İSTANBUL TİCARET ÜNİVERSİTESİ FEN BİLİMLERİ DERGİSİ

İstanbul Commerce University Journal of Science

http://dergipark.org.tr/ticaretfbd

Derleme Makale / Review Article

DOĞAL DİL METİNLERİNDEN PROGRAMLAMA DİLİ KODU OLUŞTURMA ÇALIŞMALARI: BİR DERLEME ÇALIŞMASI*

NATURAL LANGUAGE TEXT TO PROGRAMMING LANGUAGE CODE GENERATION STUDIES: A REVIEW

Ayşegül HATİPOĞLU[1] **Turgay Tugay BİLGİN[2]**

https://doi.org/10.55071/ticaretfbd.1354040

Sorumlu Yazar / Corresponding Author
aysegul.hatipoglu@bilecik.edu.tr

Geliş Tarihi / Received
01.09.2023

Kabul Tarihi / Accepted
21.03.2024

Öz

Son yıllarda Doğal Dil İşleme (DDİ) alanındaki gelişmelerin hız kazanması, araştırmacıların ve programcıların bu alana olan ilgisini büyük ölçüde artırmıştır. Bilgisayar programlarını doğal dil komutlarıyla yazma konsepti, birçok araştırmacının odak noktası haline gelmiştir. Literatür incelendiğinde, doğal dil ile programlama üzerine yapılan araştırmaların uzun bir geçmişe sahip olduğu açıkça görülmektedir. Bu uzun soluklu araştırmalar, çeşitli çözüm önerilerini beraberinde getirmiş ve kural tabanlı yöntemlerden, olasılık tabanlı yöntemlere, makine öğrenmesi yöntemlerinden derin öğrenme yöntemlerine kadar bir dizi çözüm yaklaşımının ortaya çıkmasına neden olmuştur. Literatürdeki çalışmalar tarihsel olarak kategorize edildiğinde geçmiş tarihli çalışmalarda kural tabanlı ya da istatistik tabanlı modeller üzerine yoğunlaştığı görülürken günümüze yaklaşıldıkça makine öğrenmesi ve derin öğrenme temelli çalışmaların arttığı görülmektedir. Kural tabanlı yöntemler, olasılık tabanlı yöntemler, makine öğrenmesi yöntemleri, derin öğrenme yöntemleri gibi çeşitli yaklaşımların geliştirildiği literatürde, bu çeşitlilik yeni araştırmacıların bu alana giriş yaparken karşılaşabileceği karmaşıklığı artırabilmektedir. Bu çalışma, doğal dil girdileriyle programlama dili kodu oluşturma çalışmalarına yönelik literatürde geliştirilen 32 yöntemin detaylı bir incelenmesini sunmaktadır. Çalışmanın amacı, literatürde tespit edilen çeşitli yöntemlerin zaman içerisindeki değişimlerinin gözden geçirilmesi, çalışmaların geniş bir perspektiften incelenerek genel bir çerçeve içinde toplanması ve bu alanda çalışacak olan araştırmacılara rehberlik edebilmesidir.

Anahtar Kelimeler: Derin öğrenme, doğal dil işleme, kod üretimi, makine öğrenmesi.

Abstract

The recent surge in advancements in Natural Language Processing (NLP) has significantly heightened the interest of researchers and programmers in this field. The concept of writing computer programs using natural language commands has become a focal point for many researchers. Upon reviewing the literature, it is evident that research on natural language programming has a long history. These extensive studies have led to the proposal of various solutions, ranging from rule-based methods to probability-based approaches, machine learning methods, and deep learning techniques. When the studies in the literature are categorized historically, it is seen that the past studies focused on rule-based or statistical-based models, while machine learning and deep learning-based studies have increased as we approach the present. The diversity of approaches, including Rule-based methods, probability-based methods, machine learning methods, deep learning methods, and others, as found in the literature, can potentially confuse newcomers entering this field. This paper presents a detailed review of 32 methods developed in the literature for generating programming language code with natural language input. The goal of this study is to review the changes in various methods identified in the literature over time, to collect the studies in a general framework by examining them from a broad perspective and provide guidance to researchers intending to work in this area.

Keywords: Code generation, deep learning, machine learning, natural language processing.

Bu yayın Ayşegül HATİPOĞLU isimli öğrencinin Bursa Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Tezli Yüksek Lisans Programındaki Yüksek Lisans tezinden üretilmiştir.

[1]Bilecik Şeyh Edebali Üniversitesi, Mühendislik Fakültesi, Bilgisayar Yazılımı Anabilim Dalı, Bilecik, Türkiye.
aysegul.hatipoglu@bilecik.edu.tr, Orcid.org/0000-0003-1584-0945.

[2]Bursa Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü, Bursa, Türkiye.
turgay.bilgin@btu.edu.tr, Orcid.org/0000-0002-9245-5728.

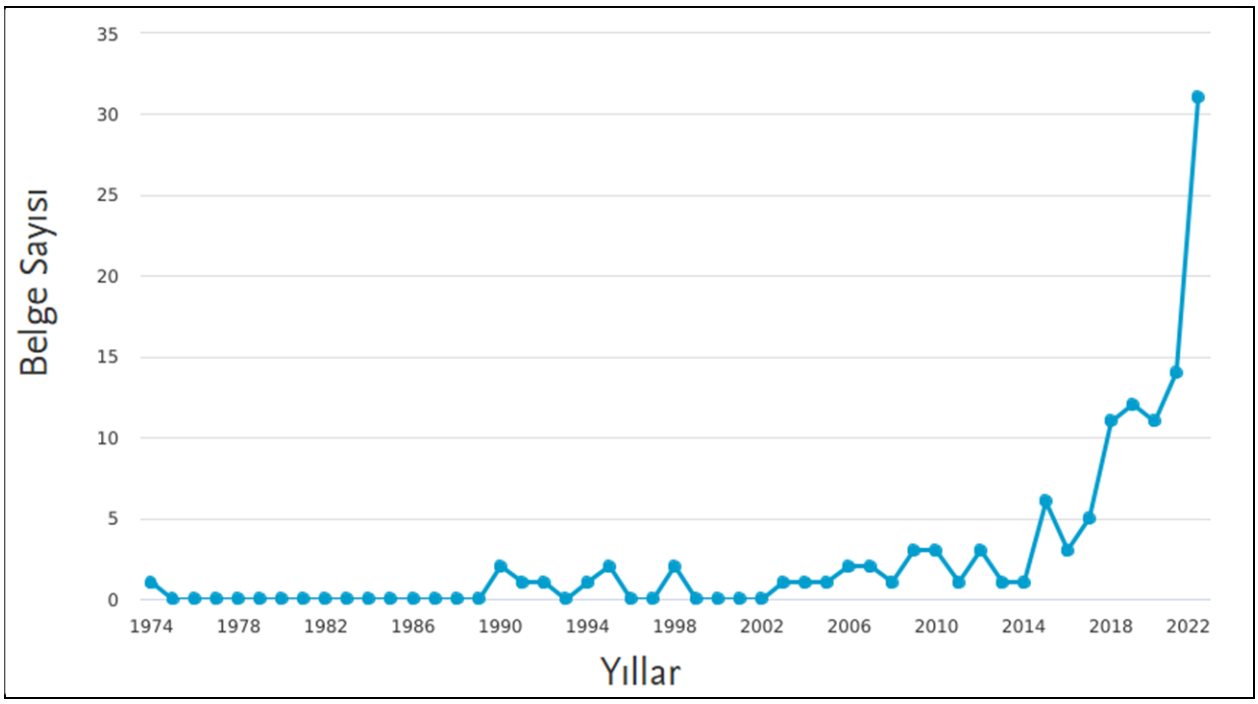
1. GİRİŞ

Uzun yıllardır, bilgisayar bilimcileri tarafından programlama dili kodunun doğal dil kullanılarak üretilmesi konusunda çeşitli araştırmalar yürütülmektedir. Son yıllarda, makine öğrenmesi ve derin öğrenme alanlarında kaydedilen ilerlemeler, bu alanda önemli başarılar elde edilmesine olanak tanımıştır. Doğal dil işleme çalışmalarında başarılı sonuçlar elde edebilmek için, kuşkusuz en önemli faktörlerden biri veri kalitesidir. Teknolojik gelişmelerle birlikte artan büyük bir veri dünyasının içinde bulunmaktayız. Ayrıca, pandemi döneminde internet ticareti, uzaktan toplantılar, online hukuk işlemleri, uzaktan eğitim gibi birçok örnek veri dünyasının son yıllarda önemli ölçüde genişlediğini göstermektedir. Teknolojinin artan önemiyle birlikte, veri bilimi alanına olan ilgi de artmıştır. Bu artışa paralel olarak, doğal dil işleme alanında ve özellikle doğal dilden programlama dili kodu üretme konusunda bir artış yaşandığı söylenebilir.

Doğal diller, programlama dilleri kadar teknik bilgi gerektirmediğinden programlama dillerine kıyasla kullanımları kolay ve daha anlaşılabilirdir. Doğal dil ifadelerinden programlama dili kodu oluşturabilme imkânı çeşitli uygulama alanlarında ve farklı disiplinler arasında fayda sağlayabilir. Örneğin yazılım geliştiricilerin daha kısa sürede kolay yazılım geliştirebilmelerine imkân sağlayabileceği gibi otomasyon sistemleri gibi alanlarda kullanılarak kodlama bilgisi olmayan kişilerinde ilgili süreçlerde aktif rol almasını sağlayarak zaman ve maliyetten açısından verimliliği artırabilir. Ayrıca programlama dili öğrenme süreçlerini basite indirgeyerek eğitim öğretim alanında kolaylıklar sunabilir. Bahsedilen avantajlar göz önünde bulundurulduğunda, bu alanda gerçekleştirilecek çalışmalar önem arz etmektedir.

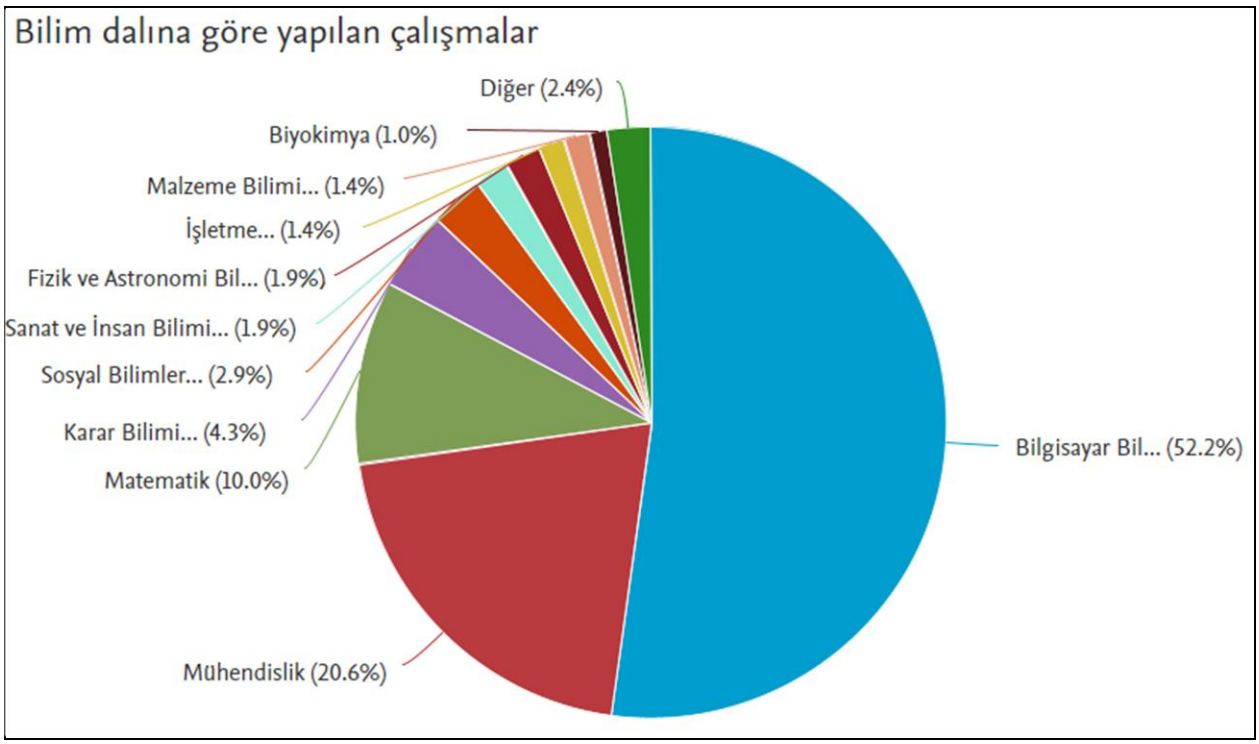
Doğal dilden programlama dili kodu üretme, geniş kapsamlı, uzun süredir güncelliğini koruyan ve hala popülerliğini sürdüren bir konudur. Bu çalışma kapsamında, doğal dilden programlama dili kodu üretme konusunda mevcut literatürde geliştirilen yöntemlerin zaman içindeki değişimi gözden geçirilmekte ve farklı çalışmalardan elde edilen bulgular yarı sistematik bir şekilde incelenmektedir. Zaman içinde mevcut yöntemlerin eksikliklerinin giderilmesi için geliştirmeler yapıldığı gibi yeni yöntemlerin ortaya çıktığı da gözlemlenmektedir. Geliştirilen bu yöntemler, yöntemlerin değişim süreçleri, değerlendirme metrikleri ve veri setleri, mevcut durumdaki eksiklikleri belirlemek ve yeni çalışmalar yapılmasına yardımcı olmak amacıyla bu literatür çalışmasında sunulmuştur.

Şekil 1, Şekil 2 ve Şekil 3 Scopus veri tabanında "Generating programming language code using natural language" ifadesi kullanılarak ve 1974-2022 yılları aralığı için filtreleme yapılarak gerçekleştirilen bir arama sonucunda elde edilen 3 farklı grafiği içermektedir. Şekil 1'deki grafik 1974-2022 yılları arasında bahsedilen konuda üretilen makale sayılarını göstermektedir. Özellikle 2000'li yıllardan itibaren gerçekleştirilen çalışma sayılarında belirgin dalgalanmaların yaşandığı ve bu alana olan eğilimin giderek arttığı gözlemlenmektedir.

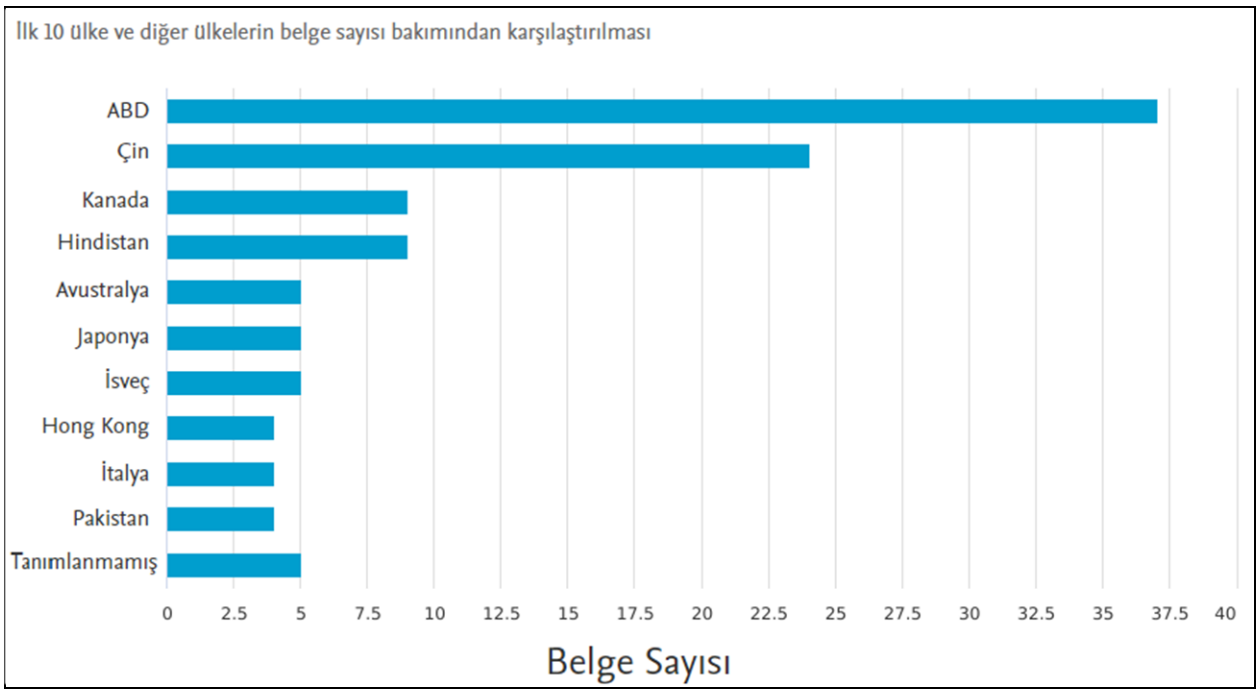


Şekil 1. 1974-2022 Yılları Arasında Doğal Dilden Kod Oluşturma Problemi için Yapılan Yayın Sayısı (Scopus, 2023)

Şekil 2'de kod üretme problemi üzerine yapılan çalışmaların bilgisayar bilimleriyle sınırlı kalmayıp matematik, fizik, tıp gibi farklı disiplinler arasında da gerçekleştirildiği görülmektedir. Bu durum, bahsedilen alanın farklı bilim dallarında da popülerliğini koruduğunu göstermektedir.



