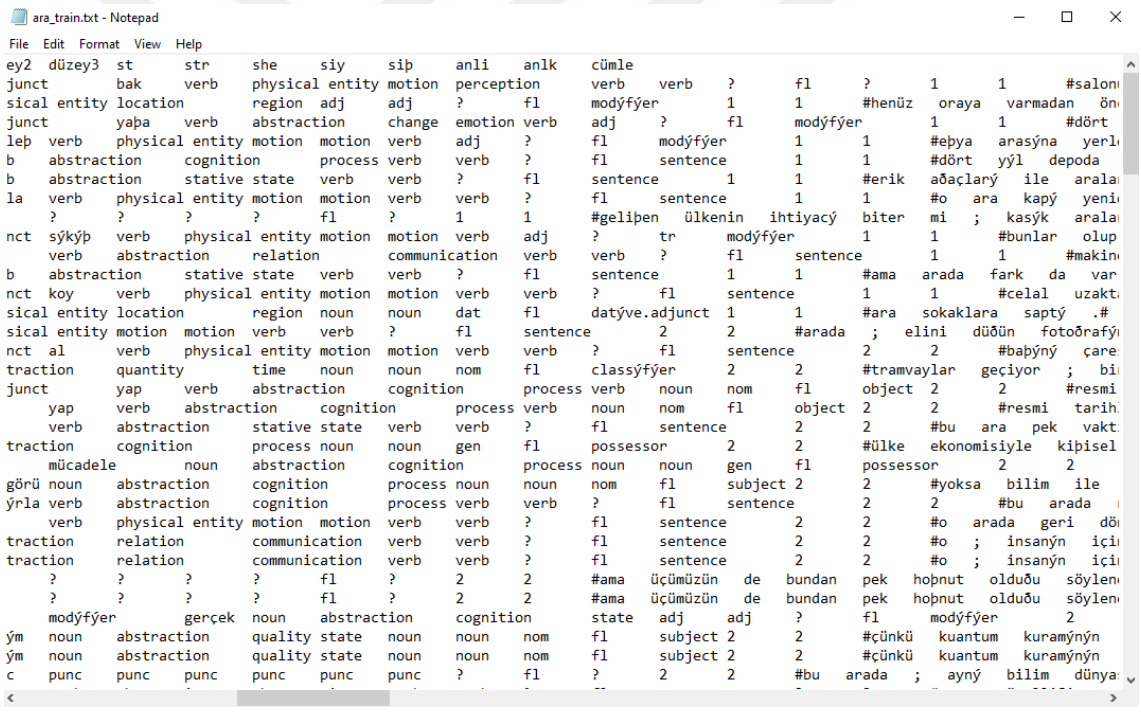
Bu çalışmada, SemEval-2007'deki Türkçe çalışmasında kullanılan veriler kullanılmıştır. Bu verilerin hazırlanmasında Türk Dil Kurumunun (TDK) oluşturduğu, internet üzerinden erişilebilen güncel Türkçe sözlük kullanılmıştır. Örnek cümleleri elde etmek

için ise Orta Doğu Teknik Üniversitesi (ODTÜ) derlemi ve Sabancı Ağaç Yapılı Derlemi kullanılmıştır (Zeynep Aktaş, 2021).

3.6 Verilerin Çözümlenmesi ve Yorumlanması

Bir kelime anlamı belirginleştirme uygulaması üzerinde çalışma yapmak için gereken birkaç hazırlık vardır. Türkçe veri, verinin sayısal değerlere dönüştürülmüş hali ya da dönüştürmeyi yapabilen bir algoritma, ve bu teze özel olarak derin öğrenme için kullanılabilecek uygulamalar ve algoritmalar gerekmektedir.

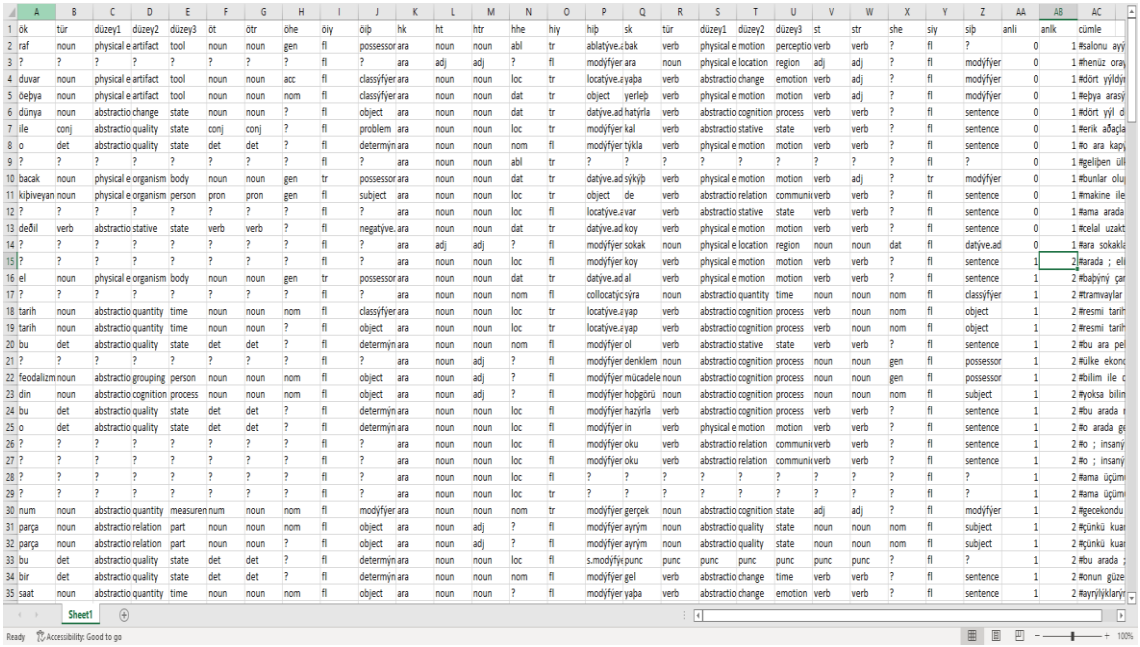
Şekil 8'de de görüldüğü gibi, SemEval-2007'den alınan Türkçe veriler txt formatındadır. Bu veriler, bu çalışmada kullanılan önişleme (*preprocessing*) yapısına uygun olması için Excel tablosuna kopyalanıp .xlsx uzantısına dönüştürülmüştür.



