T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOĞAL DİL İŞLEME TABANLI BİLGİSAYAR AĞ TERİMLERİNİN WORDNET ONTOLOJİSİ KULLANILARAK OLUŞTURULMASI

Yeşim AKTAŞ

Danışman
Doç. Dr. Abdülkadir ÇAKIR

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
ISPARTA - 2017

TEZ ONAYI

Yeşim AKTAŞ tarafından hazırlanan **"Doğal Dil İşleme Tabanlı Bilgisayar Ağ Terimlerinin Wordnet Ontolojisi Kullanılarak Oluşturulması"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak başarı ile savunulmuştur.

Danışman	**Doç. Dr. Abdülkadir ÇAKIR** Süleyman Demirel Üniversitesi	.....
Jüri Üyesi	**Doç. Dr. Ecir Uğur KÜÇÜKSİLLE** Süleyman Demirel Üniversitesi	.....
Jüri Üyesi	**Yrd. Doç. Dr. Fırat YÜCEL** Akdeniz Üniversitesi	.....

Enstitü Müdürü **Prof. Dr. Yasin TUNCER** .....

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Yeşim AKTAŞ

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

DOĞAL DİL İŞLEME TABANLI BİLGİSAYAR AĞ TERİMLERİNİN WORDNET ONTOLOJİSİ KULLANILARAK OLUŞTURULMASI

Yeşim AKTAŞ

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Abdülkadir ÇAKIR

Bilgisayar ile doğal dilde iletişimin sağlanması için bilgisayarın doğal dil kurallarını öğrenmesi gerekmektedir. Türkçe bir kelime, ekleriyle birlikte yeni bir kelime oluşturma özelliğine sahiptir. Doğal dil işleme; yaygın olarak bilinen yapay zeka ve dilbilimin alt kategorisi olup doğal dillerin kurallı yapısını çözümleyerek anlaşılması veya yeniden anlam katması amacını taşır.

Türkçe için yapılan doğal dil işleme çalışmaları, bilgisayar terimleri için otomatikleştirme adına yetersizdir. Bu tez çalışması bilgisayar terimlerinin ontoloji üzerine doğru yerleştirilmesi amacıyla yapılmıştır.

Yapılan bu çalışmada, eklerinden ayrılmış ağ terimlerini içeren bir Wordnet ontolojisi oluşturularak, ontolojide bulunan iki terim arasındaki bağlantı hesaplanmıştır. Terimlerin adlandırılması için en etkili yol ise doğal dil işlemedir. Verilen bir yazı paragrafında bilgisayar ağ terimlerinin ontolojik bir sözlükte aranması ve sözlükte bulunmayan kelimelerin otomatik olarak ontolojiye eklenmesi gerçekleştirilmiştir. Yakın anlamlı ve alt-üst terim ilişkisi içerisinde bulunan terimler graf yapısı şeklinde birbirleri ile bağlanmıştır. Bütün terimlerin birbirleri ile bağlanması sonucu sözlük ontolojisi oluşturulmuştur. Bağıntı kurmada öncelikle kavram haritasından yararlanılmış, sonrasında doğal dil işleme algoritması ile otomatik hale getirilmiştir. Algoritma paragrafta bulunmayan kelimenin sağında ve solunda bulunan ve sözlükte bulunan 10 kelimenin ontolojideki düğümlerine bakılarak hazırlanmış ve en üst düğüme yakın anlam ilişkisi ile ilişkilendirilmiştir. Geliştirilen algoritma C# form application uygulaması şeklinde hazırlanmıştır.

Anahtar Kelimeler: Wordnet, Ağ, Ontoloji, Doğal Dil İşleme.

2017, 62 sayfa

ABSTRACT

M.Sc. Thesis

NATURAL LANGUAGE PROCESSİNG BASED COMPUTER NETWORK TERMS USING WORDNET ONTOLOGY CREATION

Yeşim AKTAŞ

Süleyman Demirel University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Abdülkadir ÇAKIR

To learn natural language rules of the computer to communicate with the computer through natural language. Create a Turkish word, create a new word with its attachments. Natural language processing; Is a subcategory of artificial intelligence and linguistics which is widely known and aims to understand or reinforce meaning by analyzing the canonical structure of natural tongues.

Natural language processing exercises for Turkish are insufficient for the automation of computer terms. This thesis study was carried out in order to correctly place computer terms on ontology.

In this thesis study, a Wordnet ontology containing network terms separated from its suffix is created and the connection between the two terms ontology is calculated. The most effective way of naming terms is natural language processing. In a given article paragraph, computer network terms were searched in an ontological dictionary and words not in the dictionary were automatically added to the ontology. Terminology that is in the relation of close meaning and lower-upper term is connected with each other in the form of graphical structure. The terminology dictionary ontology has been established to connect all terms with each other. In order to construct the correlation, firstly the concept map is utilized and then it is automated by natural language processing algorithm. The algorithm was prepared by looking at the ontological nodes of the 10 baldens found in the dictionary located on the left and right of the missing vein in the paragraph and was associated with the closest meaning relationship. The algorithm developed is C # form application.

Keywords: Wordnet, Network, Ontology, Natural Language Processing.

2017, 62 pages

TEŞEKKÜR

Bu çalışma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli Danışman Hocam Doç. Dr. Abdülkadir ÇAKIR'a ve çalışmanın her aşamasında yardımlarını esirgemeyen değerli hocam Öğr. Gör. Dr. Ebru YILMAZ İNCE'ye teşekkür ederim.

4534-YL1-15 No`lu Proje ile tezimi maddi olarak destekleyen Süleyman Demirel Üniversitesi Bilimsel Araştırma Projeleri Yönetim Birimi Başkanlığı'na teşekkür ederim.

Bursiyer olarak çalıştığım 114E952 numaralı proje ile maddi destek sağlayan TÜBİTAK'a teşekkür ederim.

Tez yazdığım süre boyunca manevi desteklerinden dolayı Süleyman Demirel Üniversitesi Yalvaç Teknik Bilimler Meslek Yüksekokulu'nda eğitim görmüş ve görmekte olan sevgili öğrencilerime, akademik ve idari personele teşekkür ederim.

Bugünlere gelmemi ve iyi bir eğitim almamı sağlayan, tezimin her aşamasında beni yalnız bırakmayan sevgili annem Bahar AKTAŞ'a ve babam Sadık AKTAŞ'a, biricik kardeşim Yasemin AKTAŞ' a teşekkür eder, sonsuz sevgi ve saygılarımı sunarım.