Şekil 2. 1974-2022 Yılları Arasında Doğal Dilden Kod Oluşturma Problemi için Yapılan Yayın Sayısının Alanlara Göre Dağılımı (Scopus, 2023).



Şekil 3. 1974-2022 Yılları Arasında Doğal Dilden Kod Oluşturma Problemi İçin Çalışma Yapan Ülkeler (Scopus, 2023).

Şekil 3'te, bu alanda en fazla çalışma yapan ülkelerin çalışma grafiği sunulmaktadır. Şekilden de anlaşılacağı üzere, dünya genelinde giderek popüler hale gelen bir konu ülkemizde henüz yeterince ilgiyi görememektedir. Bu durum, ülkemizde yapılacak çalışmaların güncel teknolojilere ayak uydurma açısından kritik bir öneme sahip olduğunu göstermektedir.

2. DOĞAL DİLDEN PROGRAMLAMA DİLİ KODU OLUŞTURMA PROBLEMİ İÇİN KULLANILAN YÖNTEMLER

2.1. Temel Terimler

2.1.1. Doğal dil işleme

Doğal dil işleme, bilgisayarların doğal dil ifadelerini anlamak ve üzerinde değişiklikler gerçekleştirebilmek için bu doğal dil verilerini nasıl kullanılabileceğini araştıran bir uygulama alanıdır. Bu alandaki araştırmacılar, bilgisayar sistemlerinin belirli görevleri yerine getirebilmesi için doğal dilleri anlamasını ve manipüle etmesini sağlamak amacıyla insanların dili anlamlandırma süreçlerini anlamayı hedefler (Chowdhary, 2020).

Doğal dil anlama, insanların kullandığı doğal dil ifadelerini anlama sürecini temsil eder ve bu, Doğal dil işlemenin önemli bir alt alanını oluşturur. Doğal dil anlama sadece insanların kendi aralarındaki iletişimi kolaylaştırmakla kalmaz, aynı zamanda insan-makine etkileşimine de katkı sağlar. Son yıllarda, makine öğrenmesi ve derin öğrenme terimleri sıklıkla doğal dil işleme kavramlarıyla birlikte anılmaktadır. Doğal dil işleme alanında yapılan çalışmalara bakıldığında, derin öğrenme mimarileri ile geliştirilen yöntemlerin diğer pek çok yöntemden daha başarılı sonuçlar elde ettiği görülmektedir.

İncelenen makalelerin temelinde, doğal dilin anlaşılması gibi kritik bir problem yatmaktadır. Kelimelerin morfolojik yapısı, kullanılan doğal dilin doğası, dilin gramer kuralları, doğal dil

metinlerinin yan anlamları, bağlamları, aldığı ekler vb. birçok kritik etmen, dil anlama sürecini karmaşıklaştırmaktadır.

2.2. Doğal Dil İfadelerinden Programlama Dili Kodu Üretme

Programlama dilleri, bilgisayarlarda belirli görevleri gerçekleştirmek üzere kurallara sahip, belirli formatlara uygun ve kesin talimatlar içeren dillerdir. İnsanlar ve bilgisayarlar arasındaki iletişimi sağlayan araçlardır. Bilgisayarlar kendilerine verilen talimatları işleyerek sonuç üretirler. Açık kaynak kodlu dağıtılan veya ticari olarak satılan birçok programlama dili bulunmaktadır; her birinin avantajları, dezavantajları, farklı kullanım alanları ve sözdizimi kuralları mevcuttur. Bu farklılıklar, birden fazla programlama dilini öğrenmeyi zorlaştırabilir. Bu zorlukları aşmak için, kullanıcılara günlük yaşamlarında iletişim için kullandıkları doğal dil aracılığıyla yazılım geliştirme olanağı sunmak önemlidir.

İnsanlar, doğal dil ile programlama dili arasındaki ilişkiyi anlamak için düşünme yeteneklerini kullanırlar. İstenen programlama dilinin üst düzey kod kalıpları, farklı kod ifadeleri arasında aynı anlamı taşıyan kalıpları içerebilir. Örneğin, bir terim doğal dilde bir anlam ifade ederken programlama komutlarında farklı bir anlam taşıyabilir. Kaynak dildeki dil bilgisi özellikleri ve düşük seviyeli bir programlama dilinde uygulama yapmanın karmaşıklığı gibi nedenlerle bu ilişkileri makinelere aktarmak zordur. Bu zorluğun üstesinden gelmek için araştırmacılar zaman içinde farklı yaklaşımlar benimsemişlerdir. Literatürdeki çalışmalar tarihsel olarak kategorize edildiğinde geçmiş tarihli çalışmalarda kural tabanlı ya da istatistik tabanlı modeller üzerine yoğunlaştığı görülürken günümüze yaklaşıldıkça daha çok makine öğrenmesi, derin öğrenme temelli çalışmaların arttığı görülmektedir.

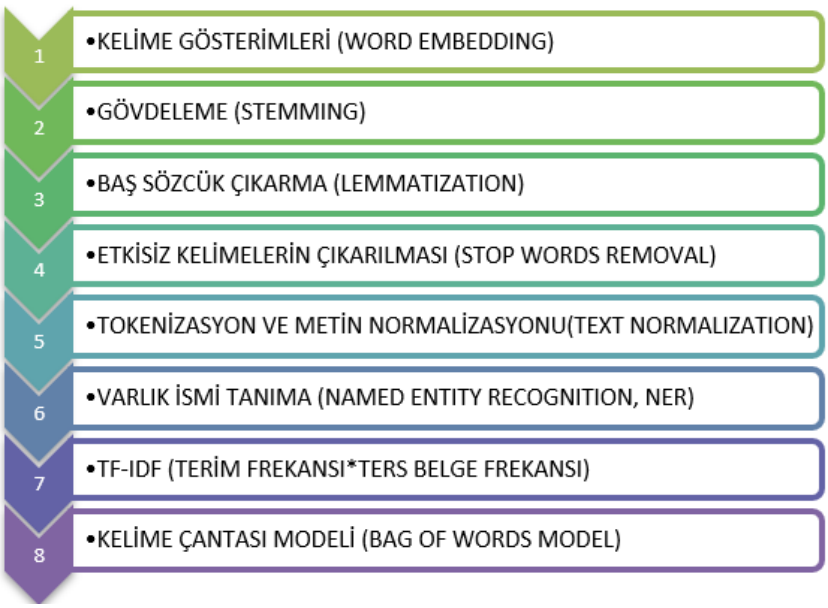
2.3. Metodoloji

Doğal dil ifadelerinden programlama dili kodu üretimi bilgisayar ve yazılım dünyasında sıklıkla karşılaşılan ve çeşitli yöntemler ile çözüme kavuşturulması denenen bir konudur. Zamanla doğal dil işleme ve büyük veri alanındaki gelişmelerin artmasıyla önemli ilerlemeler yaşanmıştır. Teknolojinin tarihsel seyrine paralel olarak yöntemler de değişim göstermiştir. Bu bölümde, doğal dil verileri kullanılarak programlama dili kodu oluşturmak amacıyla geliştirilen yöntemler sınıflandırılmaktadır. Temel olarak bu yöntemler doğal dil işleme teknikleri, istatistiksel ve olasılık tabanlı yöntemler, kural tabanlı yöntemler, makine öğrenmesi ve derin öğrenme kullanan yöntemler olarak kategorilere ayrılabilir.

2.3.1 Doğal dil işleme teknikleri

Doğal dil işleme, geniş bir araştırma alanını kapsar. Makineler, çeşitli dil bilimine dayalı teknikler ve algoritmalar kullanarak herhangi bir veriden bilgi çıkarabilirler. Doğal dil işleme teknikleri, programlama dili oluşturma problemi için tek başına kullanılarak mantıksal çıkarımlar ve sonuçlar üretebileceği gibi istatistiksel yöntemler veya makine öğrenmesi gibi diğer metodlarla birlikte de kullanılabilmektedir. Bu çalışmalara katkı sağlamak amacıyla, doğal dil işleme alanında birçok kütüphane geliştirilmiş ve bu kütüphaneler aracılığıyla işlemler otomatikleştirilmeye çalışılmıştır. Bu bölümde bahsedilen bu tekniklerin neler olduğu açıklanmaktadır.

Şekil 4'te, yaygın olarak kullanılan doğal dil işleme teknikleri gösterilmektedir.



Şekil 4. En Yaygın Kullanılan Doğal Dil İşleme Teknikleri

Kelime Gösterimleri metinsel ifadelerin vektörel olarak temsil edilmesi anlamına gelmektedir. Kelimenin diğer kelimeler ile olan ilişkisine, kullanıldığı bağlama, metinde geçme sıklığı gibi faktörlere bakılarak doğal dil işleme görevlerinde kullanılmaktadır. Kısaca özetlemek gerekirse birbirine benzer anlam ifade eden kelimeler benzer vektörel özelliklerle temsil edilmektedir. (Almeida & Xexéo, 2019).

Gövdeleme (Stemming) işlemi bir kelimenin farklı varyasyonlarının en yalın haline yani kök haline indirgenmesi olarak tanımlanabilir. Bu işlemdeki amaç kelime çeşitliliğini azaltarak bilgisayarın işleyeceği ve anlayacağı bir forma dönüştürmektir. Doğal dil işleme alanında kullanılan metin ön işleme tekniklerinden birisi de baş sözcük çıkarma (lemmatization) yöntemidir. Bir kelimenin çekim eki alan kısımlarının tek bir unsur olarak tanımlanabilecek şekilde bir araya getirilmesi işlemine denir. Yani Lemma, tüm çekimli kısımların temelidir. Baş sözcük çıkarma işlemi Gövdeleme işlemine göre daha karmaşıktır ve daha fazla dil bilgisine ihtiyaç duymaktadır (Siddhartha ve ark., 2021).

Bir metin içerisinde bulunan ancak anlamında farklılık yaratmayan, cümleden çıkarılsa bile anlamını değiştirmeyen kelimelere etkisiz kelimeler denir. Bu tarz kelimelerin veri seti içerisinde bulunması, kullanılan veri setinin büyüklüğüne göre gereksiz yer işgaline neden olmaktadır. Bu nedenle veri ön işleme aşamalarından birisi de etkisiz kelimelerin çıkarılmasıdır. Farklı programlama dilleri için geliştirilen doğal dil işleme kütüphanelerinde bu ön işleme adımını gerçekleştirmek için çeşitli fonksiyonlar tanımlanmıştır. Türkçe dili için “ve”, ”bir”, “de”, ”için”, “halen”, “gibi” vb. kelimeler etkisiz kelimelere örnek olarak verilebilir.

Metinlerden bilgi çıkarmak için kullanılan yöntemlerden birisi de Varlık ismi tanımadır. Belgelerin işlenmesi için kişilere, yerlere, kuruluşlara ve şirketlere atıfta bulunan ifadelerin tanımlanması anlamına gelir. Doğal dil işlemenin temel adımlarından birisidir. İnsanlar için bu işlem basittir çünkü birçok adlandırılmış varlık özel isimdir ve ilk harfleri büyüktür. Bu nedenle tanınması kolaydır ancak makineler için durum bu kadar basit değildir. Adlandırılmış varlıklar özel isimler olduğundan sözlükler oluşturularak sınıflandırmak kolay değildir, zamanla yeni özel isimler eklenmektedir. Bunun yanı sıra bu problem de farklı sorunlar ile karşılaşılmaktadır. Örneğin “Beyaz Saray” ifadesi ne zaman organizasyon olarak kullanılır ne zaman bir konum olarak kullanılır anlamak kolay değildir. Ya da “Nisan” kelimesi ne zaman ay olarak ne zaman

isim olarak kullanılır bilgisayarların anlaması insanların anlaması kadar kolay değildir (Mansouri ve ark., 2008).

Metinlerden bilgi elde etme için önemli bir diğer ön işleme adımı tokenize etme işlemleridir. Tokenize etmek demek metinleri tokenlarına ayırmak anlamına gelir. Kelimeler ya da cümleler olarak tokenize etme işlemi gerçekleştirilebilir. Metinsel verileri işleyebilmek adına doğal dil işleme de sıklıkla kullanılır. Metin normalizasyonunda ise metinsel bir veriyi belirli işlem adımlarından geçirerek standart bir form haline getirmeye çalışılır. Veriyi daha tutarlı ve kullanılabilir hale getirmek, bilgisayarın çalışacağı bilgi miktarını azaltmak dolayısıyla verimliliği artırmak için gerçekleştirilir (Yse, 2021). Önceki kısımlarda bahsedilen gövdeleme, baş sözcük çıkarma ve sonraki kısımlarda bahsedilen etkisiz kelimelerin çıkarılması gibi teknikler metin normalleştirme tekniklerinden bazılarıdır.

TF-IDF değeri bir kelimenin belge koleksiyonunda bulunan bir belge ile ne kadar ilişkili olduğunu, o belge için ne anlam ifade ettiğini belirlemeye yardımcı olan istatistiksel hesaplamalara dayanan bir değerlendirme yöntemidir. En önemli kullanımı otomatik metin analizi olmakla beraber yaygın bir kullanıma sahiptir. Kelimenin belgede bulunma sayısıyla orantılı olarak TF-IDF değeri artmaktadır (Stecanella, 2019).

Kelime çantası modeli bir metin belgesi içerisinden seçilen bölümlerin, kelime frekansı gibi kriterlere göre basitleştirilerek temsil edilmesi için kullanılır. BoW tekniği bilgisayarlı görü, doğal dil işleme, bayesian spam filtreleri, makine öğrenmesi ile belge sınıflandırma gibi çeşitli alanlarda kullanılmaktadır. Modelde kelimelerin sırası değil frekansı önemlidir. Bu nedenle belge ya da metinler kelime torbası gibi düşünülür. Modelde oluşturulan matris formundaki kelimeler arasındaki anlamsal ilişki ve dil bilgisi kuralları göz ardı edilirken çokluk dikkate alınır (Kowsari ve ark., 2019).

2.3.2 İstatistiksel ve olasılık tabanlı yöntemler

Zaman serisi verilerini modelleme, konuşma ve sinyal işleme, konuşma etiketleme, isim-sözcük öbeklemesi, bilgi çıkarımı vb. birbirlerinden farklı yapıda ve zorluktaki görevlerde istatistiksel ve olasılık tabanlı modellerle çözümler gerçekleştirilebilmektedir. Bu yöntemlerde kelimeler arasındaki benzerliklere dayalı olarak verileri analiz edip istatistiksel çıkarımlar ve tahminler yapılmaktadır. İstatistiksel dil modelleri ile doğal dil işleme özellikleri birleştirilerek, birçok görevde başarılı çalışmalar gerçekleştirilmiştir. Bu kapsamda, gizli markov modelleri, n-gramlar, maksimum entropi modelleri ve bayes ağları gibi istatistiksel yöntemler öne çıkmaktadır.