Şekil 8. SemEval-2007 Türkçe verilerdeki Ara Kelimesinin TXT Formatının Bir Kısmı

Önişleme aşamasında verilerin çözümlenmesi için SemEval-2007 Türkçe verilerindeki özellikler tek öznitelikli ve çok öznitelikli olarak iki farklı yapıya ayrılarak kullanılmıştır. Tek öznitelik kullanılan yapıda, Türkçe verilerinden sadece "cümle" sütunu alınmış, "anli" sütununu kategorize edip karşılaştırarak derin öğrenme yapılmıştır. Çok öznitelik kullanılan yapıda ise Türkçe verilerinden "dosya no", "cümle no", "geçiş no" ve "anlk"

sütunları çıkarılarak geri kalan bütün sütunlar öznitelik olarak kullanılmış ve “anli” sütunu kategorize edilerek karşılaştırılmıştır. Verilerdeki “dosya no”, “cümle no” ve “geçiş no” sütunları önceki çalışmalarda hedef kelimeyi bulmak için kullanılmıştır. Bu çalışmada hedef kelime verileri işlerken ayrıldığı için bu sütunların kullanılmasına gerek görülmemiştir. “anlk” sütunu ise birden çok hedef kelimeyi veya birden fazla anlama da gelebilen kelimeyi işaretlemek için kullanılmıştır. Bu çalışmada tek bir hedef kelime arandığı için bu sütuna ihtiyaç duyulmamıştır.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	ök	tür	düzey1	düzey2	düzey3	öt	ötr	öhe	öiy	öiþ	hk	ht	htr	hhe	hiy	hiþ	sk	tür	düzey1	düzey2	düzey3	st	str	she	siy	siþ	anli	anlk	cümle
2	raf	noun	physical e	artifact	tool	noun	noun	gen	fl	possessor	ara	noun	noun	abl	tr	ablatýve.a	bak	verb	physical e	motion	perceptio	verb	verb	?	fl	?	0	1	#salonu ayý
3	?	?	?	?	?	?	?	?	fl	?	ara	adj	adj	?	fl	modýfyer	ara	noun	physical e	location	region	adj	adj	?	fl	modýfyer	0	1	#henüz oray
4	duvar	noun	physical e	artifact	tool	noun	noun	acc	fl	classýfyer	ara	noun	noun	loc	tr	locatýve.a	yaþa	verb	abstractio	change	emotion	verb	adj	?	fl	modýfyer	0	1	#dört yýldý
5	öebya	noun	physical e	artifact	tool	noun	noun	nom	fl	classýfyer	ara	noun	noun	dat	tr	object	yerleþ	verb	physical e	motion	motion	verb	adj	?	fl	modýfyer	0	1	#ebya arasý
6	dünya	noun	abstractio	change	state	noun	noun	?	fl	object	ara	noun	noun	dat	tr	datýve.ad	hatýrla	verb	abstractio	cognition	process	verb	verb	?	fl	sentence	0	1	#erik ağaçla
7	ile	conj	abstractio	quality	state	conj	conj	?	fl	problem	ara	noun	noun	loc	tr	modýfyer	kal	verb	abstractio	stative	state	verb	verb	?	fl	sentence	0	1	#erik ağaçla
8	o	det	abstractio	quality	state	det	det	?	fl	determýn	ara	noun	noun	nom	fl	modýfyer	týkla	verb	physical e	motion	motion	verb	verb	?	fl	sentence	0	1	#o ara kapý
9	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	abl	tr	?	?	?	?	?	?	?	?	?	fl	?	0	1	#geliþen üll
10	bacak	noun	physical e	organism	body	noun	noun	gen	tr	possessor	ara	noun	noun	dat	tr	datýve.ad	sýkýþ	verb	physical e	motion	motion	verb	adj	?	tr	modýfyer	0	1	#bunlar olu
11	kibiveyan	noun	physical e	organism	person	pron	pron	gen	fl	subject	ara	noun	noun	loc	tr	object	de	verb	abstractio	relation	communi	verb	verb	?	fl	sentence	0	1	#makine ile
12	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	loc	fl	locatýve.a	var	verb	abstractio	stative	state	verb	verb	?	fl	sentence	0	1	#ama arada
13	deðil	verb	abstractio	stative	state	verb	verb	?	fl	negatýve.	ara	noun	noun	dat	tr	datýve.ad	koy	verb	physical e	motion	motion	verb	verb	?	fl	sentence	0	1	#celal uzakt
14	?	?	?	?	?	?	?	?	fl	?	ara	adj	adj	?	fl	modýfyer	sokak	noun	physical e	location	region	noun	noun	dat	fl	datýve.ad	0	1	#ara sokakl
15	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	loc	fl	modýfyer	koy	verb	physical e	motion	motion	verb	verb	?	fl	sentence	1	2	#arada ; el
16	el	noun	physical e	organism	body	noun	noun	gen	tr	possessor	ara	noun	noun	dat	tr	datýve.ad	al	verb	physical e	motion	motion	verb	verb	?	fl	sentence	1	2	#babýnýj çar
17	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	nom	fl	collocatýc	sýra	noun	abstractio	quantity	time	noun	noun	nom	fl	classýfyer	1	2	#tramvaylar
18	tarih	noun	abstractio	quantity	time	noun	noun	nom	fl	classýfyer	ara	noun	noun	loc	tr	locatýve.a	yap	verb	abstractio	cognition	process	verb	noun	nom	fl	object	1	2	#resmi tarih
19	tarih	noun	abstractio	quantity	time	noun	noun	?	fl	object	ara	noun	noun	loc	tr	locatýve.a	yap	verb	abstractio	cognition	process	verb	noun	nom	fl	object	1	2	#resmi tarih
20	bu	det	abstractio	quality	state	det	det	?	fl	determýn	ara	noun	noun	nom	fl	modýfyer	ol	verb	abstractio	stative	state	verb	verb	?	fl	sentence	1	2	#bu ara pel
21	?	?	?	?	?	?	?	?	fl	?	ara	noun	adj	?	fl	modýfyer	denklem	noun	abstractio	cognition	process	noun	noun	gen	fl	possessor	1	2	#ülke ekon
22	feodalizm	noun	abstractio	grouping	person	noun	noun	nom	fl	object	ara	noun	adj	?	fl	modýfyer	mücadele	noun	abstractio	cognition	process	noun	noun	gen	fl	possessor	1	2	#bilim ile c
23	din	noun	abstractio	cognition	process	noun	noun	nom	fl	object	ara	noun	adj	?	fl	modýfyer	hoþgörü	noun	abstractio	cognition	process	noun	noun	nom	fl	subject	1	2	#yoksa bilin
24	bu	det	abstractio	quality	state	det	det	?	fl	determýn	ara	noun	noun	loc	fl	modýfyer	hazýrla	verb	abstractio	cognition	process	verb	verb	?	fl	sentence	1	2	#bu arada
25	o	det	abstractio	quality	state	det	det	?	fl	determýn	ara	noun	noun	loc	fl	modýfyer	in	verb	physical e	motion	motion	verb	verb	?	fl	sentence	1	2	#o arada ge
26	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	loc	fl	modýfyer	oku	verb	abstractio	relation	communi	verb	verb	?	fl	sentence	1	2	#o ; insanýy
27	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	loc	fl	modýfyer	oku	verb	abstractio	relation	communi	verb	verb	?	fl	sentence	1	2	#o ; insanýy
28	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	loc	fl	?	?	?	?	?	?	?	?	?	fl	?	1	2	#ama üçüm
29	?	?	?	?	?	?	?	?	fl	?	ara	noun	noun	loc	tr	?	?	?	?	?	?	?	?	?	fl	?	1	2	#ama üçüm
30	num	noun	abstractio	quantity	measuren	num	noun	nom	fl	modýfyer	ara	noun	noun	nom	tr	modýfyer	gerçek	noun	abstractio	cognition	state	adj	adj	?	fl	modýfyer	1	2	#gecekondu
31	parça	noun	abstractio	relation	part	noun	noun	nom	fl	object	ara	noun	adj	?	fl	modýfyer	ayrým	noun	abstractio	quality	state	noun	noun	nom	fl	subject	1	2	#çünkü kuat
32	parça	noun	abstractio	relation	part	noun	noun	?	fl	object	ara	noun	adj	?	fl	modýfyer	ayrým	noun	abstractio	quality	state	noun	noun	nom	fl	subject	1	2	#çünkü kuat
33	bu	det	abstractio	quality	state	det	det	?	fl	determýn	ara	noun	noun	loc	fl	s.modýfýr	punc	punc	punc	punc	punc	punc	punc	?	fl	?	1	2	#bu arada ;
34	bir	det	abstractio	quality	state	det	det	?	fl	determýn	ara	noun	noun	nom	fl	modýfyer	gel	verb	abstractio	change	time	verb	verb	?	fl	sentence	1	2	#onun güze
35	saat	noun	abstractio	quantity	time	noun	noun	nom	fl	object	ara	noun	noun	?	fl	modýfyer	yaþa	verb	abstractio	change	emotion	verb	verb	?	fl	sentence	1	2	#ayrýlýklarýy