Yeşim AKTAŞ
ISPARTA, 2017

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

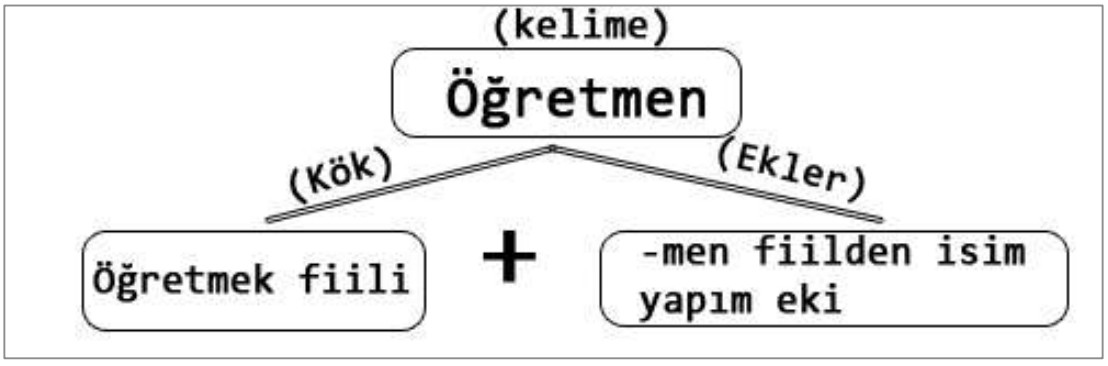
SİMGELER VE KISALTMALAR DİZİNİ

.Net	ASP.Net
C#	C Sharp
C++	C Plus Plus
DDİ	Doğal Dil İşleme
ECMA	European Computer Manufacturers Association
GAD	Gizli Anlambilimsel Dizinleme Metodu
ISO	International Organization for Standardization
SOM	Self Organizing Map
SVM	Support Vector Machines

1. GİRİŞ

Doğal dil işleme (DDİ), yapay zeka alanının alt dallarından bir olup, yapay zeka algoritmalanın bir çoğu üzerinden işlem yapılabilmektedir. Ticari, kurumsal bir çok alanda kullanılan DDİ, algoritma oluşturmanın önemli bir kısmını oluşturmaktadır. Bu algoritmalar, dilin biçimsel analizi yapılarak kodlanmaktadır. Çevrimiçi ortamlarda Microsoft, Google gibi firmalar, biçimsel analizler kullanarak çeviriler sunmaktadır. Bu çevirilerin en kapsamlısı Google tarafından yapılmıştır. Google diğer çevirilerin aksine, metinden biçime analiz yöntemini değil, direk olarak diller arası oluşturulan cümleleri kullanmaktadır. Girilen kelimenin en çok kullanılan anlamı ve yakın anlamı, ilk anlamları kabul edilmektedir (Takçı, 2004).

Ticaret, endüstri gibi birçok alanda kullanılmak üzere oluşturulan bu yapılar yapay zekanın önemli bir bölümü oluşturmaktadır. DDİ, temel görevi bir konuşma dilini çözme, eleştiri yapma ve ürün ortaya koyma olan makinelerin tasarımını ve gerçekleştirilmesini konu alan bir mühendislik dalıdır. Belli bir algoritma içermediğinden ve belirsizliklere sahip olduğundan bir bulanık mantık problemidir. Yapay zeka, biçimsel diller soyut bilgisi, tahminsel dilbilim, bilgisayar destekli dilbilim ve bilişsel pisikoloji gibi değişik alanlarda geliştirilmiş kuram, yöntem ve teknolojiler bütünüdür (Diri, 2017). Şekil 1.1'de DDİ örnek temel yapısı verilmiştir.



Şekil 1.1. DDİ Örnek Temel Yapısı

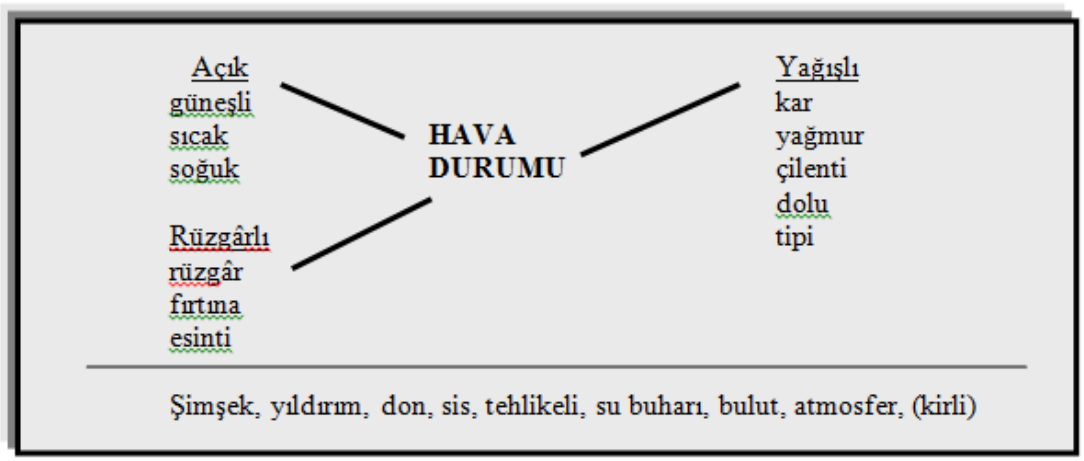
Türkçe'nin kurallı bir dil olması nedeniyle, doğal dil işlemede özel bir durumu vardır. Türkçe, kökü değişmeyen bir dildir, ek ve köklerden oluşmuş, sağlam ve

bozulmamış bir düzeni vardır. Türkçe'de kelime, kök ve kökün sonuna eklenmiş eklerden oluşmaktadır. Köklere eklenen ek sayısına bakıldığında, ek sayısı 3-5 ortalamasındadır. Kelime kökünün bilinmesi, kelimenin temel anlamını vermektedir. Her ek, kelimeye neredeyse birbiriyle bağlantılı yeni anlam kazandırır. Bir kelimenin köküne ulaşmak için veri listeleri olarak hazırlanmış sözlükler kullanılabilir. Çizelge 1.1'de bu duruma örnek verilmiştir. Aynı zamanda Türkçe ek-kök yapılı bir dil olduğu için sözlüklere gerek duymadani kurallarla köküne ulaşılabilir. (Cebiroğlu, 2002).

Çizelge 1.1. Türkçe Dilinde Kelime Türemesine Örnek (Türk Dil Bilgisi, 2016)

Göz	Göz-lük	Göz-cü	Göz-de	Göz-lem	Göz-et(mek)
Yaş	Yaş-lı	Yaş-lan	Yaş-ıt	Yaş-a	Yaş-lı-lık
Sus-(mak)	Sus-kun	Sus-tur(mak)	Sus-tur-ucu	-	-
(kök)	(gövde)	(gövde)	(gövde)	(gövde)	(gövde)

Türkçede aynı sözcük bulunduğu sıraya bakıldığında, değişik anlamlar içerebilmektedir. Alt-üst sınıf düzeni, özel ve genel anlam ilişkilerine bakılarak oluşturulmuştur. Kavramsal sınıflar bir ontoloji haritasında gösterilir ve özel anlamlar genel anlam sözcüklerine dahildir. Sözcükler, sahip oldukları anlamlardan ibaret değildir. Benzer özelliklere sahip birçok sözcük birlikte kavramsal sınıfları oluşturur. Örneğin; Wordnet dünyanın en büyük kavram ilişkileri barındıran sözlüğüdür, alt-üst sınıflar ve eş-zıt anlamlı sınıflar gibi pek çok kavramsal sınıf içerir. Bir kelime hangi kavramsal sınıfa aitse, buradan yola çıkılarak kelime hakkında bilgi sahibi olunabilir. Örneğin: "SRAM" sözcüğü sırasıyla "bilgisayar kasası > bellekler > ram > sram >" düğümlerinden geçmektedir. Bu durumda sram hem bir ram hem de bellektir. Bununla birlekte bilgisayar kasasında bulunur. (Mahcup, 2017). Şekil 1.2'de örnek alt kavram-üst kavram ilişkisi verilmiştir.