2.3.3 Kural tabanlı yöntemler

Belirli bir alandaki problemleri çözmek, ilgili alana özgü yeni yöntemler geliştirmek için belirlenen kurallar setine ve algoritma adımlarına bağlı kalarak çeşitli kararlar alabilen, bu kararlara göre çeşitli eylemler gerçekleştirebilen yöntemler olarak tanımlanabilir. Doğal dil işleme ve doğal dilden kod oluşturma problemi için kural tabanlı yöntemlere bakıldığında ise ifade edilen metinleri anlamak, öğelerine ayırmak, anlamlandırmak ve bu anlamlı ifadelerden kod üretebilmek için belli modeller ile bir sonuca ulaşmak ya da verileri öğrenmek yerine, işlemleri önceden tanımlanan kurallar ve kodlama yoluyla yönlendirerek çözmeye dayanır. Girdi verisi belirli kurallar kullanılarak işlenir ve amaca yönelik olarak sonuç üretimini gerçekleştirilir. Çoğunlukla anlaşılmaya çalışılan dilin dil bilgisi kuralları dikkate alınarak sonuca ulaşılır. Bu yöntemlerin geliştirildiği alana göre belirli avantaj ve dezavantajları bulunur. Geliştiricilerin kuralları ve kapsamı önceden belirlemesi gerektiğinden daha spesifik amaçlar için kullanılır. Genellikle basit problemler ele alınacaksa, problemin çözümü için oluşturulması gereken kurallar açıkça ifade edilebiliyorsa ve statik bir durum söz konusuysa kural tabanlı bir yöntem geliştirilerek daha az

maliyet ile çözüme ulaşılabilir. Fakat dinamik bir süreç yönetimi ve karmaşık durumlar söz konusu ise kural tabanlı yöntemler kullanılarak sonuca ulaşmak çoğu zaman mümkün olmayabilir. Kurallar dışında oluşan yeni görevlerde başarı oranları düşer, esneklikleri azalır.

2.3.4 Makine öğrenmesi kullanılan yöntemler

İnsanda öğrenme süreci yeni davranışlar, bilgiler, beceriler kazanılması ya da mevcut davranışların değiştirilmesi ile ilgilidir. Makinelerde ise öğrenme süreci veriye dayanmaktadır. Verileri öğrenmek ve sonuç çıkarmak için makine öğrenmesi modellerini ve algoritmalarını kullanan yapay zekâ biliminin alt çalışma alanlarından birisi makine öğrenmesi alanıdır. Makine öğrenmesi, bilgisayarların insan davranışlarına benzer davranışlar taklit etmesini sağlayan çok disiplinli bir alandır. Makine öğrenmesi kullanılarak oluşturulan her etkileşim, gerçekleştirilen her eylem, bilgisayar sistemlerinin öğreneceği ve deneyimleyerek kullanabileceği bir bilgi haline getirilir (Alzubi ve ark., 2018). Makine öğrenmesi yöntemleri, farklı bilgisayar bilimi alanlarında kullanıldığı gibi doğal dili işleme ve anlama alanında da kullanımına sık rastlanılan yöntemlerdendir. Makine öğrenmesi denetimli öğrenme, yarı denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeli öğrenme olarak kategorilere ayrılır.

Denetimli öğrenme sistem eğitiminin etiketlenmiş veri kümeleri kullanımıyla gerçekleştirildiği öğrenme çeşididir. Model etiketlenmiş girdi ve çıktı verileri arasında ilişki kurarak öğrenme işlemini gerçekleştirir. Böylece sistem çıktısını bilmediği veriler için tahminler gerçekleştirebilir (Delua, 2021). Denetimsiz öğrenmede ise bu durumun aksine etiketlenmemiş veri kümeleri kullanılarak eğitim gerçekleştirilir.

Bilimsel çalışmalar için etiketli veri elde etmek maliyetli bir süreçtir. Elimizdeki problemin büyüklüğüne ya da zorluğuna göre her zaman etiketli veri elde etmek mümkün olmayabilir. Etiketli veriye nispeten etiketsiz veri elde etmek daha kolaydır. Pek çok alanda büyük miktarda etiketlenmemiş veri kaynağı mevcuttur ancak bu verileri kullanmanın az yolu vardır. Hem etiketli hem de etiketsiz verilerin bir kombinasyonundan öğrenme anlamına gelen yarı denetimli öğrenme metotlarıyla büyük miktardaki etiketsiz veriyi etiketli veriler ile beraber kullanarak bu sorun aşılmaya çalışılmaktadır (Pise & Kulkarni, 2008).

Takviyeli öğrenme olarak da bilinen pekiştirmeli öğrenmede ise bir ajan (agent), bulunduğu ortamla etkileşim kurarak eylem ve deneyimlerinden elde ettiği geri bildirimler aracılığıyla öğrenme işlemini gerçekleştirir. Hem denetimli öğrenme hem de pekiştirmeli öğrenme türünde girdi çıktı arasında bir haritalama vardır ancak pekiştirmeli öğrenme de farklı olarak ödül ve ceza sistemi uygulanarak hedefe yönelim belirlenir. Yani pekiştirmeli öğrenmede amaç ödül puanını maksimuma çıkararak uygun eylemi belirlemektir (Bhatt, 2018).

2.3.5 Derin öğrenme kullanılan yöntemler

İnsan beyni bilgisayarlara göre büyük avantajlara sahiptir. Örneğin kötü ışıklandırmalı kalabalık bir odada yandan bakınca bile kısa sürede bir yüzü tanıyabiliriz ya da çok gürültülü bir ortamda bir kişinin konuşmasını algılayabiliriz. İnsan beyninin en önemli özelliklerinden birisi öğrenebilmesidir. Beyin, tanıma, algılama, öğrenme gibi eylemleri gerçekleştirebilmek için gereken hesaplamaları aksonlar, sinapslar ve dendritlerden meydana gelen sinirsel yapılar aracılığıyla iletişim kuran nöron ağlarıyla gerçekleştirir (Krogh, 2008). İnsan beynindeki bu nöral ağlardan esinlenerek insana ait bazı özellikleri bilgisayarlara yaptırabilmek için insan beynini modelleyen yapılara Yapay sinir ağları (Artificial neural networks) denir. Bilgisayar dünyasındaki zorlu görevleri gerçekleştirebilmek adına yapay nöron yapıları kullanılarak birçok yapay sinir ağı modeli geliştirilmiştir. Giriş, çıkış katmanı ve gizli katman(lar)dan oluşan yapay sinir ağları bir makine öğrenmesi türüdür. Giriş katmanı veri girişini alır ve bu verileri bilgisayar için anlamlı

hale getirebilmek adına gizli katman(lar)a iletir. Gizli katman(lar)da gerçekleştirilen çeşitli işlemler sonucunda elde edilen veriler dış dünyaya aktarılmak üzere çıkış katmanına iletilir. Basit problemler için temelde bir ya da iki adet gizli katman oluşturmak sonuca ulaşmak için yeterli olurken daha karmaşık problemleri anlayabilmek, örüntüleri tanımak ve çeşitli ilişkiler çıkarabilmek için onlarca hatta yüzlerce gizli katman kullanmak gerekebilir.

Derin öğrenme, çok katmanlı yapay sinir ağlarını kullanan ve doğal dil işleme alanında en sık kullanımına rastlanılan makine öğrenmesi türlerinden biridir. Gerçek dünya problemlerinin çözümü için insan benzeri performans göstermeye oldukça yakın bir makine öğrenmesi türüdür. İstatistiksel yöntemler ve geleneksel makine öğrenmesi yöntemlerinden farklı olarak belirli kurallar kullanarak öğrenmek yerine elde bulunan veriden, verinin büyüklüğüne paralel olarak artan bir başarı oranı ile karmaşık özellikleri otomatik olarak öğrenmektedir. Her katmandaki bilgi önceki katmanlardaki bilgilerden hesaplama yoluyla temsil edilerek verilerdeki karmaşık yapılar keşfedilmeye çalışılır. Yapay sinir ağları kullanılarak birçok derin öğrenme türü geliştirilmiştir. Doğal dil işleme alanında sıklıkla karşılaşılan derin öğrenme yöntemlerine Üretken Çekişmeli Ağlar (Generative Adversarial Networks, GAN), Tekrarlayan Sinir Ağları (Recurrent Neural Network, RNN), Uzun Kısa Süreli Bellek (Long Short Term Memory, LSTM), Geçitli Yinelenen Birim (Gated Recurrent Unit, GRU), Transformer mimarisi örnek verilebilir.

3. ARAŞTIRMA METODU

Bu bölümde, gerçekleştirilen çalışma kapsamında incelenen literatürdeki makaleleri belirleme kriterleri açıklanmaktadır. Literatür araştırması için Google Akademik arama motoru tercih edilmiştir. Aramalar gerçekleştirilirken "code generation", "generating programming language code from natural language", "generating code from natural language" gibi aynı anlama gelen farklı anahtar kelimeler kullanılarak makaleler taranmıştır. Çalışmalar 2000 ila 2022 yılları arasında sınırlandırılmıştır. Taranan makaleler arasından belirlenen çalışmalar, bulundukları veri tabanlarında detaylı bir şekilde incelenmiştir. Aramalar gerçekleştirilirken makale türü, makale dili, herhangi bir doğal dil veya hedeflenen programlama dili için özel ölçüt belirlenmemiştir. Literatür araştırması için seçilen çalışmaların uygunluk kriterleri belirlenirken çalışmaların özetleri okunmuş, özet bilgilerin yetersiz kaldığı durumlarda çalışmaların tamamı okunarak detaylı bir biçimde analiz edilmiştir. Ek olarak, incelenen çalışmaların kullandığı referanslar taranmış ve ilgili olabilecek diğer makaleler değerlendirilmiştir. Gerçekleştirilen bu işlemlerin ardından, belirlenen 32 adet çalışma detaylı bir biçimde analiz edilmiştir.

3.1. Derleme Çalışmasının Amacı

Bu derleme çalışması, aşağıdaki araştırma sorularının yanıtlarını aramaktadır:

Q1: Doğal dilden programlama dili kodu üreten çalışmalarda hangi yaklaşımlar kullanılmaktadır?
Q2: Hangi doğal dil işleme teknikleri bu çalışmalarda kullanılmaktadır?
Q3: Kullanılan veri setleri ve bu veri setlerinin büyüklükleri nelerdir?
Q4: Yaklaşımlarda kullanılan yöntemlerin temel karakteristikleri nelerdir?
Q5: Çalışmaların performansları nasıl değerlendirilmiştir?

Belirlenen bu araştırma sorularına dayalı olarak, kapsamlı taramalar gerçekleştirilmiştir.

4. BULGULAR VE TARTIŞMA

Literatürde doğal dil metinlerinden programlama dili kodu oluşturmak amacıyla geliştirilen 32 adet çalışma bu bölümde karşılaştırmalı olarak incelenmiş ve detaylı bir biçimde analiz edilmiştir. Bu yöntemlerin avantajları, dezavantajları, uygulama alanları ve performansları üzerine odaklanılmış, elde edilen bulgular tablolar halinde sunulmuştur. Ayrıca, bu alanda karşılaşılan zorluklar ele alınmış, çözüm önerileri sunulmuş ve gelecek çalışmalar için öngörüler paylaşılmıştır.

4.1. İncelenen Çalışmalar

Bu bölümde, doğal dilden programlama dili kodu oluşturma problemine yönelik literatür çalışmaları ve ilgili çalışmaların literatüre katkıları ele alınmıştır. İncelenen çalışmalar çeşitli veri setlerini içermekte ve farklı amaçlara hizmet etmektedir. Bu bölümde bahsedilen çalışmalar tarihsel bir sırada detaylı şekilde incelenmiştir.

Pek çok araştırmacı doğal dilden programlama dili kodu üretme üzerine çalışmalar gerçekleştirmiştir. Literatür incelendiğinde yapılan çalışmalarda satır satır kod açıklamasını tüm detaylarıyla alan ve kaynak kod üreten yazılımlara rastlandığı gibi daha soyut ifadeleri algılayan programlar geliştirildiği de gözlemlenmiştir. Detaylı açıklama gerektiren programların spesifik alanlara özgü olduğu ve bu programların yaptığı işlerin belirli kısıtlamalar içerdiği saptanmıştır. Price ve diğerleri, satır satır doğal dil açıklamaları alan ve Java programları oluşturan bir arayüz olan NaturalJava'yı önermişlerdir. NaturalJava, doğal dil açıklamalarından Java kodu üretmek için Sundance, PRISM (The Programming Instruction Synthesis Module), TreeFace olarak adlandırılan 3 alt sisteme sahiptir. Sundance doğal dil açıklamasından programlama terimlerini temsil edebilen vaka çerçeveleri oluşturmak için bilgi çıkarım işlemleri gerçekleştiren doğal dil ayrıştırıcısıdır. PRISM'ın alt sistemi ise vaka çerçevesi yorumlayıcısıdır. Oluşturulacak programın Soyut Söz Dizimi Ağacını(AST) yönetmek için TreeFace adı verilen bir Java AST yöneticisi kullanılmıştır (Price ve ark., 2000). Liu ve Lieberman, tasarımcıların ve kod geliştiricilerinin soyutlama becerilerini artırmalarına yardımcı olması amacıyla İngilizce olarak alınan doğal dil ifadelerini kod olarak görselleştiren bir arayüz olan Metafor'u önermişlerdir. Metafor kullanıcı ile etkileşimli olarak çalışmakta ve Python programlama dilinde bir iskelet program kodu oluşturabilmektedir. MontyLingua (Liu, 2004) dil anlama sistemini kullanarak cümlenin ayrıştırılmasını sağlar. MetaforLingua yapısı ile bilgi temsilini sağlar. Nesneleri ve özel yapıları anlamlandıran Programmatic Interpreter bileşeni kullanılır (H. Liu & Lieberman, 2005). Knöll ve Mezini, 2006 yılında doğal dil tanımından Java kaynak kodu üreten bir araç olan Pegasus'u önermişlerdir. Pegasus bir programın tanımını satır satır alıp tamamen çalıştırılabilir kaynak kod üretmektedir. Pegasus kendisine giriş verisi olarak verilen doğal dil cümlelerinin gramer yapısını inceleyerek if, then gibi mantıksal anlamlı ifadeleri belirleyip mantıksal bir yapı çıkarmaktadır. Pegasus'un sahip olduğu bu bilgiler beyin olarak tanımlanan bellekte muhafaza edilir. Kaynak kod üretimi için Anlam-Kitaplığı (Meaning-Library) adı verilen bir veri tabanı kullanılmaktadır (Knöll & Mezini, 2006). Little ve Miller anahtar kelimeler içeren komutları web ve Microsoft Word'de yürütülebilir koda dönüştüren bir sistem önermişlerdir. Sistemde anahtar kelimelerin varlığına odaklanılır. Bu sistem sayesinde dil yapılarına duyulan ihtiyacın en aza indirgenmesi amaçlanmaktadır. İlgili sistem elde edilen web sayfası veya bir belge dosyası ile doğal dil açıklamasından çalıştırılabilir kod üretir (Little & Miller, 2006).