Şekil 9. Ara Kelimesinin Excel Formatına Aktarıldıktan Sonrası

Şekil 9’daki gibi Excel dosyasına kopyalanan bu ham verileri Jupyter uygulaması üzerinde koda geçirmek için pandas kütüphanesi kullanılmıştır. Geçirilen bu ham veriler işlenilmeden önce kelimelerdeki kodlayıcı (*encoding*) sorunları, noktalama işaretleri ve çalışmada kullanılmayan semboller kaldırılmıştır; ardından, alınan dosyadaki düzenlenmiş özellikler ile hedef anlamlar kaydedilmiştir. Bu işlemleri gerçekleştirmede kullanılan kod parçacığı Şekil 10’da gösterilmektedir.

```
In [8]: import math

ansi_encoding = ['ý','ð','þ']
utf8_encoding = ['ı','ğ','ş']
remove_elements = ['#',';','.',':']

def fixEncoding(list_input):
    read_sentences = list_input
    sentences = read_sentences
    for i, sentence in enumerate(read_sentences):
        new_sentence = sentence
        for x in range(len(ansi_encoding)):
            new_sentence = new_sentence.replace(ansi_encoding[x],utf8_encoding[x])
        for y in range(len(remove_elements)):
            new_sentence = new_sentence.replace(remove_elements[y],"")
        sentences[i] = " ".join(new_sentence.split())
    return sentences
```

Şekil 10. Önişleme Aşamasında Yapılan Veri Temizleme İşlemi

Doğal dil işlemede verileri makinenin işlemesi için kelimelerin makinenin anlayabileceği sayı değerlerine çevrilmesi gerekmektedir. Bunun için çevirme işlemlerinde BERT'in *Tokenizer* fonksiyonu veya Keras kütüphanesindeki *Text Vectorization* katmanı kullanılmıştır. İşlenilen Türkçe veriler *Tokenizer* fonksiyonu kullanılarak numaralandırıldıktan sonra Keras ve Tensorflow kütüphanelerindeki derin öğrenme algoritmalarıyla makinenin eğitimine başlanmıştır.

Derin öğrenme ile makineyi eğitirken, birçok farklı derin öğrenme metodu denenmiştir. Tunalı'nın (2022) çalışmasında kullandığı Baseline ANN, CNN, Convolutional BiLSTM ve BiLSTM modelleri ile Devlin ve diğerleri tarafınca (2019) yapılan BERT modelinin kullanılmasına karar verilmiştir. Eğitimin ardından modelde bulunan *evaluate* [değerlendirme] fonksiyonu ve *sklearn.metrics* kütüphanesinden *classification_report* sınıflandırma fonksiyonu kullanılarak doğruluk, geri çağırım, duyarlılık ve F1 değerleri incelenmiştir. Tahmin skorlarını hesaplamada kullanılan kod parçacıkları Şekil 11 ve Şekil 12'deki gibidir.

```
from sklearn.metrics import classification_report

predicted_raw = model.predict(train_inputs, batch_size=32)
predicted_raw[0]

y_predicted = np.argmax(predicted_raw, axis = 1)
y_true = train_targets

train_report = classification_report(y_true, y_predicted)

predicted_raw = model.predict(trial_inputs, batch_size=32)
predicted_raw[0]

y_predicted = np.argmax(predicted_raw, axis = 1)
y_true = trial_targets

trial_report = classification_report(y_true, y_predicted)

split_train = train_report.split()
split_trial = trial_report.split()
```

Şekil 11. Text Vectorization Modelleri için Tahmin Skorları Hesaplama

```
from sklearn.metrics import classification_report

predicted_raw = model.predict({'input_ids':x_train['input_ids'],'attention_mask':x_train['attention_mask']})
predicted_raw[0]

y_predicted = np.argmax(predicted_raw, axis = 1)
y_true = train_targets

train_report = classification_report(y_true, y_predicted)

predicted_raw = model.predict({'input_ids':x_test['input_ids'],'attention_mask':x_test['attention_mask']})
predicted_raw[0]

y_predicted = np.argmax(predicted_raw, axis = 1)
y_true = trial_targets

trial_report = classification_report(y_true, y_predicted)
```

Şekil 12. BERT Modelleri için Tahmin Skorları Hesaplama

Örnek olarak, ara kelimesi için kullanılan bütün modeller ve kod yapıları Şekil 13 ile Şekil 21 arasında gösterilmektedir.