Şekil 1.2. Alt Kavram- Üst Kavram İlişkisi Örneği

Şekil 1.2'de görüldüğü üzere, hava durumu örneğine bakıldığında, havanın açık olma durumunda; güneşli, sıcak ve soğuk olduğu anlaşılabilir. Başka bir örnek verilecek olursa havanın karlı olması durumu aynı zamanda yağışlı olması durumudur.

2. KAYNAK ÖZETLERİ

Güngör (2007), çalışmasında eş anlamlılık ilişkilerini çıkarmıştır. Doğal dil işleme, sistem olarak kullanılan terimleri biçim bilimsel ve sözdizimsel olarak incelemekte, anlambilimsel özelliklerini dikkate almamaktadır.

Yaşar (2007) yaptığı çalışmada, Wordnet üzerine bilişim ontolojisi oluşturmuştur. Doğal dil işleme için kelimelerin anlam temelini esas almıştır. Wordnet ontolojisinin sınıf yapısı, dil fark etmeksizin özelliğini korumaktadır. Wordnet'te yapılar bir tablo halinde tutulur ve istenilen değerler çekilerek kullanılabilir. Aradaki kelime değerlerinin tutulduğu birden fazla tablo mevcuttur. Bu tablolara yönetici işlemleri (ekleme-silme) yapılabilmektedir.

Delibaş (2008), DDİ ile Türkçe yazım hatalarının denetlenmesini yaptığı çalışmasında, verilen bir metindeki kelimlerin köklerinin bulunması dolayısı ile eklerine ayrılması, kelimenin doğru olup olmadığını bulunması, tahmini doğru sözcük önerilmesi ve Yabancı kelimelerin sözlüğe eklenmesi gerçekteştirilmiştir.

Karadeniz (2007), araştırmasında biçimbilimsel çözümlemenin ürettiği sonuçların doğru olanının tespit edilmesini amaçlamıştır. İlk olarak Türkçe kelimelerin belirsizlik dağılımlarını çıkarmış ve daha sonra kelimeleri belirsizlik niteliklerine göre kümelemiştir. Her belirsizlik için ayrı kurallar yazmıştır. Geliştirilen çalışmanın başarımı % 82,6 olduğu verilmiştir.

Çelik (2012) yaptığı çalışmada, Wordnet'e geniş yer vermiştir. Wordnet, İngilizce sözcüklerden oluşmuş bir veritabanıdır. Her sözcük türünden kümeler halinde gruplandırılmış kelimelerdir. Synsets-kavramsal semantik ve sözcüksel ilişkiler yoluyla birbirleri ile ilişkilendirilmiştir. Wordnet'in bu yapısı gereği dilbilim ve doğal dil işleme için kullanılması kaçınılmaz bir hal almıştır.

Sosyal (2010), DDİ ve alan ontolojisi yöntemlerini kullanarak, Türkçe radyolaji raporlarından çıkarılmış bilgi elde edilmiştir. Çalışmada sistem bir morfolojik analizör olarak kullanılışmıştır.

Fırat (2017), çalışmasında biçimsel anlam analizinin uygulandığı üç boyutlu kafeslerde kullanılmasını, FrameNet etkileşimi ve FrameNet sözlük kaynağını biçimlendirmiştir. Tematik rol kafeslerinin matematik yapısını ortaya çıkarmıştır. Böylece insana insana ait zihin yapısını ifade eden ontolojiler ile dil arasındaki ilişkini de gerçekleştirmiştir.

Düzağaç (2014) çalışmasında, eş anlamlar üzerinde durmuştur. Buna göre Wordnet'te birbiri ile en yakın ilişki "eş anlamlı" ilişkisidir. Eş anlamlılar yakın anlamlara sahip kelimelerdir. Ayrıca, eş anlamlı kümesi Wordnet terimi "synset" ile ifade edilir. Synset kelimesi kullanılan kelimeye ait bütün ilişkileri gösteren soyut bir ifadedir. Wordnet; isimler, fiiller, sıfatlar, zarflar ve fonksiyon kelime olarak beş kategoride kelimeleri parçalar. Wordnet'te bulunan 27 ilişki çeşidini çalışmasında açıklamıştır. Bunlar; "alsosee", "antonym", "attribute", "cause", "derived", "derivedfromadjective", "entailment", "holonymmember", "holonympart", "holonymsubstance", "hypernym", "hypernyminstance", "hyponym", "hyponyminstance", "meronymmember", "meronympart", "meronymsubstance", "participle", "pertainym", "region", "regionmember", "similarto", "topic", "topicmember", "usage", "usagemember", "verbgroup"dur.

Per (2011), çalışmasında Türkçe için kavram çıkarımı oluşturmuştur. Türkçe karakterlerin bilgisayar dilinde bulunma sıkıntısı ve Türkçe'nin sondan eklemeli olması nedeniyle ön işleme sokulmuştur. Ön işlem sonucu eklerinden ayrılan kelimelerin isim türünde olanları kullanılmıştır. Sistem %51 başarı elde etmiştir.

Güngör (2007), çalışmasında Türkçe biçim özellikleri analizi yapmıştır. Ek yapısı ve sıralamasını incelemiş, Türkçe'nin genişletilmiş geçiş ağı formunda biçimbilimsel yapısını modellemiştir. Buna ek olarak yazılımda yazım kontrolü ve düzeltmesi yapılmaktadır.

Yavanoğlu ve Sağıroğlu (2010), çalışmalarında internet ortamında bulunan MS Word ve HTML sayfalarının dilini bilmek ve içerdiği verileri değişik dillere çevirmek için bir mekanizma oluşturmuştur. Geliştirilen sistem birden fazla işlemin işlenmesini gerçekleştirmek ve kullanıcıların bilmediği bir dili yaay sinir

ağları tabanlı akıllı bir yöntem ile doğrudan tespiti ve 40 dile çevrilebilecek mekanizmadan oluşmaktadır. 15 dil için testler uygulamışlar ve bu testlerde sistemin tam zamanında çalışma başarısının yüksek olduğunu gözlemlemişlerdir.

Nabdel (2011) çalışmasında, elemanlar hakkında ek bilgi sağlamak amacıyla öznitelikleri belirtmiştir. Öznitelikler, çalışmasında her zaman tırnak içerisinde ifade edilmiş, bu tırnak tek ve çift tırnak biçiminde belirlenmiştir.

Gruber (1993) yaptığı çalışmada, bilginin anlamsal temsili için ontolojiler kullanılmakta olduğunu savunur. Ontoloji kavramının en çok kullanılan açıklaması bilinen bir kelimeleştirmenin biçim olarak ve açıkça bir gösterimidir.

Şahin (2001) araştırmasında, anlamlı öğrenme üzerinde çalışmıştır. Dünyadaki veri artması, teknolojideki ani gelişmeler, güncel eğitim sisteminde gelişmeleri gerektirmiştir. Bilgi dünyasının bireyleri, bilgiyi tüketici değil üretici ve probleme yanıt bulucu olmalıdır. Günümüzde ezbere dayalı öğrenimin yerini anlamaya dayalı öğrenim almaktadır. Anlamlı öğrenmenin bize kazandırdığı, gelen bilginin kalıcı süre bellekte tutulması, istenildiğinde bu yeni bilgiye ulaşılabilmesi, daha sonraki bir veriyi öğrenmeyi basitleştirmesi ve daha önce karşılaşılmayan bir sorunla baş ederken bu bilginin mantıksal olarak eleştirme ve karar verme yeteneğidir.