Somasundaram ve Swaminathan tarafından minimum maliyetle hedeflenen dile dönüştürülmesini kolaylaştırmak için doğal dil ifadelerini ayrıştıran bir sistem olan NLC (Natural Language Compiler) önerilmiştir. Sistem Sözcüksel Çözümleyici (Lexical Analyser), Semantik Çözümleyici (Semantic Analyser), Sözdizimsel Çözümleyici (Syntactic Analyser), Kod Oluşturucu (Code Generator) olmak üzere 4 alt bileşenden oluşur. Tavsiye yolu ile öğrenme stratejisine benzer

şekilde kendi kelime dağarcığıyla eşleştirilerek ve kullanıcı ile etkileşim yoluna girerek öğrenmeyi gerçekleştirir. TPR(True Positive Rate) değeri sistemin ilk kullanım durumlarında düşük iken kelime dağarcığı genişletildikçe bu değerin yükseldiği gözlemlenmektedir (Somasundaram & Swaminathan, 2011). Cozzie ve arkadaşları soyut doğal dil girdilerini alarak java kaynak kodu ve testi üreten Macho adını verdikleri sistemi önermişlerdir. Macho 4 alt sistem içermektedir: Doğal dil ayrıştırıcısı, İngilizce ile Java kodlarını eşleştirmeyi sağlayan veri tabanı, Kod parçalarını birleştiren bir birleştirici ve Aday programı test eden otomatik bir hata ayıklayıcı(automated debugger). Macho soyut dili ayrıştırır ve karşılaşılan belirsizlikten kurtulmak için birden çok aday program üretebilir. Aday programlar oluşturulan test ile kontrol edilir. Kontrol edilen programlar arasından en uygun olan seçilir. Bu uygunluk seçimi Bayes Olasılıksal Modeli kullanılarak belirlenir (Cozzie ve ark., 2011; Cozzie & King, 2012).

Le ve diğerleri, doğal dil açıklamalarından akıllı telefon komut dosyaları sentezleyen etkileşimli bir araç olan SmartSynth'ı önermişlerdir. Sistem, doğal dil ifadelerini kullanarak çeşitli platformlara sahip akıllı telefonlarda programlanabilecek kodlar üretmek için tasarlanmıştır. Çalışmada farklı bileşenleri ve bunların veri akış ilişkilerini tanımlamak için doğal dil işleme teknikleri kullanılırken eksik veri akışını oluşturmak için tür tabanlı program sentezi tercih edilmiştir. SmartSynth giriş açıklamasında bulunan belirsizliği veya bilinmeyen öğeleri çözmek için kullanıcıyla etkileşime girer (Le ve ark., 2013). Manshadi ve diğerleri, 2013 yılında doğal dil tanımlarından kaynak kodu oluşturmak için bir problemi bağımsız alt problemlere ayıran ve bu alt problemlerin her birini ayrı ayrı çözen Versiyon Uzayı Cebiri Fikrini (The Idea of Version Space Algebra) kullanan hibrit bir istatistiksel yöntem sunmuşlardır. Bu yöntemde amaç çözüme giden yolların karmaşıklığını azaltmak ve problemi daha basit problemlere indirgemektir. Olası çözümlerin sayısını azaltmak için temel PbE (Programming by Example) üzerine geliştirilen bir model olan Örnekle Olasılıklı Programlama Yöntemi (Probabilistic Programming by example, PPbE) kullanarak geliştirilen bir model önermişlerdir. Sistem, bir programı diğerine tercih etmek için En Kısa Program Buluşsal Yöntemini (The Shortest Program Heuristic) kullanır (Manshadi ve ark., 2013). Gulwani ve Marron, 2014 yılında elektronik tablo programlama için doğal dil tabanlı bir arayüz aracılığıyla kullanıcı etkileşimini destekleyen bir Excel eklentisi olan NLyze'ı önermişlerdir. NLyze ile doğal dil ifadeleri ve elektronik tablo dosyası alınarak hesaplamalar ve tablo konfigürasyonları gibi Excel işlevlerine dönüştürme gerçekleştirilir. Çalışmada elektronik programlama için kapsamlı Etki Alanına Özgü (DSL, Domain Specific Language) bir dil tasarımı sunulmuştur (Gulwani & Marron, 2014).

Sayısal işlemler yapmak ya da kelimelere dayalı problemler çözmek bilim insanları için genellikle ilgi çekici ve kafa karıştırıcı görülmüştür. Matematiksel problemleri çözmek amacıyla sistemler geliştirmek de zorlu bir araştırma alanıdır. Literatür çalışmaları incelendiğinde araştırmacıların bu gibi karmaşık problemleri bilgisayar programlarına çözdürme istekleriyle karşılaşılır. Shi ve diğerleri 2015 yılında Semantik ayrıştırma ve akıl yürütme yaklaşımı kullanarak matematiksel kelime problemlerini çözmek amacıyla SigmaDolphin adını verdikleri sistemi önermişlerdir. Sistem, matematiksel kelime probleminin tanımını alır ve ilgili kaynak kodunu ve probleminin cevabını üretir. Üretilen kaynak kod, Shi ve diğerleri tarafından tasarlanmış semantik temsil dili olan DOL dili (Dolphin Language) biçimindedir. Araştırmacılar problemi başlangıçta DOL ağacına dönüştürür ve sonrasında DOL ağacından matematik ifadeleri türetirler (Shi ve ark., 2015). Quirk ve diğerleri, sıradan kullanıcılardan toplanan gürültülü eğitim verileri üzerinde anlamsal ayrıştırma yaklaşımlarını karşılaştırmışlardır. Önerdikleri yaklaşım kendilerinden önceki sistemlerin geliştirilmiş halidir. Doğal dil ifadesini IFTTT(If-this-then-that) tarifiyle eşleştiren bir anlamsal ayrıştırıcı sistemi önermişlerdir. Çalışmada karakter ve kelime n-gram özelliklerine sahip bir log-lineer model eğitmişlerdir (Quirk ve ark., 2015).

Doğal dili anlama ile ilgili problemleri ele almak için kullanılan yöntemlerin çoğu yıllar boyunca kural tabanlı yöntemler ve geleneksel makine öğrenme modelleri ile sınırlı kalmıştır. Bu yöntemler

üzerlerinde uzun soluklu ve titiz çalışmalar gerçekleştirilmesi gereken araştırma yöntemleridir. Son yıllarda oldukça popüler olan derin öğrenme modelleri ile doğal dil işleme alanında geleneksel makine öğrenmesi modellerinden daha başarılı sonuçlar elde edildiği ortaya konmuştur. Allamanis ve arkadaşları, doğal dilden kaynak koda ve kaynak koddan doğal dile giden dil modeli oluşturma fikri üzerine her iki yönde haritalama gerçekleştirebilen sinirsel bir dil modeli uygulamışlardır. Elde edilen sonuçlar, doğal dilden kaynak kodu üretmenin, kaynak koddan doğal dil üretmeye göre çok daha zor olduğunu göstermektedir. Bunun nedeni programlama dillerinin katı sözdizimi kurallarından oluşmasıdır (Allamanis ve ark., 2015). Mou ve arkadaşları 2015 yılında uçtan uca programlama yaptıkları çalışmalarını sunmuşlardır. Çalışmada kullanıcılar istedikleri kodu doğal dilde ifade etmekte, önerilen yöntem ile girilen ifadeler karakter karakter okunarak Tekrarlayan Sinir Ağları yardımıyla koda dönüştürülmektedir (Mou ve ark., 2015). Ling ve arkadaşları kaynak kodu üretimi için Latent Predictor Networks (LPN) adını verdikleri bir sinir ağı mimarisi önermişlerdir. Araştırmacılar çalışmalarıyla 2 yeni veri külliyatı (Python'da HearthStone ve Java'da Magic the Gathering) oluşturmuşlardır (Ling ve ark., 2016). Desai ve diğerleri, 2016 yılında doğal dil açıklamasını alan ve karşılık gelen kaynak kodunu sentezleyebilen bir sistem önermişlerdir. Eğitim veri seti, alana özgü bir dilin kod ve açıklama çiftinden oluşmaktadır. Girilen veri seti ile sistem istenen dili öğrenir ve programın tanımıyla eşleşen bir kod parçacığı oluşturur. Sistem, sistemin dilden bağımsız olmasını sağlayan giriş veri kümesinden herhangi bir dili öğrenebilir (Desai ve ark., 2016).

Dong ve Lapata, girdi verilerini vektör temsiline dönüştüren ve bu temsilleri kullanarak çıktı dizileri ve ağaç yapılarını koşullandırarak mantıksal biçimlerini üreten, doğal dil girdisini haritalayan Dikkati Artırılmış Kodlayıcı-Kod Çözücü (Attention Enhanced Encoder-Decoder) modeline dayanan bir yöntem önermişlerdir (Dong & Lapata, 2016). Raghothaman ve arkadaşları Bing arama motorundan gelen tıklanma verilerini kullanarak kullanıcı sorgularını cevaplamak için, açık kaynaklı kod havuzlarından öğrenilen kod kalıpları yardımıyla kod sentezleyebilen bir araç olan SWIM'i önermişlerdir. SWIM, İngilizce ifadelere karşılık gelen API'yi öneren API eşleyicisi ve önerilen API'leri kullanarak kod oluşturan sentezleyiciden oluşur (Raghothaman ve ark., 2016). Nguyen ve arkadaşları 2016 yılında sundukları çalışma ile bir programlama görevini oluşturmak için görevin İngilizce tanımını alan ve eğitim verilerinden öğrenerek API kullanımı için şablon sentezleyen T2API adını verdikleri istatistiksel makine çevirisi tabanlı bir araç önermişlerdir. T2API 2 adımda çalışmasını gerçekleştirir. İlk olarak açıklama-kod verilerinden istatistiksel olarak öğrenir ve görevin ilgili olduğu API'yi üretir. İkinci aşamada grafik tabanlı bir dil modeliyle kod topluluğundan API kullanımlarını öğrenen bir sentez algoritması ile API kullanımını birleştirir (Nguyen ve ark., 2016).

Mandal ve Naskar, 2017 yılında yaptıkları çalışmada doğal dilde Matematiksel Kelime Problemi (Mathematical Word Problem, MWP) metinlerini alan ve bilgi çıkarımı yaparak bu problemleri çözen Java programları üreten bir sistem önermişlerdir. Önerilen sistemde bilgi çıkarımı yapıldıktan sonra bu bilgiler önceden tanımlı olan şablonlarda depolanır. Mandal ve Naskar bilgi çıkarım işlemi için OIA üçlüsü (Owner-Item-Attribute) adı verilen bir şablon tasarlamışlardır (Mandal & Naskar, 2017). Yin ve Neubig, 2017 yılında doğal dil açıklamalarında Python programlama dili komutları oluşturmak için dilin gramer yapısını dikkate alarak olasılıksal dil bilgisi modeli kullanan yeni bir sinir mimarisi modellemişlerdir. Araştırmacılar çalışmalarının anlamsal ayrıştırma kullanan diğer kod oluşturma yaklaşımlarına göre daha iyi performans gösterdiğini belirtmişlerdir (Yin & Neubig, 2017). Lin ve arkadaşları doğal dil girdilerinden karmaşık dosya sistemi işlemlerini gerçekleştirebilmek için gereken bash komutları üretebilmek amacıyla Tellina'yı önermişlerdir. Tellina'nın sistemi geliştirilirken doğal dil işleme tekniklerinden biri olan tekrarlayan sinir ağları (Recurrent Neural Networks, RNN) ve anlamsal ayrıştırma kullanılmıştır. Doğal dil verisi okunduğunda detayları anlayabilmek için varlık tanıma olarak adlandırılan anlamsal ayrıştırma tekniği kullanılmıştır. Sonrasında doğal dil verilerini program şablonuna dönüştürebilmek için RNN kullanılmıştır. Sistemin ilk 3 önerisinden birisinin

doğru olma olasılığı %80'dir. Araştırmacılar programcıların performansını ölçen bir araştırma da sunmuşlardır. Bu çalışmaya göre programcılar Tellina'dan her zaman tam doğru komut önerisi alamasalar bile Tellina'nın çalışma sürelerini %21,7 oranında kısalttığı gözlemlenmiştir (Ernst, 2017; Lin ve ark., 2017). Zhong ve çalışma arkadaşları doğal dil sorgularına karşılık gelen SQL sorgularını üretebilmek için pekiştirmeli öğrenme kullanarak derin bir sinir ağı olan Seq2SQL modelini önermişlerdir. Ağ optimizasyonunu sağlamak için çapraz entropi kaybı ve öğrenmek için ödül ağırlıkları kullanılmıştır. Araştırmacılar çalışmada kendi derledikleri WikiSQL üzerinde deney yapmışlardır (Zhong ve ark., 2017).

Zhang ve Kim isimli araştırmacılar, 2018 yılında CODEnn(Code-Description Embedding Neural Network) adını verdikleri yeni bir derin ağı sistemini sunmuşlardır. Çalışmada CODEnn doğal dil açıklamaları ve karşılık gelebilecek kod parçacıkları arasındaki benzerlikleri yakalamak için eğitilmiştir. Doğal dil sorguları ve kod parçaları heterojendir ve kolayca eşleştirilemez. Önerilen bu sinir ağı ile bahsedilen veriler vektör uzayına gömülerek vektör benzerlikleri ölçülerek eşleştirilmişlerdir. İlgili çalışmayla CODEnn'den yararlanan DeepCS adını verdikleri yine derin öğrenme tabanlı bir kod arama aracını da sunmuşlardır (Gu ve ark., 2018). Liu ve Wu sundukları çalışmayla farklı yazılım mühendisliği amaçları için geliştirilen PiE, Natural Shell ve EasyACL olmak üzere 3 programlama aracını incelemişlerdir. PiE(Programming in Eliza) grafik çizimi için program sentezleyen bir eğitimsel programlama aracıdır. PiE Eliza ile kullanıcılar arasındaki konuşmalardan grafikler çizmek için LOGO programlama dilinde program sentezler. Sistem Eliza, PiE betiği ve LOGO olmak üzere 3 kısımdan oluşur. Eliza ve LOGO arasında bağlayıcılık görevi gören PiE betiği bulunur. Bu betik doğal dil açıklamalarını işler ve LOGO modülü tarafından yürütülecek olan LOGO programlama dilindeki programları sentezler. Natural Shell çeşitli platformlarda komut dosyası oluşturmaya yardımcı olan bir araçtır. Command Box, Uni-Shell Command Box ve Execution Result Box olmak üzere 3 kısımdan oluşur. Natural Shell basit komutlardan oluştuğu için acemi kullanıcıların çalışmalarını kolaylaştırmaktadır. EasyACL ise hem Cisco hem de Juniper sistemleri için erişim kontrol listesi oluşturmak ve doğrulamak için çalışır. Örneğin ağ üzerinde bir bağlantı hatası bulunması durumunda gereksinimler doğal dilde tanımlanabilmektedir (Liu & Wu, 2018).

Schlegel ve çalışma arkadaşları, 2019 yılında Vajra adını verdikleri sistem ile doğal dil açıklamalarını alan ve bunlara karşılık gelen Python kod parçacıklarını üreten çalışmayı sunmuşlardır. Bu çalışmada Anlamsal Ayrıştırma (Semantic Parsing) tekniği kullanılmıştır. Sistem olası ifadelerin bir listesini ve anlamsal olarak birbirlerine en çok benzeyen ilişkili parametreleri oluşturur. Sistem kullanıcı ile etkileşim halindedir. Akışta oluşabilecek belirsizlikleri çözmek amacıyla birden fazla örnek prosedür arasından seçim yapmaktadır (Schlegel ve ark., 2019). Phan, doğal dilde yazılmış metot adı ya da metot açıklamalarından metot gövdeleri oluşturan bir yaklaşım olan MethodInfoToCode'u önermiştir. Çalışmada metinsel bir açıklamadan kod tahmini için Cümle Tabanlı İstatistiksel Makine Çevirisi Modeli (Phrase-based Statistical Machine Translation) ve büyük bir veri seti kullanılmıştır (Phan, 2019). Agashe ve arkadaşları 2019 yılında gerçekleştirdikleri çalışmayla JuICe adını verdikleri geniş bir veri külliyatı sunmuşlardır. Kendi kategorisindeki diğer veri kümelerine bakıldığında JuICe insanlar aracılığı ile derlenmesi ve çok fazla eğitim verisi sunması ile ön plana çıkmaktadır. Çalışmacılar sundukları bu veri kümesi ile modellerini eğitmişlerdir. En iyi performansı LSTM modelinin gösterdiğini deneyimlemişlerdir (Agashe ve ark., 2019). Shin ve arkadaşları bir corpustan öğrenilen kodlama kalıplarını bir nöral sentezleyicinin sözlüğüne dahil eden, bu kod deyimlerini araştırmak için Bayesçi çıkarım kullanan, AST eğitimi için deterministik olmayan bir eğitim rejimi kullanarak program üreten bir sistem olan PATOIS'u önermişlerdir. PATOIS'u 2 semantik ayrıştırma veri kümesi üzerinde değerlendirmişlerdir (Shin ve ark., 2019). Zhu ve arkadaşları doğal dilden programlama dili kodu üretimi için GAN(Generative Adversarial Networks) tabanlı bir otomatik programlama yaklaşımı olan GANCoder'ı önermişlerdir. GANCoder'ın yapısında bulunan kodlayıcı doğal dil ifadelerini anlamsal vektörler olarak kodlar. Bu kodlamanın ardından

kod çözücü dilin gramer yapısını dikkate alarak bu kodları AST ağaçlarına dönüştürür. Oluşturulan AST ağaçları da program kodlarına dönüştürülür. Doğal dili kodlamak için çift yönlü LSTM, AST'yi oluşturmak için normal bir LSTM yapısı kullanılır (Zhu ve ark., 2019).