3.6.1 Text vectorization katmanı ile tek öznitelikli modeller

Bu modellerde optimize etme metodu olarak *adam* ve kayıp fonksiyonu olarak *categorical_crossentropy* kullanılmıştır.

Bu modellerde veriler ham olarak giriş katmanına verilir, *Text Vectorization* katmanında cümle sütunundaki veriler sayısal değerlere dönüştürüldükten sonra *one hot encoding* işlemi yapılmış hedef kelimelerle birlikte ilgili model katmanları ile eğitilir.

Bu modellerde eğitim yaparken *batch_size* parametresi olarak 32, epok sayısı olarak ise 1000 kullanılmıştır.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVec  (None, 78)               0
 torization)

 embedding (Embedding)       (None, 78, 32)            49760

 bidirectional (Bidirectiona  (None, 128)              49664
 l)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 7)                 903

=================================================================
Total params: 100,327
Trainable params: 100,327
Non-trainable params: 0
_________________________________________________________________
```

Şekil 13. Tek Öznitelikli BiLSTM Modeli

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization_1 (TextV  (None, 78)               0
 ectorization)

 embedding_1 (Embedding)     (None, 78, 32)            49760

 flatten (Flatten)           (None, 2496)              0

 dense_1 (Dense)             (None, 128)               319616

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 7)                 903

=================================================================
Total params: 370,279
Trainable params: 370,279
Non-trainable params: 0
_________________________________________________________________
```

Şekil 14. Tek Öznitelikli Baseline ANN Modeli

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization_3 (TextV  (None, 78)               0
 ectorization)

 embedding_3 (Embedding)     (None, 78, 32)            49760

 conv1d (Conv1D)             (None, 72, 32)            7200

 global_max_pooling1d (Globa  (None, 32)               0
 lMaxPooling1D)

 dense_5 (Dense)             (None, 128)               4224

 dropout_3 (Dropout)         (None, 128)               0

 dense_6 (Dense)             (None, 7)                 903

=================================================================
Total params: 62,087
Trainable params: 62,087
Non-trainable params: 0
_________________________________________________________________
```

Şekil 15. Tek Öznitelikli Convolutional Modeli

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization_4 (TextV  (None, 78)               0
 ectorization)

 embedding_4 (Embedding)     (None, 78, 32)            49760

 conv1d_1 (Conv1D)           (None, 72, 32)            7200

 max_pooling1d (MaxPooling1D  (None, 36, 32)           0
 )

 bidirectional_1 (Bidirectio  (None, 256)              164864
 nal)

 dense_7 (Dense)             (None, 128)               32896

 dropout_4 (Dropout)         (None, 128)               0

 dense_8 (Dense)             (None, 7)                 903

=================================================================
Total params: 255,623
Trainable params: 255,623
Non-trainable params: 0
_________________________________________________________________
```

Şekil 16. Tek Öznitelikli Convolutional BiLSTM Modeli

```python
def bilstm(model, vocab_size):
    model.add(Embedding(vocab_size, 32))
    model.add(Bidirectional(LSTM(64)))
    model.add(Dropout(0.1))
    model.add(Dense(sense_count, activation="softmax"))

    return model

def baseline_ann(model, vocab_size):
    model.add(Embedding(vocab_size, 32))
    model.add(Flatten())
    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.1))
    model.add(Dense(sense_count, activation="softmax"))

    return model

def convolutional(model, vocab_size):
    model.add(Embedding(vocab_size, 32))
    model.add(Conv1D(32, 7, activation='relu'))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.1))
    model.add(Dense(sense_count, activation="softmax"))

    return model

def cnn_bilstm(model, vocab_size):
    model.add(Embedding(vocab_size, 32))
    model.add(Conv1D(32, 7, activation='relu'))
    model.add(MaxPooling1D())
    model.add(Bidirectional(LSTM(128)))
    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.1))
    model.add(Dense(sense_count, activation="softmax"))

    return model

def initialize_model(model_name, vocab_size):
    text_dataset = Dataset.from_tensor_slices(train_inputs)

    vectorize_layer = TextVectorization(
     output_mode='int',
     output_sequence_length=padding_size)

    vectorize_layer.adapt(text_dataset.batch(64))

    model = Sequential()

    model.add(vectorize_layer)

    if(model_name=="bilstm"):
        return bilstm(model, vocab_size)
    elif(model_name=="baseline_ann"):
        return baseline_ann(model, vocab_size)
    elif(model_name=="convolutional"):
        return convolutional(model, vocab_size)
    elif(model_name=="cnn_bilstm"):
        return cnn_bilstm(model, vocab_size)
```

```python
model_name = "bilstm"
model = initialize_model(model_name, train_vocab_size)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc', Precision(), Recall(), f1_m])
model.summary()
```

Şekil 17. Text Vectorization Kullanan Modellerin Yapısı

Şekil 17'de görülen yapıda, *Text Vectorization* katmanı kullanan modellerin parametreleri gösterilmektedir. Bu modelde cümle sütunundaki veriler *Text Vectorization* katmanının içinde veri setine dönüştürülmektedir, *padding* (dolgu) parametresi olarak en uzun cümle büyüklüğü verilmektedir. *Embedding* (gömme) katmanında ise cümle sütununun kelime hazinesi boyutu verilmektedir.

3.6.2 BERT ile tek öznitelikli model

Bu modelde veriler, BERT ile eğitilebilecek şekilde işlemlerden geçirilmiştir. Önceden eğitilmiş *dbmdz/bert-base-turkish-128k-cased* BERT modelini kullanarak cümle sütunundaki veriler sayısal değerlere dönüştürülmüş ve model oluşturulmuştur. Hedef kelimeler için ise Keras kütüphanesindeki *to_categorical* fonksiyonu kullanılmıştır. Sayısal değerlere dönüştürülmüş öznitelikler ve işlenmiş hedef kelimeler modele girdi olarak verilmiştir.