Yılmaz (2009) çalışmasında, anlamlı öğrenmeyi gerçekleştirmek ve bunun sonucu olarak bilginin daha uızun zaman hafızada tutulması ve istenildiğinde hatırlanmasını sağlamak için kullanılan öğretimsel kavramlardan birininde "kavram haritaları" olduğunu belirtmiştir.

Jonassen (1993) ve Açar (2007)'ın yaptıkları çalışmalarda kavram haritasını önemli yapan bireysel olmasıdır tezini ortaya koymuşlardır. Bireyler uğraşlarından onlara göre ne anladıklarını ifade etmek için, harita çizer. Kişiler veriyi anlamak için kullandıkları beceri, yetenek ve bilişsel tercih farklılığından, farklı kişiler değişik kelimeler kullanarak kendilerine has farklı kavram haritaları oluştururlar.

Kabaca (2007)'ya göre kavram haritalarında bir kelime yalnızca bir defa kullanılmalıdır. Kavramlar çizgisel şekillerle içinde haritayı oluşturanın tercihine göre unutmayı engellemek amacıyla istenilen bir biçimde oluşturulabilir. Haritalarda en bilinen kelimelerle haritanın en üstünde ya da ortasında karşılaşılır. Neredeyse benzer öneme sahip kelimeler ontolojik olarak aynı düzen sırasında bulunurlar. Kelimeler özelleştikçe daha genel olan kelimein alt düğümünde oluşturulur. Birden fazla kavramın birbiri ile ilişkilendirilmesi düğümler ile gerçekleştirilir. Kavram haritalarında ne kadar istenirse o kadar düğüm bağlantısı oluşturulabilir.

McGowen (1999) çalışmasında, bilgisayar teknolojisinin, kişilere bilgilerin, grafikse ve sembolik gruplamalarla verilebileceğinin münkün olduğundan söz eder.

Chang (2001)'a göre, öğrenmeyi daha anlamlı yapmak bilginin depolanmasını beklenenden daha uzun süreli yapabilir. Kağıt ve kalemle oluşturulan kavram haritalarının içerdiği zorluklar, öğrencilerin kavram haritalarını daha kolay yapılandırabilmeleri için araştırmacıların bilgisayar destekli kavram haritalarının ortaya çıkmasını sağlamıştır. Bilgisayar destekli kavram haritalarının geçerliliği birçok deneysel çalışma ile kanıtlanmıştır.

Yılmaz (2009), çalışmasında kavram haritalarının kağıt üstünde oluşturulmasının çok sık karşılaşılan metotlardan olduğunu savunur. Ancak günümüzde bilgisayarların kullanılabilirlik bakımından objeleri daha estetiğe uygun, çok yönlü ve basit şekilde kullanılmaya hazır olduğu zaman klasik kavram haritası oluşturmak yerine dijital ortamda oluşturmanın daha uygun olduğu söylenebilir. Kavram haritası ile ilgili "işlemciler modülünün web tabanlı uzaktan eğitim ile kavram haritası tekniği desteği ile öğretilmesine örnek bir uygulama" çalışmasında, meslek liselerinin bilgisayar konusunda okutulan işlemciler modülü için; içinde değişik animasyonlar, konu sunumları ve kavram haritaları bulunan bir öğretim metodu oluşturup bu metodu işlemciler modülünü açıklayıp uygulamıştır. Kavram haritaları ile ilgili kısımda dersler sonunda öğrencilere

uyguladığı anket sonucunda, bilgisayar ortamında verilen kavram haritalarının işlemciler modülünün dersin sunumunu eğlenceli hale getirdiği ve yapılan yazılı sınav uygulamarında kavram haritası boşluklarını doldurma gibi sorulara daha olumlu yaklaşıldığı sonucuna varılmıştır.

Kapucu (2008)'nun yaptığı "bilgisayar destekli kavram haritası kullanımının, öğrencilerin bilişsel senaryo oluşturma becerileri, erişi, öğrenmelerinin kalıcılığı ve derse yönelik tutumları üzerindeki etkileri" isimli çalışmada, sosyal bilgiler dersi için bilgisayar destekli kavram haritalarının kullanılmasının, ilköğretim 6. sınıf öğrencileri üzerinde öğrenmenin kalıcılığı sonucuna uluştığını görmüştür.

Aykanat (2005) çalışmasında, ilköğretim 6. sınıf öğrencilerini kontrol ve deney olmak üzere iki gruba ayırıp öğrencilerinin ön sınav, kalıcılık sınavı ve son sınav puanları arasında ayırt edilebilir bir farklılık olmadığını fakat deney grubundaki öğrencilerin sayısının artması koşulu ile bilgisayar destekli kavram haritası kullanılarak hata oranının düzgün bir biçimde azaldığını belirlemiştir.

Anderson-Inman ve Ditson (1998)'ın araştırmalarında; bilgisayarda oluşturulan kavram haritalarının, anlamlı bir öğretim metodu olarak kabul edilmiştir. Çalışma, bilgisayarda oluşturulan kavram haritalarının, öğrencilerde yazma aynı zamanda okumalarında görselleştirmenin, eğitimi olumlu yönde etkilediği görülmüştür.

Bloehdorn ve Hotho (2006), 2004 yılındaki çalışmalarında AdaBoost adında yeni bir sistem tanıtmaktadırlar. AdaBoost, terimleri vektörler halinde getirerek kavramsal özelliklere dayalı son sınıflandırmayı yapmak için önerilmektedir.

Çelik (2012) çalışmasında yapay öğrenme tekniklerinin gelişmesi ile terimlerin dijital ortama taşınması, terimler arasındaki ilişkileri bularak terimlerin sınıfının otomatik halde bulunmasına neden olmakta görüşünü savunur. Terimleri sınıflandırma, öğrenme modellerini kullanarak hakkında ilk defa bilgi edinilen terimleri hangi sınıfa ait olduğuna karar verecek daha önceden belirlenmiş olan sınıfa atamaktadır. Çalışmasında, terimlerin sınıflandırma işleminde, terimleri

istatistiksel verilerden yararlanarak bir dizi halinde tutmayı hedefleyip, sınıflandırma işlemi için yapay öğrenme tekniklerini kullanmıştır.

Savaşçı (2010), çalışmasında Türkçe ve İngilizcenin bitişkenlik derecelerini incelemiştir. İstatiksel verilere göre bitişkenlik derecesi Türkçe için 0.56 ve İngilizce için 0.12 bulmuştur. Buna göre Türkçe'de yapılacak olan DDİ çalışmaları İngilizce'ye göre daha zordur.

Sebastiani (2002)'ye göre yapay öğrenmenin terim sınıflandırmada kullanılması ile terim sınıflandırma işleminin hızlandırılmış olması bu alanda en önemli gelişmelerden biri olmaktadır. Yapay öğrenme tekniklerinin kullanımının giderek artması bu tür olasılık hesabı ile sınıflandırma yapılacak işlemlerde, karar ağaçları, en yakın komşu sınıflandırması ve yapay sinir ağları terim sınıflandırmada kullanılmaktadır.