Gemmell ve arkadaşları, 2020 yılında yayınladıkları çalışma ile kod oluşturma görevleri için Transformer mimarilerini incelemiştir. Geri bildirim mekanizması aracılığı ile güçlendirilmiş derin öğrenme modellerinden biri olan Transformer modeli üzerinde iyileştirmeler yaparak Relevance Transformer modelini önermişlerdir (Gemmell ve ark., 2020). Perez ve arkadaşları yayınladıkları çalışmada LSTM, RNN, GRU da dahil olmak üzere çeşitli yapıları incelenmişlerdir. Sonuç olarak Transformer mimarisine dayanan bir doğal dil işleme modeli olan GPT-2'ye bazı parametre ayarları yapmışlar ve diğer modellerden daha başarılı olduğunu gözlemlemişlerdir (Perez ve ark., 2021). Hong ve arkadaşları, 2021 yılında hedeflenen dilde program sentezleme için nöral sentezleyici olan Gizli Programcı (Latent Programmer, LP)'yı önermişlerdir. Önerilen sistem 3 temel bileşenden oluşur: Gizli programcı (Latent predictor,), Gizli program kod çözücü(Latent program decoder) ve Program kodlayıcı(Program encoder). Sentezleyici önce istenen programı tanımlayıcı bir plan oluşturur (üst düzey tanımlayıcı bir simge dizisi) ve ardından bu planı uygulayarak hedeflenen dilde program sentezini gerçekleştirir (Hong ve ark., 2021).

Nizzad ve Thelijjagoda önceden eğitilen bir model kullanarak mikrofon aracılığıyla sesli komutları alan ve alınan sesi metne çevirerek elde edilen doğal dil verilerinden python kaynak kodu üreten bir sistem olan PyVoice IDE'yi önermişlerdir. Modelde öğrenmeyi gerçekleştirmek için GPT (Generative Pretrained Transformer) modeline dayanan transfer öğrenimini kullanmışlardır. Çalışma sonuçlarına göre bir diziyi başka bir diziye eşlemek için Transformer mimarisinin daha güçlü olduğunu doğrulamışlardır (Nizzad & Thelijjagoda, 2022). Yüksel ve Karabıyık, 2022 yılında doğal dil işleme yöntemleri aracılığıyla metin verisine karşılık gelen SQL kodunu tahmin edebilen bir sistem önermişlerdir. Araştırmacılar çalışmalarında makine çevirisi yöntemini kullanarak birden fazla dili destekleyen (33 tane doğal dilden İngilizceye çeviri bulunmaktadır.) bir sistem tasarlamışlardır. Kullandıkları veri setleri İngilizce olmakla beraber makine çevirisi yardımıyla farklı doğal dillerden kod üretme işlemini gerçekleştirmişlerdir (Yüksel & Karabıyık, 2022). Zhao ve arkadaşları GAP-Gen olarak adlandırdıkları Python dilinin sözdizimsel ve anlamsal kısıtlamaları tarafından yönlendirilen, doğal dil tanımından otomatik Python kaynak kodu oluşturma yöntemini sunmuşlardır. Çalışmanın ön eğitim ile kaynak girdisinden doğrudan kod üreten diğer yöntemlerden temel farkı, kaynak kodun sözdizimsel bilgilerini daha iyi öğrenmek ve kullanmak için öncelikle kod yapısını bir ara durum olarak üretmesi ve sonrasında bu yapıyı kullanarak kod üretimini gerçekleştirmesidir (Zhao ve ark., 2022).

4.2. Kullanılan Yöntemlerin Analizi

İncelenen çalışmalarda farklı yöntemler ve modeller kullanılmıştır. Bazı araştırmacılar kendi veri setlerini oluştururken, bazıları daha önce çalışan araştırmacılar tarafından oluşturulmuş veri setlerini kullanmışlardır. Bazı çalışmalarda ise veri seti bilgileri bulunmamaktadır. Çalışmalar farklı problemlere çözüm üretmek amacıyla gerçekleştirilmiştir. Birebir aynı amaçlarda ve eşit kriterlerde gerçekleştirilmediğinden, çalışmaları doğrudan karşılaştırmak uygun bir yaklaşım değildir. Bu kısımda, Tablo 1 aracılığıyla çalışmalarda kullanılan yöntemler, kaynak ve hedef programlama dili gibi genel bilgiler verilmektedir.

Tablo 1. Çalışmalarda Kullanılan Yöntem ve Teknikler

Çalışma	**Yıl**	**Kullanılan/Önerilen Teknikler**	**Kaynak Dil (Doğal Dil)**	**Hedef Dil (Programlama Dili)**
(Nizzad & Thelijjagoda, 2022)	2022	• Transformer Mimarisi (Transformer Architecture)	İngilizce	• Python
(Yüksel & Karabıyık, 2022)	2022	• Yarı otoregresif bir model olan SmBop (Semi-auto-regressive Bottom-up Semantic Parsing) • Uzun Kısa Süreli Bellek (LSTM)	Türkçe Birden fazla dil desteği	• Sql
(Zhao ve ark., 2022)	2022	• Syntax-Flow Dil Modeli • Variable-Flow Dil Modeli • Yönlendirilmiş Kod Üretimi Dil Modeli (Guided Code Generation Language Model) • T5-English Dil Modeli • CodeT5 Dil Modeli	İngilizce	• Python
(Perez ve ark., 2021)	2021	• Uzun Kısa Süreli Bellek (LSTM) • Sinirsel Kelime Çantası (Neural-Bag-Of-Words) • GPT-2 (Generative Pre-trained Transformer 2)	İngilizce	• Python
(Hong ve ark., 2021)	2021	• İki Seviyeli Işın Araması (Two-level beam search) • Ayrık Otomatik Kodlayıcılar (Discrete autoencoders)	Giriş/Çıkış çifti	• Python
(Gemmell ve ark., 2020)	2020	• Geri alma algoritması (Retrieval algorithm • Transformer Mimarisi (Transformer Architecture) • Relevance Transformer Model	İngilizce	• Python
(Phan, 2019)	2019	• Cümle Tabanlı İstatistiksel Makine Çevirisi Modeli	İngilizce	• Java

(Agashe ve ark., 2019)	2019	• Çift yönlü LSTM (Bidirectional LSTM, BiLSTM) • Uzun Kısa Süreli Bellek (LSTM) • Transformer Mimarisi (Transformer Architecture)	İngilizce	• Python
(Shin ve ark., 2019)	2019	• Çift yönlü LSTM (Bidirectional LSTM, BiLSTM) • Bayesçi Çıkarım • Soyut Sözdizimi Ağacı (Abstract Syntax Tree, AST) eğitimi için yeni deterministik olmayan bir eğitim rejimi	İngilizce	• Python • Sql
(Schlegel ve ark., 2019)	2019	• Destek Vektör Makinesi (Support Vector Machine, SVM) • Anlamsal Ayrıştırma (Semantic Parsing)	İngilizce	• Python
(Zhu ve ark., 2019)	2019	• Tek/Çift yönlü LSTM • Işın Arama algoritması (Beam Search algorithm)	İngilizce	• Python • Prolog • Mantıksal Formlar(λ hesabı şeklinde)
(Gu ve ark., 2018)	2018	• Çift Yönlü LSTM • Heterojen verilerin ortak gömülmesi tekniği (Joint Embedding of Heterogeneous Data)	İngilizce	• Java
(Zhong ve ark., 2017)	2017	• Kural Tabanlı Takviyeli Öğrenme (Policy-based reinforcement learning) • Uzun Kısa Süreli Bellek (LSTM)	İngilizce	• SQL
(Yin & Neubig, 2017)	2017	• Soyut Sözdizimi Ağacı (Abstract Syntax Tree, AST) • Uzun Kısa Süreli Bellek (LSTM) • Olasılıksal dil bilgisi kullanan yeni bir sinir mimarisi önermişlerdir.	İngilizce	• Python • IFTTT

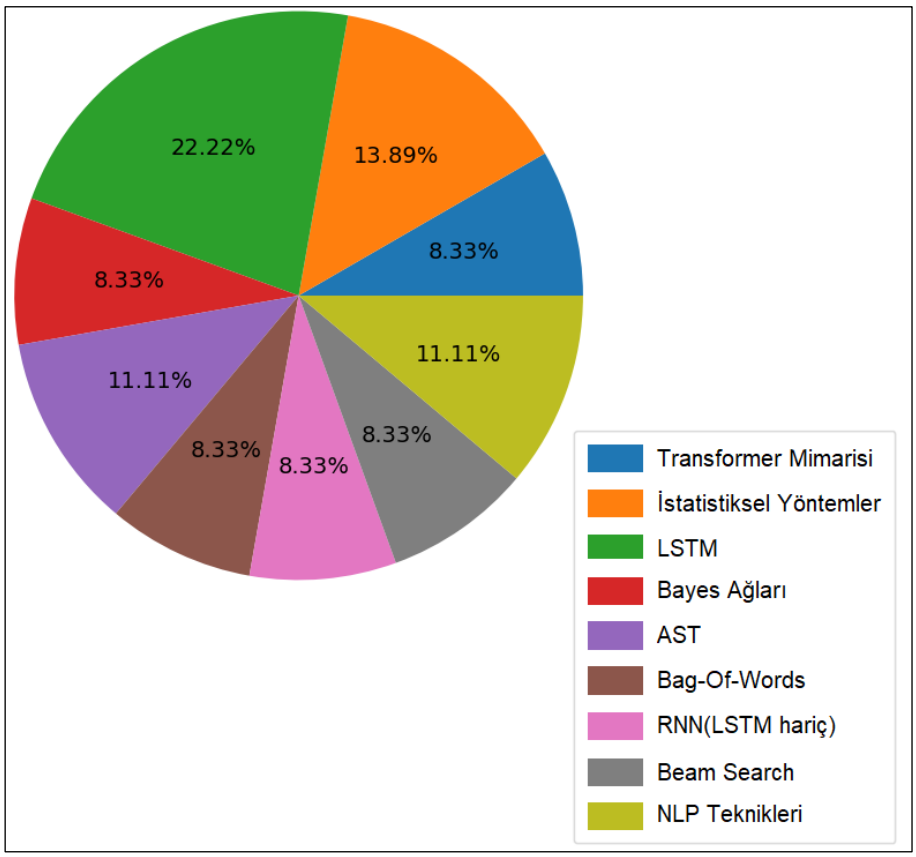
(Ernst, 2017; Lin ve ark., 2017)	2017	• Çift yönlü RNN (Bidirectional RNN) • K En Yakın Komşu (K-Nearest Neighbors, KNN) • Kararlı Eşleştirme Algoritması (Stable Matching Algorithm)	İngilizce	• Bash komutları
(Mandal & Naskar, 2017)	2017	• SRL (Semantic Role Labelling) • Şablon tabanlı bir IE (Information Extraction) yaklaşımı OIA (Owner-Item-Attribute) üçlüsü kullanılmıştır.	İngilizce	• Java
(Raghothaman ve ark., 2016)	2016	• Beklenti maksimizasyon algoritması (Expectation maximization algorithm) • Kosinüs benzerliği fonksiyonu • İstatistiksel Yöntemler	İngilizce	• C#
(Nguyen ve ark., 2016)	2016	• İstatistiksel makine çevirisi (Statistical Machine Translation, SMT) • İngilizce kelimelerin API dizileriyle ilişkilendirilmesi amacıyla IBM (Brown ve ark., 1993) modelinden yararlanılır. • Grafik tabanlı bir dil modeli olan GraLan (A. T. Nguyen & T. N. Nguyen, 2015) kullanılmıştır. • Sıralama sorunu çözümü için bir grafik sentez algoritması geliştirilmiş ve grafik sentezlemek için Işın Arama Stratejisi(A Beam Search Strategy) kullanılmıştır.	İngilizce	• API kullanım şablonu (Eclipse IDE eklentisi)
(Ling ve ark., 2016)	2016	• C2W Modeli • Dikkat Modeli (Attention Model) • RNN tabanlı kod oluşturma modeli olan Latent Predictor Networks (LPN) önerilmiştir.	İngilizce	• Python • Java

(Desai ve ark., 2016)	2016	• Çanta Algoritması (A bag algorithm) • Naive Bayes Sınıflandırıcı (Naive Bayesian Classifier) • Özellik çıkarma (Feature extraction for ranking) • POS etiketleme ve diğer doğal dil özelliklerini çıkarmak için Stanford NLP Motoru kullanılmıştır.	İngilizce	• Belirli bir DSL
(Dong & Lapata, 2016)	2016	• Uzun Kısa Süreli Bellek (LSTM)	İngilizce	• Prolog • IFTTT • Mantıksal Formlar (λ hesabı şeklinde)
(Shi ve ark., 2015)	2015	• CFG (Context-Free Grammar) • Öğrenme temelli istatistiksel bir yöntem (KAZB) • Benzerlik hesabıyla çalışan İstatistiksel bir yöntem (BasicSim)	İngilizce	• DOL-Matematiksel İfade
(Allamanis ve ark., 2015)	2015	• Kelime Çantası (Bag of words, BoW) • Ayrıştırma Ağacı (Parse Tree) • Log-bilinear model adı verilen bir sinirsel dil modeli • Çarpımsal Model (Multiplicative model) • Toplamsal Model (Additive model)	İngilizce	• C#
(Gulwani & Marron, 2014)	2014	• Anahtar Kelime Programlama ve Semantik Ayrıştırma fikri birleştirilerek dinamik programlama tabanlı bir yaklaşım ve sıralama şeması kullanılır.	İngilizce	• DSL

(Le ve ark., 2013)	2013	• Ağaç-tabanlı NLP Teknikleri • Kelime Çantası (Bag of words, BoW) • Düzenli İfadeler (Regular expressions) • İfade uzunluğu ölçümü (Phrase length measuring) • Noktalama işareti algılama (Punctuation detection) • Ayrıştırma Ağacı (Parse Tree) • Tip tabanlı sentez algoritması (Type-based synthesis algoritması) • Reverse Parsing benzeri bir algoritma • Kural tabanlı ilişki algılama algoritması (Rule-Based relation detection algorithm)	İngilizce	• SmartScript
(Manshadi ve ark., 2013)	2013	• Denetimli öğrenme yöntemi • Viterbi Algoritması • En kısa program sezgiselliği (The shortest program heuristic) • Versiyon Uzayı Cebiri tekniği (Technique of version space algebra) • Örnekle olasılıklı programlama yöntemi (The probabilistic PbE, PPbE)	İngilizce	-
(Cozzie ve ark., 2011; Cozzie & King, 2012)	2011, 2012	• Jeneratif Naive Bayes modeli (Generative Naive Bayes model)	İngilizce	• Java • Java test kodu
(Somasundaram & Swaminathan, 2011)	2011	• Tavsiye yoluyla öğrenme • Kural tabanlı bir sistem	İngilizce	• Ara temsil
(Knöll & Mezini, 2006)	2006	• Mantıksal bir yapı çıkarımı ve karşılığında eşleştirme yapılır.	Almanca, İngilizce	• Java
(Little & Miller, 2006)	2006	• Kendi geliştirdikleri çeviri algoritması ve ağaç yapısı kullanılır.	İngilizce	• Visual Basic • Chickenfoot
(H. Liu & Lieberman, 2005)	2005	• Doğal dil işleme teknikleri	İngilizce	• Python

(Price ve ark., 2000)	2000	• Soyut Sözdizimi Ağacı (Abstract Syntax Tree, AST) • Karar Ağacı (Decision tree) • Bilgi Çıkarma(Information extraction, IE) Teknolojisi	İngilizce	• Java

Şekil 5'te incelen çalışmalarda kullanılan yöntemler grafik olarak gösterilmektedir.