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 78)]	0	[]
attention_mask (InputLayer)	[(None, 78)]	0	[]
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 78, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	184345344	['input_ids[0][0]', 'attention_mask[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 768)	0	['tf_bert_model[0][0]']
dense (Dense)	(None, 128)	98432	['global_max_pooling1d[0][0]']
dropout_37 (Dropout)	(None, 128)	0	['dense[0][0]']
dense_1 (Dense)	(None, 32)	4128	['dropout_37[0][0]']
dense_2 (Dense)	(None, 7)	231	['dense_1[0][0]']

Total params: 184,448,135
Trainable params: 184,448,135
Non-trainable params: 0

Şekil 18. Tek Öznitelikli BERT Modeli

```
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(imported_train_dataframe.anli)
y_test = to_categorical(imported_trial_dataframe.anli)
```

```
In [ ]: import transformers

from transformers import AutoTokenizer,TFBertModel
tokenizer = AutoTokenizer.from_pretrained("dbmdz/bert-base-turkish-128k-cased")
bert = TFBertModel.from_pretrained("dbmdz/bert-base-turkish-128k-cased")
```

```
In [ ]: x_train = tokenizer(
    text=train_inputs.values.tolist(),
    add_special_tokens=True,
    max_length=max_length,
    truncation=True,
    padding='max_length',
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
x_test = tokenizer(
    text=trial_inputs.values.tolist(),
    add_special_tokens=True,
    max_length=max_length,
    truncation=True,
    padding='max_length',
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

```
In [ ]: max_len = max_length
input_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
input_mask = Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")
embeddings = bert(input_ids,attention_mask = input_mask)[0]
out = tf.keras.layers.GlobalMaxPool1D()(embeddings)
out = Dense(128, activation='relu')(out)
out = tf.keras.layers.Dropout(0.1)(out)
out = Dense(32,activation = 'relu')(out)
y = Dense(sense_count,activation = 'softmax')(out)
model = tf.keras.Model(inputs=[input_ids, input_mask], outputs=y)
model.layers[2].trainable = True
```

```
In [ ]: from keras.metrics import Precision, Recall

optimizer = Adam(
    learning_rate=5e-05,
    epsilon=1e-08,
    decay=0.01,
    clipnorm=1.0)

loss =CategoricalCrossentropy(from_logits = True)
metric = [CategoricalAccuracy('balanced_accuracy'), Precision(), Recall(), f1_m],

model.compile(
    optimizer = optimizer,
    loss = loss,
    metrics = metric)
```

Şekil 19. Tek Öznitelikli BERT Modeli Yapısı

Şekil 19'da görülen yapıda tek öznitelikli BERT modelinde kullanılan parametreler gösterilmektedir. Bu modelde cümle sütunundaki veriler, *Tokenizer* fonksiyonunda text

parametresi olarak verilmektedir. *padding* parametresi olarak ya en uzun cümle büyüklüğü ya da 200 sayısı verilmektedir.

3.6.3 BERT ile çok öznitelikli model

Bu modelde veriler, BERT modeli ile eğitilebilecek şekilde işlemlerden geçirilmiştir: Önceden eğitilmiş *dbmdz/bert-base-turkish-128k-cased* BERT modelini kullanarak öznitelikler sayısal değerlere dönüştürülmüş ve model oluşturulmuştur. Hedef kelimeler için ise Keras kütüphanesindeki *to_categorical* fonksiyonu kullanılmıştır. Sayısal değerlere dönüştürülmüş öznitelikler ve işlenmiş hedef kelimeler modele verilmiştir.

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_ids (InputLayer)         [(None, 200)]        0           []

 attention_mask (InputLayer)    [(None, 200)]        0           []

 tf_bert_model (TFBertModel)    TFBaseModelOutputWi  184345344   ['input_ids[0][0]',
                                thPoolingAndCrossAt               'attention_mask[0][0]']
                                tentions(last_hidde
                                n_state=(None, 200,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 global_max_pooling1d (GlobalMa  (None, 768)         0           ['tf_bert_model[0][0]']
 xPooling1D)

 dense (Dense)                  (None, 128)          98432       ['global_max_pooling1d[0][0]']

 dropout_37 (Dropout)           (None, 128)          0           ['dense[0][0]']

 dense_1 (Dense)                (None, 32)           4128        ['dropout_37[0][0]']

 dense_2 (Dense)                (None, 7)            231         ['dense_1[0][0]']

==================================================================================================
Total params: 184,448,135
Trainable params: 184,448,135
Non-trainable params: 0
__________________________________________________________________________________________________
```

Şekil 20. Çoklu Öznitelikli BERT Modeli Yapısı

```
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(train_targets)
y_test = to_categorical(trial_targets)
```

```
import transformers

from transformers import AutoTokenizer,TFBertModel
tokenizer = AutoTokenizer.from_pretrained("dbmdz/bert-base-turkish-128k-cased", use_fast=False)
bert = TFBertModel.from_pretrained("dbmdz/bert-base-turkish-128k-cased")
```

```
x_train = tokenizer(
    text=train_inputs.values.tolist(),
    add_special_tokens=True,
    max_length=max_length,
    truncation=True,
    padding='max_length',
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True,
    is_split_into_words=True)
x_test = tokenizer(
    text=trial_inputs.values.tolist(),
    add_special_tokens=True,
    max_length=max_length,
    truncation=True,
    padding='max_length',
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True,
    is_split_into_words=True)
```

```
input_ids = x_train["input_ids"]
attention_mask = x_train['attention_mask']
```

```
max_len = max_length
input_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
input_mask = Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")
embeddings = bert(input_ids,attention_mask = input_mask)[0]
out = tf.keras.layers.GlobalMaxPool1D()(embeddings)
out = Dense(128, activation='relu')(out)
out = tf.keras.layers.Dropout(0.1)(out)
out = Dense(32,activation = 'relu')(out)
y = Dense(sense_count,activation = 'softmax')(out)
model = tf.keras.Model(inputs=[input_ids, input_mask], outputs=y)
model.layers[2].trainable = True
```

```
from keras.metrics import Precision, Recall

optimizer = Adam(
    learning_rate=5e-05,
    epsilon=1e-08,
    decay=0.01,
    clipnorm=1.0)

loss =CategoricalCrossentropy(from_logits = True)
metric = [CategoricalAccuracy('balanced_accuracy'), Precision(), Recall(), f1_m],

model.compile(
    optimizer = optimizer,
    loss = loss,
    metrics = metric)
```

Şekil 21. Çok Öznitelikli Bert Modeli Yapısı

Şekil 21'de görülen yapıda çok öznitelikli BERT modelinde kullanılan parametreler gösterilmektedir. Bu modelde cümle sütunundaki veriler, *Tokenizer* fonksiyonunda *text* parametresi olarak verilmektedir. *padding* parametresi olarak ya en uzun cümle büyüklüğü ya da 200 sayısı verilmektedir.

3.6.4 Kullanılan metrikler

Bu çalışmada elde edilen sonuçları değerlendirmek için duyarlılık, geri çağırım ve F1 skoru metrikleri kullanılmaktadır. Bu metrikleri elde etmek için *sklearn.metrics* kütüphanesinden *classification_report* fonksiyonu kullanılmaktadır. Metriklerin hesaplanmasında, *true positive* (TP) [doğru pozitif], *true negative* (TN) [doğru negatif], *false positive* (FP) [yanlış pozitif], *false negative* (FN) [yanlış negatif] değerleri kullanılmaktadır. Önceki çalışmalar ile karşılaştırma yapmak için önceki çalışmaların duyarlılık ve geri çağırım değerleri kullanılarak F1 skorları hesaplanmaktadır.