Cortes ve Vapnik (1995), yapay bir öğrenim yöntemi olarak Support Vector Machines (SVM) yöntemini ortaya çıkararak terim sınıflandırma işleminde de kullanılabilecek yöntemler arasında saymaktadır. SVM yöntemi kullanılarak ilk terim sınıflandırma işlemi 1998 yılında Joachims tarafından başlatılmıştır. SVM yöntemi doğrusal olmayan bir model olup diğer dört sınıflandırıcı ise:

- NaiveBayes(NB)
- Rocchio metodu
- K-en yakın komşusu sınıflandırıcı(k-NN)
- C4.5 decisiontree algoritması

Çelik (2012) çalışmasında sınıflandırıcılarla karşılaştırma yapıp en iyi sonucu veren sınıflandırıcının SVM yöntemi olduğu kanısına varmıştır.

Forman (2003) çalışmasında, sınıflandırma işlemi yapılmadan önce verinin boyutunda azaltmaya giderek gereksiz verileri hesaba katmayıp önem katsayısı yüksek verilerin işleme katılmasını savunmuştur. Bu işleme öznitelik seçimi adı

verilmektedir. Terim sınıflandırma işleminde de öznitelik çıkarımı önemli bir adım olmaktadır. Öznitelik seçimi için literatürde birden fazla yol bulunmaktadır.

Chua ve Kulathuramaiyer (2004)'in çalışmalarında terim sınıflandırma işleminin öznitelik çıkarımı Wordnet ile sağlanmıştır. Öznitelik seçimi olarak Wordnet'i kullanmış ve sonucun başarılı olduğu görülmüştür.

Zhang (2005)'a göre, terimleri sınıflandırmada öznitelikleri seçim aşamasında semantik özelliklerini kullanmak ve bu özellikleri Wordnet ile seçmenin iyi bir kaynak olduğunu iddia edilmektedir. Çalışmasına semantik özellikleri ayırt etmeye çalışarak başlamış ve kullandığı yöntemin semantik özellikleri farklı sınıflar üzerinde sınıflandırma özelliğinin artacağı kanısına varmıştır.

Adalı (2009), çalışmasında Türkçe metinlerin otomatik olarak işlenmesi için, belgeler üzerinde bilgi çıkarımı yapmıştır. Bu çıkarımı yaparken belge içindeki ipuçlarından yararlanılmıştır. Belge tutarlılığı için, belgede bulunan varlıklar arası ilişkiler incelenmiştir.

Li (2009), çalışmasında anlam özelliklerinden yola çıkmıştır. Anlam özelliklerini bulma konusunda da Wordnet'ten yararlanmıştır.

3. MATERYAL VE METOT

3.1. Ontolojik Sözlük

Ontolojiler varlıklar için ortak tanımlamalardır. Farklı terimlerin açıklanması için ontolojilere ihtiyaç duyulmaktadır (Can, 2010). Birbirili ilişkili yani ontolojik sözlük, birden çok anlam dikkate alınarak oluşruşulmuştur. Eş anlam ve yakın anlam ilişkilerinin tutulduğu geliştirilen yeni sözlük için 4 farklı sözlük ve "Bilgisayar Haberleşmasi ve Ağ Teknolojileri" kitabı (Çölkesen, 2003) kelimeleri kullanılmıştır.

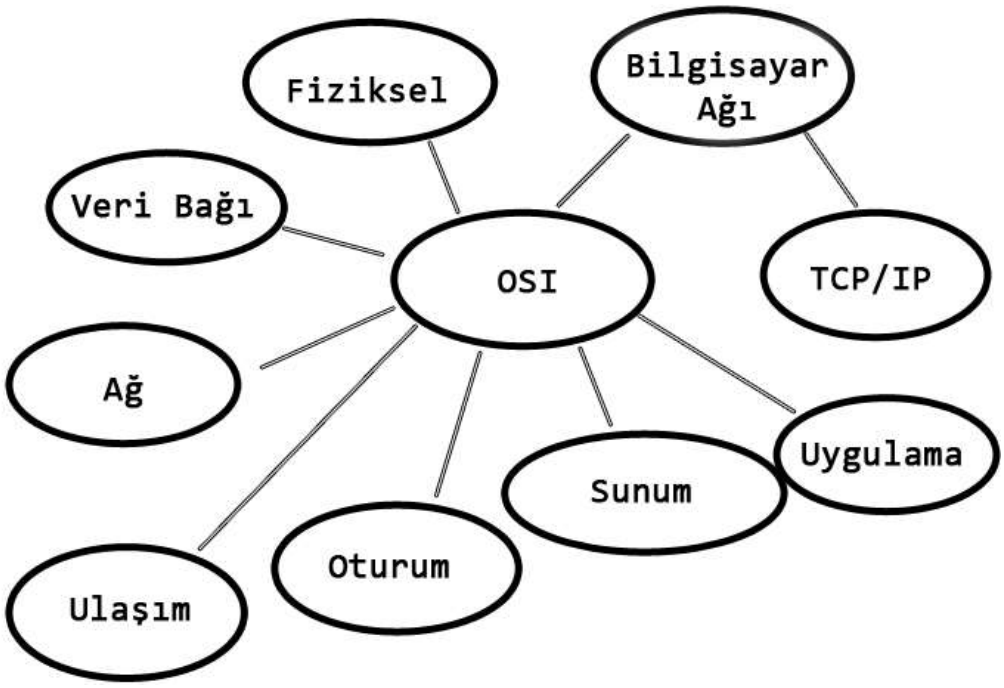
3.1.1. Wordnet

Wordnet projesi, 1985 yılında Princeton Üniversitesi Bilişsel Bilimler Laboratuarı'nda Miller tarafından başlatılılmış bir ontolojik sözlük projesidir. İngilizce sözcükleri eş anlam sınıfında gruplandırır ve sözcüklerin kısa, genel tanımlarından yararlanarak eş anlam ilişkisini ortaya çıkarır (Fellbaum, 1998). Bu açıklamadan yola çıkılaran iki temel amaç vardır. Bunlardan birincisi sözcük tanımı verilen sözlük listesini, kelimelerden kavramları, kavramların içerdiği özellikleri ve kavramlar arasındaki ilişkileri ortaya çıkarmaktır. İkincisi yapay zeka yöntemlerini üzerinde incelemek ve otomatik paragraf çözümlemesini desteklemektir. WordNet yayınlandığı günden beri hiçbir ücret talep etmeden sunulmaktadır. Tüm kullanıcılar bu sözlükten yararlanabilmektedir. Son sürümlerinde Wordnet; 155327 farklı kelimenin alt-üst, eş anlam, zıt anlam gibi çeşitli anlamlı ilişkilendirme sonuçlarını üretmektedir.

Wordnet bir kaç kaynakta ontoloji olarak anılmaktadır ancak oluşturulma amacı dikkate alındığında böyle bir amacı bulunmamaktadır. Asıl amaç, İngilizce sözcüklerin bir takım ilişkilerle bir yapı haline getirilmesidir (Güner, 2005). Böylece alt kavram-üst kavramlar oluşturulmuştur. Bu sebeple Wordnet kavramsal bir ontoloji olarak görülebilmektedir. Wordnet kavram örneği Şekil 3.1'de görülmektedir.