Şekil 5. İncelen Çalışmalarda Sık Rastlanılan Yöntemler

Şekil 5'teki grafik incelendiğinde çalışmalarda LSTM yönteminin sıklıkla kullanıldığı görülmektedir. Ancak bu grafik yorumlanırken kullanım sıklığına bakılarak yöntemlerin başarısını doğrudan değerlendirmek uygun bir yaklaşım olmaz. Örneğin incelenen çalışmalarda NLP tekniklerinin kullanımının %11,11 oranında olduğu görülmektedir. Ancak diğer yaklaşımlarda ön işleme adımları vb. aşamalarda NLP teknikleri kullanılabilmektedir. Bunun yanı sıra bazı çalışmalarda grafikte gösterilen yaklaşımların birkaçı birden kullanılmıştır.

Yapılan analizler neticesinde programlama dili kodu üretimi için literatürdeki çalışmalar tarihsel olarak değerlendirildiğinde geçmiş tarihli çalışmaların kural tabanlı ya da istatistiksel tabanlı modeller üzerine yoğunlaştığı, günümüze yaklaşıldıkça daha çok veriye dayalı makine öğrenmesi, derin öğrenme temelli çalışmalar ya da farklı yöntemlerin güçlü yanları birleştirilerek daha başarılı çıktılar oluşturan hibrit çalışmaların arttığı görülmektedir. Kullanılan yöntemlerin birbirlerine göre karşılaştırılabilir avantaj ve dezavantajlar barındırmaktadır.

Kural tabanlı yöntemler insanlar tarafından anlaşılması ve oluşturulması daha kolay olduğu için avantaj barındırırken spesifik alanlara çözüm üretebilmesi dezavantaj oluşturmaktadır. Ayrıca sınırlı veri ile çözüm yöntemi geliştirme imkânı sunması da avantajları arasındadır. Ancak

karmaşık ve dinamik süreç yönetimi söz konusu olduğunda esnek bir yapı sunulamamakta böylelikle yeterli başarı elde edilememektedir. Olasılık tabanlı yöntemler düşünüldüğünde istatistik ve olasılık alanında teorik temeller ve güçlü matematik alt yapısı gerektirmektedir. Derin öğrenme modellerinin ihtiyaç duyduğu ölçüde veri gerektirmese de homojen ve büyük ölçekli veri kümelerine ve hesaplama gücüne ihtiyaç duyulmaktadır. Belirsiz durumlarla başa çıkarken olasılıksal hesaplamalardan yola çıkıldığı için başarılı çıktılar üretilmektedir. Makine öğrenmesi ve derin öğrenme yöntemlerine bakıldığında esnek, dinamik ve genelleştirilebilir yapılar ortaya konabilmektedir. Ancak başarılı sonuçlar elde edebilmek için büyük miktarda ve gürültüsüz veri kullanmak gerekir. Verilerin eksik ya da yanlış olması modelin performansını doğrudan etkilemektedir. Ayrıca bu modellerin oluşturulması güçlü bilgisayarlar ve zaman gerektirmektedir.

Geliştirilen yeni yöntemler ile önceki yöntemlerin tamamen kullanımdan kalkması gibi bir durum söz konusu olmamakla beraber yeni yöntemlerden elde edilen başarı oranları da göz ardı edilmemelidir. Özetle gerçekleştirilecek çalışma için seçilecek yöntemler problem uzayına uygun şekilde tercih edilmelidir. Örneğin incelenen çalışmalarda RNN sık karşılaşılan yöntemler arasındadır. Ancak RNN mimarileri uzun vadeli bağımlılıkları yakalamakta zorlanmaktadır. RNN'deki bu eksiklik özel bir RNN türü olan LSTM yöntemi geliştirilerek çözülmüştür. LSTM ağları uzun veri girişlerini ezberler, önceki ve on anki verileri kullanarak ileriye dönük çıktılar oluşturabilir. Bu uzun veri girişi gerektiren senaryolar için bir avantaj oluşturmakla beraber RNN yapısına göre ek kapılar barındırdığından hesaplama gücü olarak maliyetli olmaktadır. Bu nedenle tek satırlık girdiler için kod çıktısı oluşturulması istenen senaryolarda geleneksel RNN kullanılarak çözüm geliştirilmesi daha uygun olacaktır. Yine Transformer mimarileri son yıllarda adını sıkça duyduğumuz yüksek başarılı sonuçlar veren yapılardır ancak büyük miktarda veri, hesaplama gücü ve zaman gerektirdiğinden oluşturulması maliyetlidir. Bu nedenle daha basit yapılar ile çözüme kavuşturulacak senaryolarda tercih edilmemelidir. Kural tabanlı yaklaşımlar, olasılıksal yöntemler ve derin öğrenme teknolojileri birlikte kullanılarak karma yöntemler oluşturulması da bir seçenektir. Böylece daha az veri ile daha yüksek başarılı çıktılar elde edilebilir. Ancak bu tarz karma yöntemler ile mimarilerin güçlü yanları birleştirilirken dezavantajları da bir araya getirilebilir. Ayrıca bu yapıları kurmak tüm yöntemlerin zorluklarını barındırdığından daha karmaşıktır.

4.2. Veri Seti Analizleri

Çalışmanın bu bölümünde araştırmacıların projelerinde girdi olarak kullandıkları veri setlerinin hangileri olduğu ve özellikleri tablo halinde sunulmuştur.

Tablo 2. Çalışmalarda Kullanılan Veri Setleri

Çalışma	Veri seti adı	Veri seti detayları
(Nizzad & Thelijjagoda, 2022)	• Github CodeSearch Net Challenge	Makine öğrenmesi ve doğal dil işleme için büyük miktarda veri içeren bir veri kümesi.
(Yüksel & Karabıyık, 2022)	• Spider	10.181 adet doğal dil sorusu ve 5.693 tane farklı sql ifadesi içeren bir veri seti.

(Zhao ve ark., 2022)	• Code Search Net(CSN)2 • Edinburgh Code-to-Docstring dataset (CDC)3 • CodeSearchNet AdvTest (Adv)4	**Code Search Net:** Public Github depolarından toplanan bir veri külliyatıdır (Husain ve ark., 2019). Gerekli veri filtreleme adımlarından sonra 412k eğitim verisi, 22k doğrulama verisi ve 22k test verisi elde edilmiştir. **Edinburgh Code-to-Docstring dataset:** Github sitesinden toplanan ve işlenen Python külliyattır (Barone & Sennrich, 2017). 109.108 eğitim verisi, 2000 doğrulama verisi ve 2000 test verisi olacak şekilde eğitim/doğrulama/test verilerine bölünmüştür. **CodeSearchNet AdvTest:** CodeSearchNet külliyatından türetilen bir Python veri kümesidir (Lu ve ark., 2021). CodeSearchNet Advtest veri seti 251.820 eğitim verisi, 9.640 doğrulama verisi ve 19.210 test verisi içermektedir.
(Perez ve ark., 2021)	• CodeSearchNet	Python, Php, Java, Go, Ruby gibi çeşitli dillerde 2 milyon kod ve açıklama çiftinden oluşan CodeSearchNet (Husain ve ark., 2019) veri seti kullanılmıştır. Ancak kullanım Python dili verileri ile sınırlandırılmıştır.
(Hong ve ark., 2021)	• Github verileri	Github sitesinden toplanan bir belge dizisi ve karşılık gelen python kod parçacıklarından oluşan 111.000 python örneği içeren veri setidir (Wan ve ark., 2018).
(Gemmell ve ark., 2020)	• DJANGO • Hearthstone • CoNaLa	**DJANGO:** Tüm veri setleri İngilizce açıklamalardan oluşmak üzere 18 bin satırın üzerinde kaynak kodu ve satır satır açıklamalarını içeren bir veri setidir. **Hearthstone:** Oyun kartlarından elde edilen 665 adet Python kodu ve İngilizce açıklamasından oluşan bir veri kümesidir. **CoNaLa:** StackOverflow soruları ve onlara verilen yanıtlardan elde edilmiş olan 2 bin soru-cevap çiftinden oluşan veri setidir.
(Phan, 2019)	• Github Corpusu	JDK, Android, GWT, Joda-Time, Hibernate ve Xstream'den oluşan 6 popüler kitaplığı kullanan bir Github Projesi verilerinden 2,86 milyon metot çağrısı kullanılmıştır.
(Agashe ve ark., 2019)	• Eğitim: Jupyter notebook • Değerlendirme: nbgrader	Modelleri eğitmek için Github sitesi üzerinde public 659.000'den fazla Jupyter notebook'u kullanılmıştır. Modelleri değerlendirmek için 13.905 nbgrader programlama egzersizleri ve 3.725 çözüm çiftinden oluşmaktadır

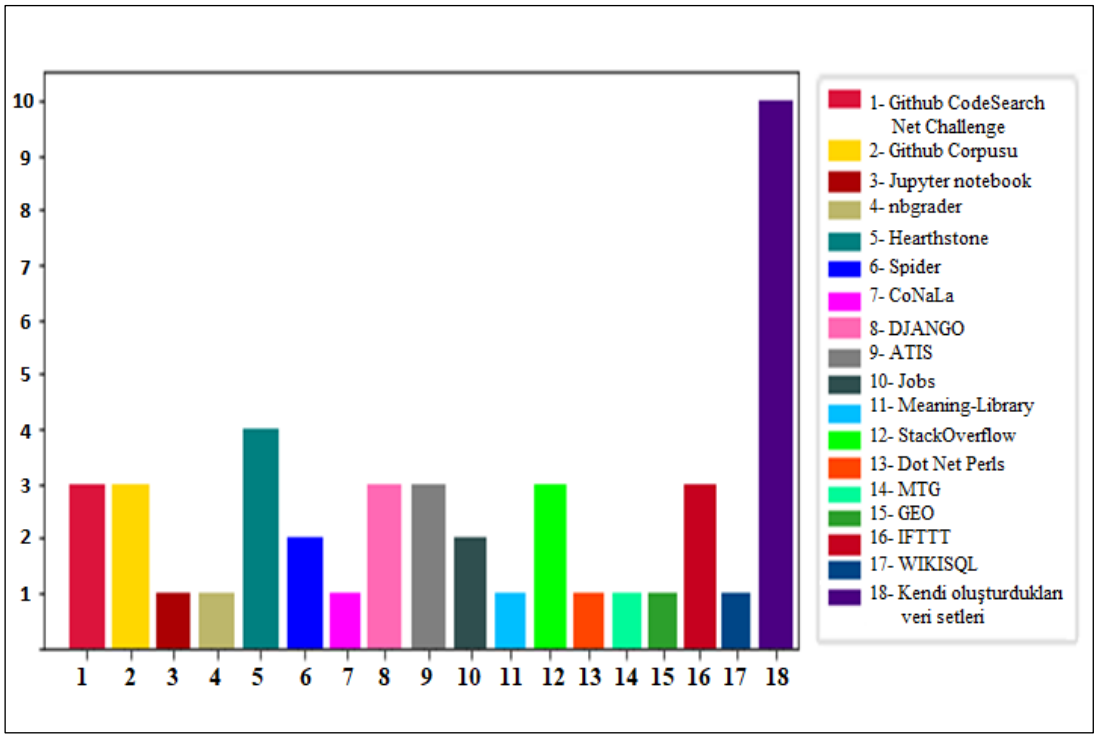
(Shin ve ark., 2019)	• Hearthstone • Spider	Python programlarından oluşan bir veri seti olan Hearthstone (Card2code, 2017) ve Çeşitli veri tabanlarından elde edilen SQL sorgularından oluşan büyük bir veri seti olan Spider (Spider, 2018) kullanılmıştır.
(Schlegel ve ark., 2019)	• Kendi oluşturdukları veri seti	Kendi geliştirdikleri küçük boyutlu bir veri setinden bahsedilmiş ancak detay verilmemiştir.
(Zhu ve ark., 2019)	• Django • ATIS • Jobs	**Django:** Her bir kod satırının manuel şekilde doğal dil açıklaması ile etiketlendiği Python web çerçevesidir. **ATIS:** 4434 eğitim veri seti ve 140 test veri setinden oluşan, λ hesabı şeklinde kodlar içeren ATIS havayolu seyahat sistemi veri kümesidir. **Jobs:** 500 eğitim veri seti ve 140 test veri setinden oluşan sorgulan doğal dil ifadelerinden ve bu ifadelere karşılık gelen prolog biçiminde kodlardan oluşan bir veri kümesidir.
(Gu ve ark., 2018)	• Github verileri • Stack Overflow verileri	Github sitesinden toplanan 18,2 milyon Java kod parçacığından (yorumlanan yöntemler biçiminde) oluşan bir külliyat üzerinde eğitim gerçekleştirilmiştir. Stack Overflow'tan elde edilen 50 sorgu kullanarak kod araması yapılmıştır.
(Zhong ve ark., 2017)	• WIKISQL	Kendi oluşturdukları bir veri tabanı olan WIKISQL Wikipedia'daki 24.241 HTML tablosundan çıkarılan doğal dil soruları, SQL sorguları ve tablolarının 80.654 adet açıklamalı örneğinden oluşan bir külliyattır. WikiSQL'de kullanılan tabloları hem ham JSON formatında hem de bir SQL veritabanı formatında yayınlamışlardır (WikiSQL, 2017).
(Yin & Neubig, 2017)	• Hearthstone • DJANGO • IFTTT	Python programlarından oluşan bir veri seti olan Hearthstone ve Her kod satırının manuel şekilde doğal dil açıklaması ile etiketlendiği Django kullanılmıştır. HS veri seti daha homojen yapıdadır bunun yanında Django veri seti daha çeşitlilik barındıran gerçek dünya verilerini içermektedir. IFTTT veri seti ise IFTTT Görev Otomasyon uygulaması tarafından kullanılan alan özgü bir dilde (DSL-Domain-Specific Language) yazılmıştır. Örnekler uygulama kullanıcıları tarafından oluşturulduğundan gürültülü veriler içermektedir.

(Ernst, 2017; Lin ve ark., 2017)	• Kendi oluşturdukları veri seti	Modeli eğitebilmek için web ortamında 8.000 doğal dil açıklaması ve bash komutu çiftinden elde edilen veriler belirli bir filtreleme işleminden geçtikten sonra 5.413 adet (NL, bash) çiftinden oluşan yeni bir külliyat oluşturulmuştur. Kümeleri rastgele %80 eğitim, %10 geliştirme ve %10 test seti olarak ayırmışlardır.
(Mandal & Naskar, 2017)	• Kendi oluşturdukları veri seti	189 kelime probleminden oluşan veri seti.
(Raghothaman ve ark., 2016)	• Bing arama motoru ve Github'dan elde edilen veriler	Bing'den alınan 15 günlük arama verileri ve Github sitesinden alınan 25.000 açık kaynak projeden oluşan çağrılar kullanılmıştır. Değerlendirme için Bing'den elde edilen 30 sorgu kullanılmıştır.
(Nguyen ve ark., 2016)	• StackOverflow Corpusu	Büyük sayıda veri içeren bir Stack Overflow corpusu kullanıldığından bahsedilmiştir.
(Ling ve ark., 2016)	• MTG (Magic the Gathering) • HS (Hearthstone)	**MTG (Magic the Gathering):** 13.297 adet Java kodundan oluşan bir veri setidir. **Hearthstone:** 665 adet Python kodu ve İngilizce açıklamasından oluşan bir veri kümesidir.
(Desai ve ark., 2016)	• 1272 kod ve açıklama çiftinden oluşan veri külliyatı toplamışlardır.	535 ATIS (Air Travel Information System) verisi, Excel kitapları ve yardım forumlarından toplanan görevlere kullanıcı açıklamaları eklenerek oluşturulan 510 farklı kod açıklama çifti ve Bağımsız bir Corpus'tan elde edilen 227 görev için 227 açıklama eklenerek oluşturulan veriler olmak üzere toplamda 1272 kod açıklama çiftinden oluşan bir veri setidir.