3.6.4.1 Duyarlılık

Duyarlılık, yapılan tüm tahminlerin tüm pozitif tahminlere oranıdır. Duyarlılık metriğinin en iyi değeri 1 iken en kötü değeri 0'dır. Duyarlılık denklemi (3.2)'deki gibidir:

Duyarlılık = Hata! (3.2)

3.6.4.2 Geri çağırım

Geri çağırım pozitif olarak tahmin edilmesi gereken değerlerin ne kadarının pozitif olarak tahmin edildiğinin oranıdır. Geri çağırım metriğinin en iyi değeri 1 iken en kötü değeri 0'dır. Geri çağırım denklemi (3.3)'deki gibidir:

Geri Çağırım = Hata! (3.3)

3.6.4.3 F1 skoru

F1 skoru, duyarlılık ve geri çağırım değerlerinin harmonik ortalamasıdır. F1 skorunun en yüksek değeri 1, en düşük değeri 0'dır. F1 skorunun denklemi (3.4)'deki gibidir:

$$F_1 \text{ Skoru} = 2 \times \text{Hata!} \tag{3.4}$$

4. BULGULAR VE TARTIŞMA

4.1 Bulgular

Bu çalışmada SemEval-2007 çalıştayında kullanılan işaretlenmiş Türkçe veriler derin öğrenme modelleri ile analiz edilmiştir. Makine, derin öğrenme modelleri ile cümle ve anlam arasındaki ilişkiler verilerek eğitildiğinde, önceki birkaç çalışmaya göre daha yüksek metrik değerleri elde edilmiştir.

Tablo 7, Tablo 8 ve Tablo 9'da her sütundaki en yüksek üç değer koyu harflerle işaretlenmiştir.

Tablo 7. Önceki Birkaç Çalışmada Alınan Metrik Değerleri

Çalışma		**İsim**	**Fiil**	**Diğer**
SemEval 2007-İnce Ayrım	Duyarlılık	0.150	0.100	0.130
	Geri Çağırım	0.50	0.380	0.500
	F1 Skoru	0.231	0.158	0.206
Naive Bayes	Duyarlılık	0.610	0.460	0.560
	Geri Çağırım	0.580	0.430	0.580
	F1 Skoru	**0.595**	**0.444**	**0.570**
Karar Ağacı Algoritması	Duyarlılık	0.550	0.418	0.514
	Geri Çağırım	0.590	0.535	0.608
	F1 Skoru	**0.569**	**0.469**	0.557
K En Yakın Komşuluk Algoritması	Duyarlılık	0.596	0.412	0.460
	Geri Çağırım	0.460	0.116	0.486
	F1 Skoru	0.519	0.181	0.472
Destekçi Vektör Makineleri Algoritması	Duyarlılık	0.496	0.264	0.786
	Geri Çağırım	0.222	0.160	0.992
	F1 Skoru	0.307	0.199	**0.877**
Rastgele Orman Algoritması	Duyarlılık	0.954	0.160	0.797
	Geri Çağırım	0.460	0.965	0.852
	F1 Skoru	**0.620**	**0.274**	**0.823**

Tablo 8. Bu Çalışmada Alınan Ağırlıklı Ortalama Metrik Değerleri

Çalışma		**İsim**	**Fiil**	**Diğer**
BERT	Duyarlılık	0.632	0.673	0.565
	Geri Çağırım	0.592	0.700	0.600
	F1 Skoru	**0.583**	**0.662**	**0.543**
Çok Öznitelikli BERT	Duyarlılık	0.710	0.703	0.613
	Geri Çağırım	0.645	0.715	0.637
	F1 Skoru	**0.647**	**0.694**	**0.600**
BiLSTM	Duyarlılık	0.491	0.582	0.490
	Geri Çağırım	0.419	0.489	0.473
	F1 Skoru	0.404	0.465	0.457
Baseline ANN	Duyarlılık	0.443	0.556	0.448
	Geri Çağırım	0.462	0.543	0.497
	F1 Skoru	0.408	**0.490**	**0.458**
Convolutional	Duyarlılık	0.513	0.536	0.498
	Geri Çağırım	0.447	0.502	0.482
	F1 Skoru	**0.421**	0.464	0.443
Convolutional BiLSTM	Duyarlılık	0.464	0.557	0.408
	Geri Çağırım	0.359	0.449	0.322
	F1 Skoru	0.351	0.442	0.337

Tablo 7'de, SemEval-2007 çalıştayı Türkçe veri setini kullanan önceki çalışmalarda alınan sonuçlar (Aktaş, 2021) yer almaktadır. Tablo 8'de ise derin öğrenme algoritmaları kullanılarak aynı veri seti üzerinde yapılan çalışmalarda alınan sonuçların ağırlıklı ortalaması görülmektedir.

Tablo 9. Bu Çalışmada Alınan Macro Metrik Değerleri

Çalışma		İsim	Fiil	Diğer
BERT	Duyarlılık	0.558	0.578	0.438
	Geri Çağırım	0.543	0.533	0.455
	F1 Skoru	**0.518**	**0.523**	**0.408**
Çok Öznitelikli BERT	Duyarlılık	0.667	0.593	0.503
	Geri Çağırım	0.609	0.540	0.523
	F1 Skoru	**0.598**	**0.545**	**0.492**
BiLSTM	Duyarlılık	0.404	0.444	0.378
	Geri Çağırım	0.391	0.432	0.368
	F1 Skoru	0.342	0.379	**0.347**
Baseline ANN	Duyarlılık	0.408	0.511	0.317
	Geri Çağırım	0.358	0.427	0.340
	F1 Skoru	0.333	**0.415**	0.315
Convolutional	Duyarlılık	0.436	0.461	0.375
	Geri Çağırım	0.419	0.410	0.350
	F1 Skoru	**0.363**	0.390	0.323
Convolutional BiLSTM	Duyarlılık	0.388	0.464	0.313
	Geri Çağırım	0.353	0.432	0.262
	F1 Skoru	0.303	0.391	0.262

Tablo 9 üzerinde bu çalışmada derin öğrenme kullanılarak alınan sonuçların makro ortalaması görülmektedir.

4.2 Tartışma

Bu çalışmada elde edilen sonuçlar Tensorflow ve Keras kütüphaneleri ile derin öğrenme algoritmaları kullanılarak elde edilen sonuçlardır. Model olarak basit ve karmaşık sinir ağları kullanılmıştır. Kullanılan sinir ağı modelleri Baseline ANN, Convolutional, Convolutional BiLSTM, BiLSTM ve BERT modelleridir; kullanılan modellerin parametrelerini değiştirerek farklı modeller veya veri setleri üzerinde de deneyler yapılabilmektedir.