İngilizce Wordnet'in, DDİ araştırmalarında kullanılabilirliği gözlendikçe, diğer diller için de Wordnet'ler geliştirilmeye başlanmıştır. Hâlihazırda kırkın üzerinde farklı dil için oluşturulmak üzere Wordnet çalışması yapılmıştır. Ayrıca bunun üzerine EuroWordnet ortaya çıkmıştır. EuroWordnet her dil için ayrı birer wordnetten oluşan 8 farklı Avrupa dili için geliştirilmiştir (Vossen, 1998).

EuroWordnet gibi Balkan dilleri için de Balkanet oluşturulmuştur (Tufiş, 2004). Aynı şekilde Balkanet projesi altında Türkçe Wordnet hazırlanmıştır (Bilgin, 2004).



Şekil 3.1. Wordnet kavram örneği (Yılmaz İnce, 2016)

3.1.2. Kullanılan Türkçe sözlükler

Bilişim sözcüklerinin birleştirilmesi için hazırlanan TXT dosyasında; Türkçe Wordnet, Türk Dil Kurumu TDK Eş Anlamlar Sözlüğü (Türk Dil Kurumu-TDK), Türkiye Bilişim Derneği Bilişim Sözlüğü (Türkiye Bilişim Derneği - TBD), Eş Anlam- Yakın Anlam sözlüğü (Türkçe Eş Anlamlar Sözlüğü) ve "Bilgisayar Haberleşmasi ve Ağ Teknolojileri" kitabı üzerinde bulunan kelimelerin kavramsallaştırılması sonucu oluşturulmuştur. Kullanılan sözlükler Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Bilgisayar ağ terimleri sözlüğünde kullanılan sözlükler

Sözlük	**Kelime Sayısı**
Wordnet	14.796
TDK Eş Anlamlar Sözlüğü	125.009
Eş Anlam-Yakın Anlam Sözlüğü	82.938
TBD Bilişim Sözlüğü	Yaklaşık 12.000
Kendi Oluşturduğumuz Sözlük	140.009

Bilgisayar ağ terimlerinin oluşturulma sürecine, kavram haritalarını oluşturmakla başlanmıştır. Kavramlar arası ilişki düşünüldüğünde kavram haritası kullanarak oluşturulan sözlük başarılı olacağı düşünülmüştür. Oluşturulan sözlük kelimelerin İngilizce karşılıklarının, eş anlam, zıt anlam, yakın anlam ve tanımlarının oluşturulduğu kısımdır. Kelimelerin oluşturulma sürecinde Rıfat Çölkesen' in "Bilgisayar Haberleşmesi ve Ağ Teknolojileri" kitabı içeriği kullanılmıştır.

Çizelge 3.1'de sunulan tüm sözlükler yazılım kullanılabilirliği ve hiyerarşi açısından tek bir txt dosyasında birleştirilmiştir. Tüm sözlüklerde ortak kelime bulunması dikkate alındığında tekrarlı sözcükler süzülerek 140.009 kelimelik sözlük oluşturulmuştur. Sözlüğün yapısının, her ne kadar eş anlam ve yan anlam üzerine oluşturulması düşünülse de, farklı projelere katkı sağlaması amacıyla tanım ve zıt anlamlara da yer verilmesine karar verilmiştir. Sözlük yapısı, bilgisayar ağ terimlerinin değişik anlamlarının önem sıralarına göre hazırlanmıştır.

Kelime | Kelimenin Eş Anlamı | Kelimenin Yakın Anlamı | Kelimenin Zıt Anlamı | Tanımı şeklinde "|" taglarıyla 5 bölüme ayrılmıştır Çizelge 3.2'de oluşturulan sözlüğün bir bölümü görülmektedir.

Çizelge 3.2. Oluşturulan sözlüğün bir bölümü

modem|bağlantı|(null)|(null)|(null)|
bağlantı|mcr|(null)|(null)|(null)|
bağlantı|msr|(null)|(null)|(null)|
mcr|(null)|modem denetim saklayıcısı|(null)|(null)|
mcr|(null)|modem control register|(null)|(null)|
msr|(null)|modem status register|(null)|(null)|
msr|cd|(null)|(null)|(null)|
msr|ri|(null)|(null)|(null)|
msr|dsr|(null)|(null)|(null)|
msr|cts|(null)|(null)|(null)|
msr|cd'|(null)|(null)|(null)|
msr|ri'|(null)|(null)|(null)|
msr|dsr'|(null)|(null)|(null)|
msr|cts'|(null)|(null)|(null)|
çevrim|yerel|(null)|(null)|(null)|
çevrim|uzak|(null)|(null)|(null)|
çevrim|(null)|loop|(null)|(null)|
yerel|(null)|local|(null)|(null)|
yerel|(null)|remote|(null)|(null)|
çevrim|yerel çevrim sınama|(null)|(null)|(null)|
fonksiyon|c|(null)|(null)|(null)|
c|yazan|(null)|(null)|(null)|
c|okuyan|(null)|(null)|(null)|
fonksiyon|(null)|function|(null)|(null)|
c|kesme|(null)|(null)|(null)|
yazan|int import|(null)|(null)|(null)|
yazan|unsigned char importb|(null)|(null)|(null)|
okuyan|unsigned char importb|(null)|(null)|(null)|
kesme|void interrupt|(null)|(null)|(null)|
kesme|void set vect|(null)|(null)|(null)|
osi|(null)|open systems inter connection|(null)|(null)|
osi|başvuru modeli|(null)|(null)|(null)|
osi|referance model|(null)|(null)|(null)|
osi|mimari|(null)|(null)|(null)|
osi|katman|(null)|(null)|(null)|
katman|(null)|layer|(null)|(null)|
katman|uygulama|(null)|(null)|(null)|
katman|sunuş|(null)|(null)|(null)|
katman|oturum|(null)|(null)|(null)|
katman|ulaşım|(null)|(null)|(null)|
katman|ağ|(null)|(null)|(null)|
katman|veri bağı|(null)|(null)|(null)|
katman|fiziksel|(null)|(null)|(null)|
uygulama|(null)|application|(null)|(null)|
sunuş|(null)|presentation|(null)|(null)|
oturum|(null)|session|(null)|(null)|
ulaşım|(null)|transport|(null)|(null)|
veri bağı|(null)|data link|(null)|(null)|

```
veri|(null)|data|(null)|(null)|
bağ|(null)|link|(null)|(null)|
```

3.2. Yapay Zeka

Lenat ve Feigenbaum'un tasvirlerinde; zeka, karmaşık bir sorunu cevaplayabilmek için istenilen verilerin bir araya getirilmesi yeteneği ya da karmaşık bir sorunu, sorunu genel ortamdan özelleiştirerek, cevabın daha basit yoldan bulunabilmesi yeteneğidir. Sözlük anlamında ise, bireyin düşünme, zeka yürütme, gerçekleri algıyabilme, anlayabilme, eleştiride bulunma, sonuca ulaşma becerilerinin tümüdür. Bunun yanında bilgiyi soyut hale getirme, bilgiyi kavrama ve değişik bir durumla karşılaşıldığında bilgiyi kullanma da zeka alanı içinde bulunmajtadır. Yapay zeka ise, aynı özelliklerin, makinelere verilmesi durumudur. Luger ve Stubblefield yaptıkları çalışmalarında yapay zekayı insana özgü davranışların makineleştirilmesi ile ilgilenen bilişim dalı olarak açıklamıştır.