(Dong & Lapata, 2016)	• JOBS • GEO • ATIS • IFTTT	**JOBS:** İş listeleri veri tabanına yönelik 640 sorgudan oluşan veritabanı verilerinden 500'ü eğitim 140'ı test için kullanılmıştır. **GEO:** ABD coğrafya veri tabanına ait 880 sorgu içeren veri tabanını verilerinin 680'i eğitim, 200'ü test için kullanılmıştır. **ATIS:** 5.410 sorgudan oluşan uçuş rezervasyon sistemine ait veri tabanının standart veri bölümü 4.480 eğitim, 480 geliştirme ve 450 test örneğine ayrılmıştır. **IFTTT:** Görev Otomasyon Uygulaması tarafından alana özgü bir dilde (DSL-Domain-Specific Language) oluşturulmuştur. Çalışmada Quirk ve diğerlerinin(Quirk ve ark., 2015) 77.495 eğitim, 5.171 geliştirme ve 4.294 test örneğini içeren orijinal bölümlerini kullanmışlardır.
(Shi ve ark., 2015)	• Kendi oluşturdukları veri seti	algebra.com ve answer.yahoo.com adreslerine sahip 2 web sitesinden toplanan 1.878 matematik kelime problemlerinden oluşur. Problemlerin cevapları manuel olarak eklenmiştir.
(Allamanis ve ark., 2015)	• StackOverflow • Dot Net Perls	StackOverflow ve Dot Net Perls verilerini kullanmışlardır. Veri kümelerini bir araya getirirken toplamsal gösterimi ve eleman bazında çarpımsal gösterimi kullanmışlardır. Verilerin eleman bazında çarpımsal gösteriminin toplamsal gösterimden daha iyi çalıştığını kanıtlamışladır. Ayrıca test işlemleri için Sentetik veri üretmişlerdir.
(Gulwani & Marron, 2014)	• Kendi oluşturdukları veri setleri	40 elektronik tablo görevi için 3.570 İngilizce açıklamadan oluşan veri toplamışlardır.
(Le ve ark., 2013)	• Kendi oluşturdukları veri setleri	Forumlardan toplanan 50 görev ve bu görevler için bir kullanıcı araştırmasından elde edilen 640 doğal dil açıklaması kullanılmıştır.
(Manshadi ve ark., 2013)	• Manshadi, Keenan ve Allen'ın (2012) çalışmalarında kullandıkları veri setini kullanmışlardır (Manshadi ve ark., 2012).	Veri seti rasgele olacak şekilde 3 bölüme ayrılmıştır. Kullandıkları 338 görevin 58'ini geliştirme, 210'unu eğitim ve geri kalan 70 görevi değerlendirme için kullanmışlardır.
(Cozzie ve ark., 2011; Cozzie & King, 2012)	• Kendi oluşturdukları veri setleri	Açık Kaynaklı projelerden elde ettikleri 200.000 Java dosyasını değiştirerek bu kodların yaklaşık yarısından oluşan kendi veri tabanlarını kullanmışlardır.

(Somasundaram & Swaminathan, 2011)	• Kendi oluşturdukları veri setleri	Anahtar kelime listesi kullanmışlardır.
(Knöll & Mezini, 2006)	• Anlam-Kitaplığı (Meaning-Library)	Programlama dili oluşturabilmek için doğal dil ifadelerine karşılık gelen java komutlarının tutulduğu bir veritabanı olan anlam kitaplığını kullanmışlardır.
(Little & Miller, 2006)	• Belirtilmemiş	-
(H. Liu & Lieberman, 2005)	• Belirtilmemiş	-
(Price ve ark., 2000)	• Kendi oluşturdukları veri setleri	Bilgi Çıkarım işlemleri için manuel olarak tasarladıkları 400 vaka çerçevesini (case frame) kullanmışlardır.

Literatür analizi, kod oluşturma konusundaki çalışmaların, araştırmacılar tarafından belirli veri setlerinin kullanımının sıkça tekrarlandığını göstermektedir. Şekil 6, bu çalışmalarda kullanılan veri setlerinin kullanım sıklıklarını grafiksel olarak göstermektedir. Bazı çalışmalarda ise kullanılan veri seti bilgisi eksiktir. Veri seti bilgisi belirtilmemiş olan çalışmalar, grafik içerisine dahil edilmemiştir. Grafik incelendiğinde, araştırmacıların bir kısmının kendi özgün veri setlerini oluşturmayı tercih ettiği, diğer çalışmalarda ise önceki araştırmacılar tarafından oluşturulan veri tabanlarının kullanıldığı gözlemlenmiştir.



Şekil 6. İncelen Çalışmalarda Sık Rastlanılan Veri Setleri

4.3. Performans Analizleri

Geliştirilen sistemlerin başarısını ölçebilmek için uygun değerlendirme ölçütlerinin seçilmesi önemlidir. Modelin performansının anlaşılabilmesi, benzer diğer çalışmaların başarıları ile karşılaştırılabilmesi için anlamlı ölçütler seçmek gerekmektedir. Çalışmanın bu bölümünde araştırmacıların projelerinin performansını ölçmek için kullandıkları değerlendirme metriklerine ve performans sonuçlarına yer verilmiştir.

Tablo 3. Çalışmaların Performans Metrik Analizi

Çalışma	**Metrik Adı**	**Sonuçlar**
(Nizzad & Thelijjagoda, 2022)	• Eğitim Kaybı (Training Loss) • Doğrulama Kaybı (Validation Loss)	Çalışmada ilk 20 eğitim turu (epoch) için eğitim ve doğrulama kayıplarını gösteren bir grafik sunulmaktadır.
(Yüksel & Karabıyık, 2022)	• Başarı Oranı (Success rate)	İngilizce'den Sql diline çeviri işlemlerinde %75 başarı oranı, Sistemin genelindeki testlerde ise %69,4 başarı oranı elde edilmiştir.
(Zhao ve ark., 2022)	• BLEU skoru • CodeBLUE skoru • Önceki yöntemler ile kıyaslama	CSN veri kümeleri üzerinde diğer modeller ve diğer modeller GAP-Gen'in ince ayar sonuçları; **T5** **:** 20.71 BLEU skoru 21.67 CodeBLUE skoru **GAP-GEN T5** **:** 20.86 BLEU skoru 21.68 CodeBLUE skoru **CodeT5** **:** 21.61 BLEU skoru 23.36 CodeBLUE skoru **GAP-GEN CodeT5** **:** 21.86 BLEU skoru 23.49 CodeBLUE skoru
(Perez ve ark., 2021)	• BLEU Skoru	0.22 BLEU skoru
(Hong ve ark., 2021)	• BLEU Skoru • Önceki yöntemler ile kıyaslama	**B (Beam size)** B=1 14.0 BLEU skoru B=10 18.6 BLEU skoru B=100 21.3 BLEU skoru
(Gemmell ve ark., 2020)	• BLEU Skoru	**Django** **:** 82.3 BLEU skoru **Hearthstone :** 74.5 BLEU skoru **CoNaLa** **:** 22.3 BLEU skoru
(Phan, 2019	• Kesinlik (Precision) • Duyarlılık (Recall) • F1 skoru	Kesinlik : %70,43 Duyarlılık : %75,90 F1 skoru : %73,06

(Agashe ve ark., 2019)	• BLEU Skoru • EM (Exact Match)	**LSTM:** BLEU : 20.92 EM : 5.71 (En iyi performans gösteren model(LSTM)'in context size'ı 3 olarak belirlenmiş ve model 1,5 milyon veri üzerinde eğitilmiştir.)
(Shin ve ark., 2019)	• Exact match (EM accuracy) • BLEU Skoru	**Hearthstone :** 0.197 EM 0.780 Sentence BLUE 0.766 Corpus BLUE
(Schlegel ve ark., 2019)	• Sistem Kullanılabilirlik Ölçeği (System Usability Scale, SUS)	Genel kullanılabilirliği ölçmek için programlama geçmişi olan ve olmayan 12 gönüllü ile anket gerçekleştirilmiştir. Ortalama 85.41 SUS puanı elde edilmiştir.
(Zhu ve ark., 2019)	• Doğruluk (Accuracy) • Önceki yöntemler ile kıyaslama	**ATIS :** %81,5 **Django :** %69,7 **Jobs :** %86,43
(Gu ve ark., 2018)	Önerilen yöntem olan DeepCS'ye ait; • SuccessRate@k • Precision@k • MRR değerleri Önceki 2 yöntem (**Lucene, CodeHow**) ile karşılaştırılmıştır. (En ilgili ilk k sonuç)	**Lucene** / **CodeHow** / **DeepCS** **R@1** 0,24 / 0,38 / **0,46** **R@5** 0,48 / 0,58 / **0,76** **R@10** 0,62 / 0,66 / **0,86** **P@1** 0,24 / 0,38 / **0,46** **P@5** 0,24 / 0,29 / **0,50** **P@10** 0,26 / 0,28 / **0,49** **MRR** 0,35 / 0,45 / **0,60**
(Zhong ve ark., 2017)	• Acc_{ex} (Execution accuracy) • Acc_{lf} (Logical form accuracy) • Önceki yöntemler ile kıyaslama	Acc_{ex} : %59,4 Acc_{lf} : %48,3
(Yin & Neubig, 2017)	• BLEU skoru • Doğruluk (Accuracy) • Önceki yöntemler ile kıyaslama	**HS :** 75.8 BLEU skoru 16.2 Doğruluk **DJANGO :** 84.5 BLEU skoru 71.6 Doğruluk

(Ernst, 2017; Lin ve ark., 2017)	• Acc1 *F*(top-1 full-command accuracy) • Acc3 F (top-3 full-command accuracy) • Acc1 T (top-1 command-template Accuracy) • Acc3 T (top-3 command-template Accuracy), Geliştirme seti üzerinde performansı değerlendirme metrikleri: • Kesinlik (Precision) • Duyarlılık (Recall) • F1 skoru	Acc1 F: %30,0 Acc3 F: %36,0 Acc1 T: %69,4 Acc3 T: %80,0 **KNN** k=1 Kesinlik: 82,9 - Duyarlılık: 87,0 - F1:84,9 k=5 Kesinlik: 84,6 - Duyarlılık: 89,0 - F1:86,7 k=10 Kesinlik: 82,1 - Duyarlılık: 86,2 - F1:84,1
(Mandal & Naskar, 2017)	• Doğruluk (Accuracy)	%90,48
(Raghothaman ve ark., 2016)	• FRank (sorguyla ilgili ilk üretilen çözümün sıralamasını bildirir.) • Time (İlk 10 çözümün üretilebilmesi için ihtiyaç duyulan süre)	30 adet test sorgusu için oluşturulan çözümlerin %70'inde ilk üretilen kod parçası istenen kod ile ilişkilidir. Her bir çözüm parçacığının oluşturulması için ortalama 1,5 saniye süreye ihtiyaç duyulduğu tespit edilmiştir.
(Nguyen ve ark., 2016)	-	-
(Ling ve ark., 2016)	• BLEU skoru • Doğruluk (Accuracy)	**MTG** **:** 61,4 BLEU skoru 4,8 Doğruluk **HS** **:** 65,6 BLEU skoru 4,5 Doğruluk
(Desai ve ark., 2016)	• Kesinlik (Precision) • Duyarlılık (Recall) • Hesaplama Maliyeti (Computational Cost) • Diğer yöntemler ile kıyaslama	**Precision:** İstenen çıktı 3 veri seti içinde %80'inden fazlası içinde en üst sıradadır. **Recall:** İstenen çıktı 3 veri seti içinde %90'ından fazlası için en üst sıralarda yer alan 3 sonuçtan biridir. **Computational Cost:** Giriş verilerinden sonuç elde etmek için verilerin %85'inden fazlasının 1 saniyenin altında ve çok azının 3 saniyenin üzerinde bir sürede sonuçlandığı belirtilmiştir.
(Dong & Lapata, 2016)	• Doğruluk (Accuracy) • F1 Skoru • Önceki yöntemler ile kıyaslama	**JOBS** **:** 90,0 Doğruluk **GEO** **:** 87,1 Doğruluk **ATIS** **:** 84,6 Doğruluk **IFTTT** **:** 74,2 F1 skoru

(Shi ve ark., 2015)	• Kesinlik (Precision) • Duyarlılık (Recall) • F1 skoru	**Kesinlik** : %95,4 **Duyarlılık** : %60,2 **F1 skoru** : %73,8
(Allamanis ve ark., 2015)	• MRR (Mean Reciprocal Rank)	Çarpımsal model en yüksek MRR değerine ulaşmaktadır.
(Gulwani & Marron, 2014)	• F1 skoru	%97,6 F1 skoru
(Le ve ark., 2013)	• Doğruluk (Accuracy)	%90 Doğruluk
(Manshadi ve ark., 2013)	• PbE(Programming by Example) • PPbE(probabilistic PbE) ve NLPbE(NL-based model) modelleri kıyaslanmıştır.	PPbE modelinin, taban çizgisinden %11 daha başarılı performans gösterdiği gözlemlenmiştir. NL açıklamalarının dahil edilmesiyle başarı oranı üç kattan fazla artmıştır.
(Cozzie ve ark., 2011; Cozzie & King, 2012)	• Doğruluk	69 test programından 55'ini doğru bir şekilde oluşturulmuştur.
(Somasundaram & Swaminathan, 2011)	• TPR (True Positive Rate)	TPR değeri sistemin ilk kullanım durumlarında düşük iken kelime dağarcığı genişledikçe yükseldiği ve doyum noktasına ulaştığı gözlemlenmektedir.
(Knöll & Mezini, 2006)	-	-
(Little & Miller, 2006)	• t-test	3 kadın 6 erkekten oluşan 9 kişilik (5'i Bilgisayar bilimleri anabilim dalı öğrencisi) kullanıcı grubu ile test gerçekleştirilmiştir. Programcı olmayan grup %84, programcı olan grup ise %95 başarılı olmuştur.
(H. Liu & Lieberman, 2005)	• Likert5 ölçeği	Metaforun kullanıcılar için faydasını anlamak amacıyla 7'si programcı, 6'sı programcı olmayan 13 kişilik bir kullanıcı araştırması gerçekleştirilmiştir. Kullanıcılara kâğıt üzerinde ve metafor ile beyin fırtınası yapmaları üzerine Likert5 ölçeğinde puanlamaları sorulmuştur.
(Price ve ark., 2000)	-	-

5. SONUÇ VE ÖNERİLER

Bu çalışmada, doğal dil metinlerinden programlama dili kodu oluşturma alanında literatürde bulunan çalışmalar yarı sistematik bir şekilde incelenmiştir. İncelenen makalelerin temelinde, doğal dilin anlaşılması gibi kritik bir problem yatmaktadır. Kelimelerin morfolojik yapısı, kullanılan doğal dilin doğası, dilin gramer kuralları, doğal dil metinlerinin yan anlamları, bağlamları, aldığı ekler vb. birçok kritik etmen, dil anlama sürecini karmaşıklaştırmaktadır. Kod oluşturma süreçlerindeki bu karmaşık durumlar çalışmada detaylı şekilde gözden geçirilmiştir. Bu alanda geliştirilen yöntemlerin temel özellikleri, avantajları ve dezavantajları karşılaştırmalı olarak sunulmuştur. Ayrıca, bu alanda kullanılan veri kümeleri, değerlendirme metrikleri ve zorluklar da tartışılmıştır.