Önceki çalışmalarda, bu çalışmadan farklı yöntemler kullanılmıştır. SemEval-2007 çalıştayının Türkçe çalışmasında Naive Bayes algoritmasına benzeyen ve istatiksel sonuç veren bir algoritma kullanılmıştır. Bu çalışma üzerine yapılan Tüysüz ve Güvenoğlu'nun (2014) ve Aktaş'ın (2021) çalışmalarında, WEKA yazılımı ve makine öğrenme algoritmaları kullanılmıştır. Tüysüz ve Güvenoğlu'nun (2014) yaptığı çalışmada Naive Bayes ve Karar Ağacı algoritmaları kullanılmıştır. Aktaş'ın (2021) yaptığı çalışmada ise K En Yakın Komşuluk, Destekçi Vektör Makineleri ve Rastgele Orman algoritmaları kullanılmıştır.

Bu çalışmada alınan sonuçların ağırlıklı ortalamalarına bakıldığında BERT algoritmaları, *Text Vectorization* katmanı ile yapılan çalışmalara göre daha büyük başarı göstermiştir. BERT algoritmaları arasında çok öznitelikli BERT algoritması isim, fiil ve diğer kategorilerinde en yüksek F1 skorunu alan çalışma olmuştur. *Text Vectorization* katmanı ile yapılan çalışmalar arasında en iyi F1 skorunu isim kategorisinde Convolutional modeli almaktadır. Fiil ve diğer kategorilerinde ise Baseline ANN modeli en iyi F1 skoruna sahiptir. Modellerdeki makro ortalamalara bakıldığında BERT algoritmaları, *Text Vectorization* katmanı ile yapılan çalışmalara göre daha büyük başarı göstermiştir. BERT algoritmaları arasında çok öznitelikli BERT çalışması üç kategoride de daha iyi F1 sonuçları almıştır. *Text Vectorization* katmanı ile yapılan çalışmalarda ise isim kategorisinde Convolutional modeli, fiil kategorisinde Baseline ANN modeli ve diğer kategorisinde BiLSTM modeli en iyi F1 skorunu elde etmiştir.

Bu tezde yer verilen bu ve önceki çalışmaların tüm deney sonuçlarına bakıldığında en iyi F1 skorunu elde eden çalışma isim ve fiil kategorisinde çok öznitelikli BERT modeli ve diğer kategorisinde ise Destekçi Vektör Makineleri algoritmasıdır.

5. SONUÇ ve ÖNERİLER

Bu çalışmanın amacı derin öğrenmenin Türkçe dilinde kelime anlamı belirginleştirmede aldığı sonuçları görmeyi, alınan sonuçları inceleyip önceki yapılan çalışmalarda kullanılan metotlar ile karşılaştırmayı kapsamaktadır. Karşılaştırma yapmak için önceden yapılan birkaç çalışmanın da kullandığı SemEval-2007 çalıştayında kullanılan Türkçe veriler kullanılmıştır.

Bu tez kapsamında, SemEval-2007 çalıştayındaki Türkçe verilerdeki kodlayıcı sorunları düzeltilmiş ve deneyde kullanılmayacak özel karakterler temizlenmiştir. BERT modellerinde BERT modellerinde, BERT'in *Tokenizer* fonksiyonu ve diğer derin öğrenme modellerinde ise *Text Vectorization* katmanı kullanılarak Türkçe veriler sayısal değerlere dönüştürülmüştür. Bu sayısal değerlerin uygulanabileceği yapay sinir ağı modelleri oluşturulmuş, giriş katmanında bu sayısal değerler verilmiştir. Bu modeller ile veriler eğitilip alınan sonuçlar ağırlıklı ortalama ve makro ortalama olarak yorumlanmıştır. Bu yorumlamanın sonucunda çok öznitelikli BERT modelinin bu ve önceki çalışmalara kıyasla genellikle daha iyi F1 skoru aldığı gözlemlenmiştir. Bu çalışmadaki *Text Vectorization* katmanı içeren modellerde ise kategori bazında daha düşük veya daha yüksek F1 skorları alabildiği gözlemlenmiştir.

BERT modelinin diğer yöntemlere göre daha iyi sonuç almasının nedeni BERT modelinin yapısının kullanılan makine ve derin öğrenme algoritmalarından farklı olmasıdır. BERT modeli, girdi olarak alınan veriyi kelimeler halinde ayırmak yerine cümleler olarak ayırmaktadır. Ayrıca modelde kullanılan çift yönlü öz-dikkat (*self-attention*) katmanı cümle içindeki bir kelimenin hem sağındaki hem de solundaki kelimeleri inceleyerek içerisinde bulunduğu cümlenin bağlamını anlayabilmesini sağlamaktadır (Devlin ve diğerleri, 2019).

Bu çalışmada alınan deney sonuçları incelendiğinde, derin öğrenmede kullanılan gelişmiş yapay sinir ağlarının, WEKA yazılımı ile yapılan makine öğrenme algoritmalarına göre daha iyi sonuçlar verebildiği görülmüştür.

Bu çalışmada, kullanılan sinir ağı modellerinin “isim”, “fiil”, ve “diğer” kategorilerinde aldıkları F1 skorları elde edilmiştir. Bu skorların birbirlerine yakınlığı, önceki çalışmalarda gözlemlenen makine ya da istatistiksel öğrenme modelleriyle yine aynı kategorilerde aldıkları F1 skorlarının birbirlerine olan yakınlığından daha tutarlıdır. Örneğin, sinir ağlarında en büyük F1 skoru farkı Convolutional BiLSTM modelindeki “fiil” kategorisi ile “diğer” kategorisi arasında olan 12,9 farkıdır. Önceki çalışmalarda ise, SemEval İnce Ayrım ve Karar Ağaçları algoritmaları dışındaki F1 skorları, farklı kategoriler arasında 12,9’dan daha büyük fark göstermiştir.

Gelecekte yapılacak çalışmalarda, bu tezde kullanılan Türkçe veri setiyle “anlk” sütunu üzerinde kullanılan makine öğrenme algoritmaları ve derin öğrenme modelleri kullanılabilir. Alınan sonuçlar, SemEval-2007 Türkçe çalışmasında alınan Kaba Ayrım algoritma sonuçları ile karşılaştırılabilir. Yapılabilecek bir başka çalışma olarak ise SemEval-2007 Türkçe veri seti haricinde daha büyük bir veri seti hazırlanıp daha az veya daha fazla öznitelik kullanılarak alınan sonuçlar incelenebilir.