3.2.1. Doğal dil işleme

Doğal dil işlemede amaç, makine ile konuşma dilinde iletişimin sağlanması olduğu için makinelerin öğreneceği dilin kural tabanını algılaması gerekmektedir. Bunun için bilgisayarın genel bir sözlüğe ve bu sözlüğü kullanabilmek için çeşitli algoritmalara ihtiyacı vardır. Bilgisayar konuşma dili ile ilgili genel kurallarla birlikte, ihtiyaç duyduğu ve o lisanın genel yapısından farklı şekilde algılaması gereken belleğe veya göreve özel bir bilgi listelerine de ihtiyacı vardır. Doğal dil işleme sisteminde genel olarak beş temel eleman bulunur. Bunlar, ayrıştırıcı (parser), sözlük (lexicon), anlayıcı, bilgi tabanı ve üreticidir.

3.2.1.1. Türkçe için yapılan doğal dil işleme çalışmaları

Türkçe, eklendiği her ekle birlikte yeni kelime oluşturma özelliğine sahiptir (Eryiğit, 2006). Türkçe dili için yapılan çalışmalarda (Oflazer, 2003; Oflazer, 2003; Hakkani, 2002), sözcükler çekim gruplarına ayrılmıştır. Çalışma alıntılanan

metin üzerinde bulunan kelimelerin eklere ayrılması ve kökleri üzerinde çalışılmıştır.

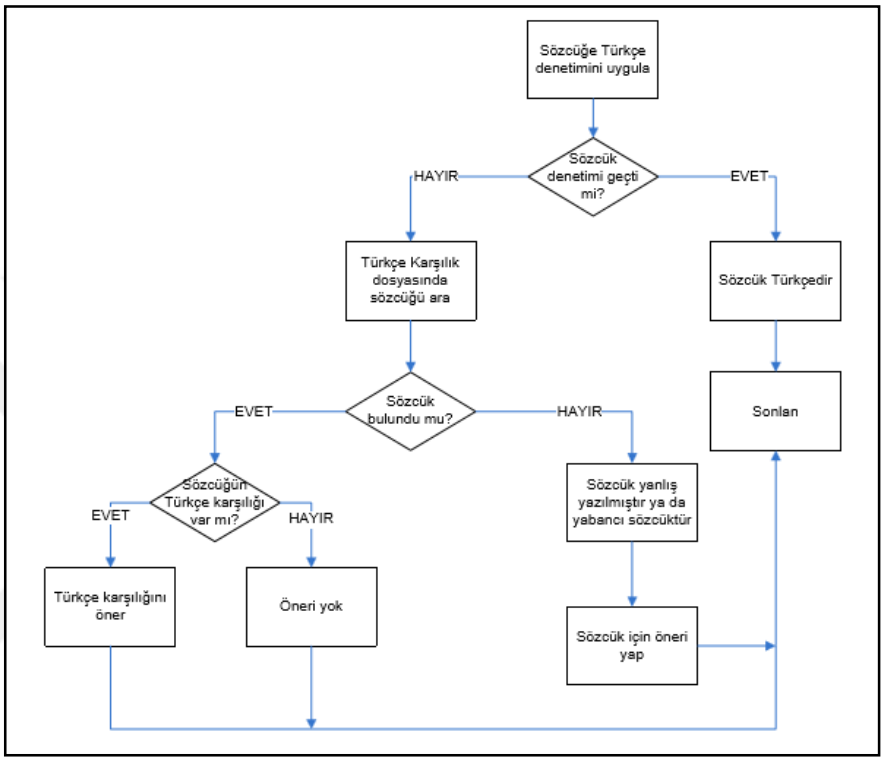
Türkçe Wordnet için yapılan çalışmalardan biri, otomatik çeviri ve şablon metotlarının kullanılmasıyla yapılan çalışmalardır. Çeviri için Wordnet ilişkileri Türkçeleştirilmiştir. Şablon metodu olarak; kavramlar arasındaki her bir ilişki için birer şablon oluşturulmuştur. Şablonların etrafındaki kelimelerin o sınıfa ait olup olmadığına bakılmış; elde edilen örnekler iki sınıfa ayrılmıştır (Amasyalı, 2005).

Güngör yaptığı çalışmada, sözlük tabanlı algoritma ile çalışmış ve başarılı sonuçlar elde etmiştir. Wordnet sözlüğünün tek bir özelliğinden yararlanılmamış, birçok özelliğe bakılmıştır. Geliştirilen algoritmaların en başarılısı; kavramın, tanım cümlesindeki alakasız sözcüklerinden arındırılmasıyla geliştirilen algoritmadır (Güngör, 2007).

Türkçe'nin kavramsal ilişkileri göz önünde bulundurularak hazırlanan çalışmada, sözcükler ek ve köklerine ayrılmıştır. Köklerin sonlarına getirilen, sonsuz sayıda ve sırada ekler, kullanımı zorlaştırdığından dolayı büyük bir sözlükte çalışmak yerine, belli bir kısımda çalışılmıştır. Kelimeler, ek ve köklerine ayrıldıktan sonra, ekler kendi aralarında ek çiftlerini oluşturmuştur. Birden fazla dizimsel ses kuralı olduğu için, temel bir istatistiksel tabanlı çalışma yapılmıştır. Hata düzeltme işlemi için, kelimenin ek ve kökleri kendi kaynak kümelerinde taranmış, hatalı ek ya da kök yerine uygun kaynak kümesine gidilerek değiştirilmesi sağlanmıştır. Bu bağlamda; bulanık mantık benzetimleri, geçmişte yapılan hatalı kavramlar ve kaynak eğitim sırasında toplanan karşılaşılma oranları olmak üzere 3 temel durumdan faydalanılmıştır (Dilsiz, 2005).

Bir başka denetleme çalışmasında, öncelikle kelimenin hecelenebilir olup olmadığına bakılmıştır. Hecelenebilir olmayan sözcükler Türkçe kabul edilmemiş ve kullanıcıya oluşturacağı sözlüğe ekleme imkânı verilmiştir. Hecelenebilir sözcükler, oluşturulan heceleme algoritmasına göre çözümlenmiştir. Bu çözümleme Türkçe ses kurallarına göre bakılıp, en az birine uymayan sözcükler

Türkçeye yabancı dillerden girmiş sözcük kabul edilmiştir. Sözlükte Türkçe karşılıkları bulunan sözcüklere Türkçe karşılıkları önerilmiştir. Çözümlenmesi yapılamayan sözcükler yabancı ya da hatalı olarak kabul edilmiş ve öneriler listelenmiştir. Şekil 3.2'de yapılan çalışmanın sözcük önerme algoritması akış diyagramı gösterilmiştir (Delibaş, 2008).