Bu çalışmadan elde edilen sonuçlar ile doğal dil metinlerinden programlama dili kodu oluşturma problemine çeşitli yaklaşımların geliştirildiği ancak geliştirilen bu yaklaşımların henüz yeterli başarıya ulaşmadığı gözlemlenmiştir. Bu alanda karşılaşılan en önemli zorluklardan biri, doğal dilin anlaşılması ve yorumlanmasıdır. Doğal dilin çok anlamlılığı, bağlama duyarlılığı, gramer kuralları, morfolojik yapısı gibi özellikleri, doğal dil metinlerinin programlama dili koduna dönüştürülmesini güçleştirmektedir. Bu nedenle, doğal dil işleme alanında geliştirilen yöntemlerin bu alana uyarlanması ve geliştirilmesi gerekmektedir. İncelenen çalışmalar kural tabanlı yöntemlerle derin öğrenme tekniklerinin beraber kullanılacağı hibrit yaklaşımların gelecek çalışmalar için umut vadettiğini göstermektedir. Ayrıca bu alanda kullanılan veri kümelerinin de yeterli büyüklükte, çeşitlilikte ve kalitede olmadığı görülmektedir. Veri kümelerinin küçük, dar kapsamlı ve gürültülü olması, yöntemlerin genelleştirilebilirliğini ve doğruluğunu etkilemektedir. Bu nedenle, bu alanda daha büyük, daha kapsamlı ve daha temiz veri kümelerinin oluşturulması ve paylaşılması önemlidir.

Bu alanda değerlendirme metrikleri olarak genellikle kodun çalışabilirliği, doğruluğu ve kalitesi gibi ölçütler kullanılmaktadır. Ancak, bu ölçütlerin yeterli olmadığı, kodun anlamsal ve işlevsel olarak doğru olmasının yanı sıra, okunabilir, anlaşılır, bakımı ve geliştirilmesi kolay olması gerektiği belirtilmektedir. Bu nedenle, bu alanda daha kapsamlı ve objektif değerlendirme metrikleri geliştirilmesi gerekmektedir.

Bu çalışma, doğal dil metinlerinden programlama dili kodu oluşturma alanında yapılan çalışmaları derlemek ve bu alandaki mevcut durumu ortaya koymak amacıyla yapılmıştır. Yapılan çalışma sonucunda bu alanda daha fazla araştırma yapılması ve geliştirilen yöntemlerin iyileştirilmesi gerektiği sonucuna varılmıştır. Bu çalışmanın giriş bölümünde, bahsedilen alana yönelik ülkeler açısından oluşturulan grafikler göz önüne alındığında özellikle Türkçe dil işleme süreçlerine daha çok odaklanılması gerektiği ve bu alanda daha fazla araştırmacıya ihtiyaç duyulduğu görülmektedir. Bu alanda yapılacak çalışmaların, doğal dil işleme, programlama dilleri ve makine öğrenmesi gibi farklı disiplinler arasında köprü kurarak hem bilimsel anlamda hem de uygulama açısından katkılar sağlayacağı düşünülmektedir.

Doğal dil işleme ve doğal dilden programlama kodu oluşturma alanında Türkçe kaynak noktasında eksiklikler bulunmaktadır. Bu çalışmanın temel motivasyonu bahsedilen alanlardaki eksikliklerin giderilmesi, zorlukların çözümüne katkı sağlanabilmesi ve bu çalışmaları daha ileri seviyelere taşıyabilme arzusudur. Bu bağlamda, geliştirilen yöntemlerin derlenmesiyle mevcut bilgi birikimi derinleştirilerek özellikle Türkçe dilinde literatürde bulunan boşlukları doldurmak ve alandaki güncel teknoloji yöntemleri sunarak katkıda bulunmak amaçlanmaktadır. Ayrıca bu çalışma ile ilgili alanda çalışacak diğer araştırmacılara rehberlik etmek ve gelecekte yapacakları çalışmalarda araştırmacılar için yol haritası oluşturabilmek hedeflenmektedir.

Yazarların Katkısı
Yazarların makaleye katkıları eşit orandadır.

Çıkar Çatışması Beyanı
Yazarlar arasında herhangi bir çıkar çatışması bulunmamaktadır.

Araştırma ve Yayın Etiği Beyanı
Yapılan çalışmada araştırma ve yayın etiğine uyulmuştur.

KAYNAKÇA

Agashe, R., Iyer, S., & Zettlemoyer, L. (2019). JuICe: A Large Scale Distantly Supervised Dataset for Open Domain Context-based Code Generation. *arXiv preprint arXiv:1910.02216.*

Allamanis, M., Tarlow, D., Gordon, A. D., & Wei, Y. (2015). Bimodal Modelling of Source Code and Natural Language. In *International conference on machine learning* (pp. 2123-2132). PMLR.

Almeida, F., & Xexéo, G. (2019). Word Embeddings: A Survey. *arXiv preprint arXiv:1901.09069*. http://arxiv.org/abs/1901.09069

Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, 1142(1). https://doi.org/10.1088/1742-6596/1142/1/012012

Barone, A. V. M., & Sennrich, R. (2017). A parallel corpus of Python functions and documentation strings for automated code documentation and code generation. *arXiv preprint arXiv:1707.02275.* http://arxiv.org/abs/1707.02275

Bhatt, S. (2018). Reinforcement Learning 101. https://towardsdatascience.com/reinforcement-learning-101- e24b50e1d292 adresinden 07 Kasım 2023 tarihinde alınmıştır.

Brown, P. E., Della Pietra, V. J., Della Pietra, S. A., & Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation.

Card2code. (2017) https://github.com/deepmind/card2code adresine 23 Temmuz 2023 tarihinde erişilmiştir.

Chowdhary, K., Chowdhary, K. R. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649

Cozzie, A., Finnicum, M., & King, S. T. (2011). Macho: Programming With Man Pages. In *13th Workshop on Hot Topics in Operating Systems (HotOS XIII).*

Cozzie, A., & King, S. T. (2012). Macho: Writing Programs with Natural Language and Examples. www.acoz.net/macho

Delua, J. (2021). Supervised vs. Unsupervised Learning: What's the Difference?. https://www.ibm.com/blog/supervised-vsunsupervised-learning/ adresinden 06 Kasım 2023 tarihinde alınmıştır.

Desai, A., Gulwani, S., Hingorani, V., Jain, N., Karkare, A., Marron, M., Sailesh, R., & Roy, S. (2016). Program synthesis using natural language. *Proceedings - International Conference on Software Engineering*, *14-22-May-2016*, 345-356. https://doi.org/10.1145/2884781.2884786

Dong, L., & Lapata, M. (2016). Language to Logical Form with Neural Attention. http://arxiv.org/abs/1601.01280

Ernst, M. D. (2017). Natural language is a programming language: Applying natural language processing to software development. *Leibniz International Proceedings in Informatics, LIPIcs*, *71*. https://doi.org/10.4230/LIPIcs.SNAPL.2017.4

Gemmell, C., Rossetto, F., & Dalton, J. (2020). Relevance Transformer: Generating Concise Code Snippets with Relevance Feedback. *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005-2008. https://doi.org/10.1145/3397271.3401215

Gu, X., Zhang, H., & Kim, S. (2018). Deep code search. *Proceedings - International Conference on Software Engineering*, 933-944. https://doi.org/10.1145/3180155.3180167

Gulwani, S., & Marron, M. (2014). NLyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 803-814. https://doi.org/10.1145/2588555.2612177

Hong, J., Dohan, D., Singh, R., Sutton, C., & Zaheer, M. (2021). *Latent Programmer: Discrete Latent Codes for Program Synthesis*.

Husain, H., Wu, H.-H., Gazit, T., Allamanis, M., & Brockschmidt, M. (2019). *CodeSearchNet Challenge: Evaluating the State of Semantic Code Search*. http://arxiv.org/abs/1909.09436

Knöll R, & Mezini M. (2006). *Pegasus – First Steps Toward a Naturalistic Programming Language*. Association for Computing Machinery.

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. In *Information (Switzerland)* (Vol. 10, Issue 4). MDPI AG. https://doi.org/10.3390/info10040150

Krogh, A. (2008). What are artificial neural networks? In *Nature Biotechnology (Vol. 26)*. http://www.r-project.org/

Le, V., Gulwani, S., & Su, Z. (2013). SmartSynth: Synthesizing Smartphone Automation Scripts from Natural Language. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services* (pp. 193-206).

Lin, X. V., Wang, C., Pang, D., Vu, K., Zeelemoyer, L., & Ernst, M. D. (2017). Program Synthesis from Natural Language Using Recurrent Neural Networks. *University of Washington Department of Computer Science and Engineering, Seattle, WA, USA, Tech. Rep. UW-CSE-17-03-01*.

Ling, W., Grefenstette, E., Hermann, K. M., Kočiský, T., Senior, A., Wang, F., & Blunsom, P. (2016). Latent Predictor Networks for Code Generation. *arXiv preprint arXiv:1603.06744*. http://arxiv.org/abs/1603.06744

Little, G., & Miller, R. C. (2006). *Translating Keyword Commands into Executable Code*. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (pp. 135-144).

Liu, H. (2004), MontyLingua v.2.1(Python and Java) A Free, Commonsense-Enriched Natural Language Understander for English. http://alumni.media.mit.edu/~hugo/montylingua/ adresine 31 Ağustos 2023 tarihinde erişilmiştir.

Liu, H., & Lieberman, H. (2005). Metafor: Visualizing Stories as Code. In *Proceedings of the 10th international conference on Intelligent user interfaces* (pp. 305-307).

Liu, X., & Wu, D. (2018). From natural language to programming language. In *Innovative Methods, User-Friendly Tools, Coding, and Design Approaches in People-Oriented Programming* (ss. 110-130). IGI Global. https://doi.org/10.4018/978-1-5225-5969-6.ch004

Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., Sundaresan, N., … Liu, S. (2021). CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation. *arXiv preprint* http://arxiv.org/abs/2102.04664

Mandal, S., & Naskar, S. K. (2017). Natural Language Programming with Automatic Code Generation towards Solving Addition-Subtraction Word Problems. Içinde *NLP Association of India*. NLPAI. http://docs.oracle.com/javase/

Manshadi, M., Gildea, D., & Allen, J. (2013). Integrating Programming by Example and Natural Language Programming. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 27, No. 1, pp. 661-667).

Manshadi, M., Keenan, C., & Allen, J. (2012, July). Using the crowd to do natural language programming. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Mansouri, A., Affendey, L. S., & Mamat, A. (2008). Named Entity Recognition Approaches. In *IJCSNS International Journal of Computer Science and Network Security (*Vol. 8, Issue 2).

Mou, L., Men, R., Li, G., Zhang, L., & Jin, Z. (2015). *On End-to-End Program Generation from User Intention by Deep Neural Networks*. http://arxiv.org/abs/1510.07211

Nguyen, A. T., & Nguyen, T. N. (2015). Graph-based statistical language model for code. *Proceedings - International Conference on Software Engineering*, *1*, 858-868. https://doi.org/10.1109/ICSE.2015.336

Nguyen, T., Rigby, P. C., Nguyen, A. T., Karanfil, M., & Nguyen, T. N. (2016). T2API: Synthesizing API code usage templates from english texts with statistical translation. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *13-18-November-2016*, 1013-1017. https://doi.org/10.1145/2950290.2983931

Nizzad, A. R. M., & Thelijjagoda, S. (2022). Designing of a Voice-Based Programming IDE for Source Code Generation: A Machine Learning Approach. *Proceedings - International Research Conference on Smart Computing and Systems Engineering, SCSE 2022*, 14-21. https://doi.org/10.1109/SCSE56529.2022.9905095

Perez, L., Ottens, L., & Viswanathan, S. (2021). Automatic Code Generation using Pre-Trained Language Models. http://arxiv.org/abs/2102.10535

Phan, H. (2019). Self Learning from Large Scale Code Corpus to Infer Structure of Method Invocations. https://www.programcreek.com/

Pise, N. N., & Kulkarni, P. (2008). A survey of semi-supervised learning methods. *Proceedings - 2008 International Conference on Computational Intelligence and Security*, CIS 2008, 2, 30–34. https://doi.org/10.1109/cis.2008.204

Price, D., Riloff, E., Zachary, J., & Harvey, B. (2000). NaturalJava: A Natural Language Interface for Programming in Java.

Quirk, C., Mooney, R., & Galley, M. (2015). Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 878-888).

Raghothaman, M., Wei, Y., & Hamadi, Y. (2016). SWIM: Synthesizing what i mean code search and idiomatic snippet synthesis. *Proceedings - International Conference on Software Engineering*, *14-22-May-2016*, 357-367. https://doi.org/10.1145/2884781.2884808

Schlegel, V., Handschuh, S., Lang, B., & Freitas, A. (2019). Vajra: Step-by-step Programming with Natural Language. *International Conference on Intelligent User Interfaces, Proceedings IUI*, *Part F147615*, 30-39. https://doi.org/10.1145/3301275.3302267

Scopus. (2023). "Generating programming language code using natural language" cümlesi kullanılarak yapılan tarama. https://www.scopus.com/ adresinden 01 Eylül 2023 tarihinde alınmıştır.

Shi, S., Wang, Y., Lin, C.-Y., Liu, X., & Rui, Y. (2015). Automatically Solving Number Word Problems by Semantic Parsing and Reasoning. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1132-1142).

Shin, R., Allamanis, M., Brockschmidt, M., & Polozov, O. (2019). Program Synthesis and Semantic Parsing with Learned Code Idioms. *Advances in Neural Information Processing Systems*, *32*.

Siddhartha, B. S., Khyani, D., Niveditha, N. M., & Divya, B. M. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. *Journal of University of Shanghai for Science and Technology*, 22(10), 350-357.

Somasundaram, K., & Swaminathan, H. (2011). Automatic Programming through Natural Language Compiler. In *Proceedings on the International Conference on Artificial Intelligence (ICA*I) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Spider. (2018) Yale Semantic Parsing and Text-to-SQL Challenge. https://yale-lily.github.io/spider adresine 23 Temmuz 2023 tarihinde erişilmiştir.

Stecanella, B. (2019) Understanding TF-ID: A Simple Introduction. https://monkeylearn.com/blog/what-is-tf-idf/ adresinden 05 Kasım 2023 tarihinde alınmıştır.

Wan, Y., Zhao, Z., Yang, M., Xu, G., Ying, H., Wu, J., & Yu, P. S. (2018). Improving automatic source code summarization via deep reinforcement learning. *ASE 2018 - Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 397-407. https://doi.org/10.1145/3238147.3238206

WikiSQL. (2017). https://github.com/salesforce/WikiSQL adresine 23 Temmuz 2023 tarihinde erişilmiştir

Yin, P., & Neubig, G. (2017). A Syntactic Neural Model for General-Purpose Code Generation. *arXiv preprint arXiv:1704.01696.*

Yse, D. L., (2021). Text Normalization for Natural Language Processing (NLP). https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa6 adresinden 03 Kasım 2023 tarihinde alınmıştır.

Yüksel, A. S., & Karabıyık, M. A.(2022). Doğal dil işleme yöntemleriyle metinden SQL sorgusu tahmini üzerine bir çalışma A study on text-to-SQL query prediction with natural language processing methods. *Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi*, 11(4), 846-855.

Zhao, J., Song, Y., Wang, J., & Harris, I. G. (2022). GAP-Gen: Guided Automatic Python Code Generation. *arXiv preprint arXiv:2201.08810.*

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. http://arxiv.org/abs/1709.00103

Zhu, Y., Zhang, Y., Yang, H., & Wang, F. (2019). GANCoder: An Automatic Natural Language-to-Programming Language Translation Approach based on GAN. In *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8* (pp. 529-539). Springer International Publishing. http://arxiv.org/abs/1912.00609