KAYNAKLAR

Aktaş, Z. (2021). *Makine öğrenmesi yöntemleriyle Türkçe için kelime anlamı belirginleştirme uygulaması* (Yayın no. 678713) [Yayınlanmış yüksek lisans tezi, Maltepe Üniversitesi]. https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

Arukgoda, J., Bandara, V., Bashani, S., Gamage, V., ve Wimalasuriya, D. (2014). A word sense disambiguation technique for Sinhala. *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, (pp. 207-211). https://doi.org/10.1109/ICAIET.2014.42

Bengio, Y., Ducharme, R., Vincent, P., ve Jauvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research, 3*, 1137-1155

Borthwick, A., Sterling, J., Agichtein, E., ve Grishman, R. (1998). Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. C. Eugene (Ed.), *Sixth Workshop on Very Large Corpora, VLC@COLING/ACL 1998, Montreal, Quebec, Canada, August 15-16, 1998*.

Devlin, J., Chang, M.-W., Lee, K., ve Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies, volume 1 (Long and short Papers)* (pp. 4171-4186). Minneapolis, Minnesota: Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Gumizawa, Y., ve Yamamoto, K. (2018). Analysis of Japanese word sense disambiguation through Kana-Kanji conversion. *Journal of Natural Language Processing, 25*(3), 255-293. https://doi.org/10.5715/jnlp.25.255

Hochreiter, S., ve Schmidhuber, J. (1997, 12). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 06*(02), 107-116. https://doi.org/10.1142/S0218488598000094

Kim, P. (2017). *MATLAB deep learning: With machine learning, neural networks and artificial intelligence.* Apress.

Knisley, J., Glenn, L. L., Joplin, K., ve Carey, P. (2007). Artificial neural networks for data mining and feature extraction. D. Hong, ve Y. Shyr (Ed.) *Quantitative medical data analysis using mathematical tools and statistical techniques* içinde (pp. 321-332). World Scientific. https://doi.org/10.1142/9789812772121_0015

Li, J., Li, R., ve Hovy, E. (2014). Recursive deep models for discourse parsing. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2061-2069). Doha, Qatar: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1220

Ma, J., Gao, W., Joty, S. R., ve Wong, K.-F. (2022). An attention-based rumor detection model with tree-structured recursive neural networks. *ACM Transactions on Intelligent Systems and Technology*, *11*(4), 1-28. https://doi.org/10.1145/3391250

Manning, C. D., ve Schütze, H. (1999). *Foundations of statistical natural language processing.* The MIT Press.

Moposita, T., Trojman, L., Crupi, F., Lanuzza, M., ve Vladimirescu, A. (2022). Voltage-to-voltage sigmoid neuron activation function design for artificial neural networks. *2022 IEEE 13th Latin America Symposium on Circuits and System (LASCAS)* (pp.1-4). Puerto Varas, Chile: IEEE. https://doi.org/10.1109/LASCAS53948.2022.9789075

Nielsen, M. A. (2015). *Neural networks and deep learning.* Determination Press.

Nithyanandan, S., ve C, R. (2019). Deep learning models for word sense disambiguation: a comparative study. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3437615

Orhan, Z. (2006). *Türkçe metinlerdeki anlam belirsizliği olan sözcüklerin bilgisayar algoritmaları ile anlam belirginleştirmesi* (Yayın no. 178147) [Yayınlanmış doktora tezi, İstanbul Üniversitesi]. https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

Orhan, Z., Çelik, E., ve Demirgüç, N. (2007). SemEval-2007 task 12: Turkish lexical sample task. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)* (pp. 59-63). Prague, Czech Republic: Association for Computational Linguistics.

S, H. J., ve Raju, M. (2022). A study on deep learning. *International Journal for Research in Applied Science and Engineering Technology*, *10*(11), 961-964. https://doi.org/10.22214/ijraset.2022.47486

Sak, H., Senior, A., ve Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proc. Interspeech 2014* (pp. 338-342). Singapore. https://doi.org/10.21437/Interspeech.2014-80

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., ve Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484-489. https://doi.org/10.1038/nature16961

Singh, V. P., ve Kumar, P. (2020). Word sense disambiguation for Punjabi language using deep learning techniques. *Neural Computing and Applications*, *32*(8), 2963-2973. https://doi.org/10.1007/s00521-019-04581-3

Sridevi, N., Kulkarni, V., ve Maria Navin, J. (2019). Machine learning algorithms: diagnosing breast cancer. *International Journal of Recent Technology and Engineering, 8*(2S6), 849-851. https://doi.org/10.35940/ijrte.B1157.0782S619

Sun, X.-R., Lv, S.-H., Wang, X.-D., ve Wang, D. (2017). Chinese word sense disambiguation using a LSTM. *ITM Web of Conferences, 12*, 01027. https://doi.org/10.1051/itmconf/20171201027

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., ve Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1-9). Boston, MA: IEEE. https://doi.org/10.1109/CVPR.2015.7298594

Tunalı, V. (2022). Improved prioritization of software development demands in Turkish with deep learning-based NLP. *IEEE Access, 10*, 40249-40263. https://doi.org/10.1109/ACCESS.2022.3167269

Türk Dil Kurumu. (2022). *Yüz ne demek.* 12 Aralık 2022 tarihinde https://sozluk.gov.tr/ adresinden indirildi.

Tüysüz, M. A., ve Güvenoğlu, E. (2014). Türkçe için karşılaştırmalı bir kelime anlamı belirginleştirme uygulaması. M. Akgül, U. Çağlayan, E. Derman, ve A. Özgit (Ed.) *Akademik Bilişim ’14* (pp. 869-874). Mersin.

Williams, R. J., ve Zipser, D. (1989). Experimental analysis of the real-time recurrent. *Connection Science, 1*(1), 87-111. https://doi.org/10.1080/09540098908915631

Xing, S., ve Wu, C. (2020). Implementation of a neuron using sigmoid activation function with CMOS. *2020 IEEE 5th International Conference on Integrated Circuits and Microsystems (ICICM)* (pp. 201-204). Nanjing, China: IEEE. https://doi.org/10.1109/ICICM50929.2020.9292239

ÖZGEÇMİŞ

Tunç GÜMÜŞDAĞ

Eğitim

Derece	Yıl	Üniversite, Enstitü, Anabilim/Anasanat Dalı
Lisans	2019	Maltepe Üniversitesi, Bilgisayar Mühendisliği Fakültesi, Yazılım Mühendisliği (İngilizce) Lisans

İş/İstihdam

Yıl	Görev
2021 - 2022	Yazılım Kalite ve Güvence Mühendisi (Otomasyon), Getir
2020 – 2021	Yazılım Kalite ve Güvence Mühendisi, Intertech
2019 – 2022	Yazılım Test Otomasyon Mühendisi, Testinium