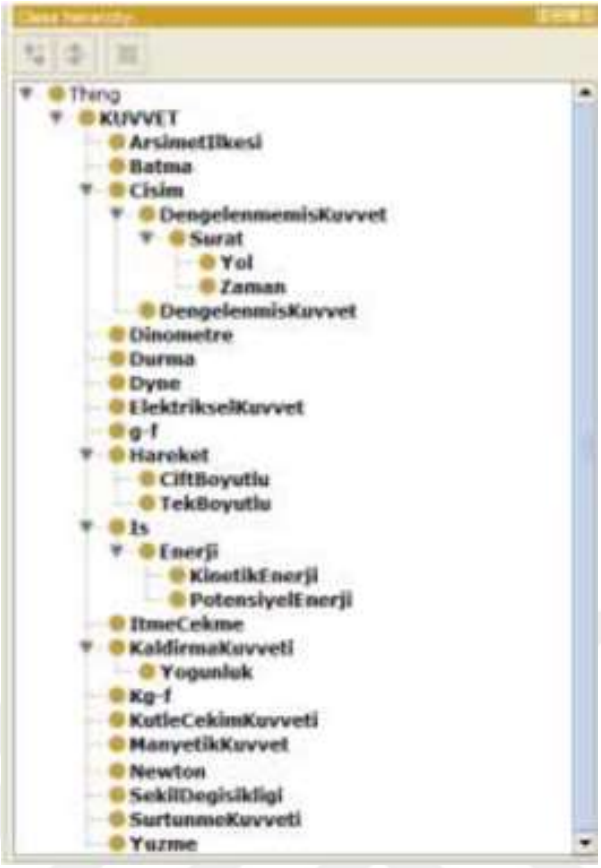
Şekil 3.2. Sözcük önerme algoritması akış diyagramı (Delibaş, 2008)

Oluşturulmuş sözlükten yola çıkıldığında; sözlük ontolojik çözümleme algoritmaları kullanılarak çözümlenebilir. Türk Dil Kurumu için yapılan çözümleme çalışmalarından birinde, kelimenin anlam çıkarımı için, 30 kelimelik kavram dizisine bakılmıştır. Kelimenin 15 sağ ve 15 solunda bulunan kelimelerden bağlam kümesi çıkarılmıştır. Örneğin metin içinde geçen kelime "RAM" olması durumunda, metnin "bilgisayar" ile ilgili olduğu çıkarımı yapılmıştır (Aydın, 2014). Bu çalışmadan yola çıkarak, kavramın eş anlam ve yakın anlam etiketleri kontrol edilerek, ağacın bir üst dalı, kapsayan kavram, alt kavramlarından birinin ana konusu olabileceği kabul edilmiştir.

Gizli Anlambilimsel Dizinleme Metodu (GAD), n gram destekli GAD çalışmasıyla, 2-gram ve 3-gram harfleri terimlerinden oluşturulmuş, GAD yöntemine göre daha anlamlı sonuçlar elde edilmiştir. Araştırma sonucunda oluşturulan terimler, kelimeler kabul edilmiş ve anlamlı birliktelikler oluşturulmuştur (Güven, 2007).

Hung yaptığı çalışmada, alan kümeleme performansını artırmak için Self Organizing Map (SOM) modeline ilave anlamsal kategori bilgisi kullanmıştır. Üç yeni vektör sunum yaklaşımını, yani genişletilmiş anlamlı vektör modeli, hipernim anlamlı vektör modeli ve hibrid vektör uzay modeli önermiştir. Bu üç modelden yola çıkarak, hipernim anlamlı vektör modeliyle, Wordnet ontolojisinden sembolik bilgi çıkartılarak, kümeleme performansının geliştirildiği görülmüştür. WordNet ontolojisi gibi bir ontolojiden gelen bilginin, SOM kümeleme performansını geliştirebildiğini gösterilmiştir (Hung, 2004). Çalışmadan yola çıkarak kavramın, yan anlam etiketiyle kavram haritası üzerindeki ilişkileri incelenerek, kavram çıkarımı yapılabilir.

Gültepe çalışmasında, ders planlanması, uygulanması ve planlanması kavram haritalarının, ontoloji tabanlı olarak oluşturmuştur. Bunun için Protege ontoloji geliştirme editörünü kullanmıştır. Bu ontolojinin grafik arayüzü, 6-7-8. sınıf mekanik konularının kavram haritalarını görsel olarak çıkarmıştır (Gültepe, 2014). Bu sınıflara ait sıra düzenlerinin, ontoloji tabanlı kavram haritası üzerinde gösterimi Şekil 3.3'te gösterilmiştir.



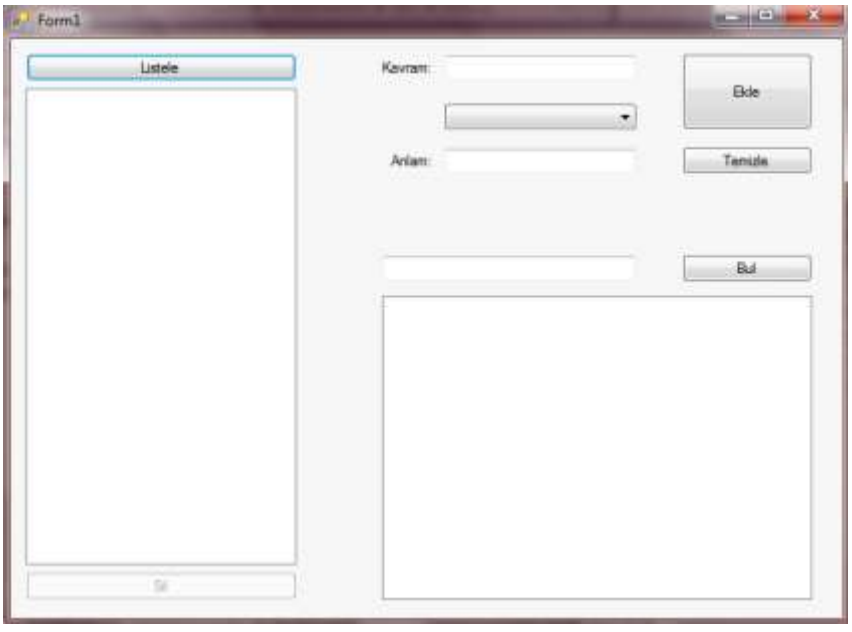
Şekil 3.3. Ontoloji tabanlı kavram haritası üzerinde sıra düzeni (Gültepe, 2014)

ODTÜ-Sabancı Ağaç Yapılı Derlemi üzerinde test edilmiş çalışmada, Türkçe için bağlılık analizinde, sınıflandırıcı olarak karar destek makineleri ve çift bağlılık olasılıklarına dayalı istatistiksel sonuçlar elde edilmiştir (Eryiğit, 2016).

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

4.1. Sözlük Yazılımının Geliştirilmesi

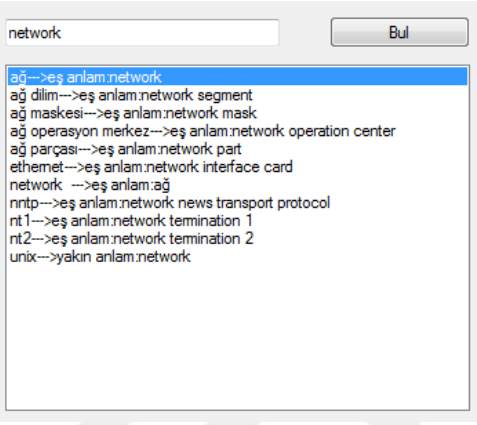
Bu çalışmada, .NET için geliştirilmiş güçlü bir dil olan C# ve bilinen en iyi derleyicisi Visual Studio kullanılarak sözlük yazılımı hazırlanmıştır (Şekil 4.1). C# yazılım dili, Microsoft tarafından geliştirilen bir dildir. Aynı zamanda .NET de Microsoft firması tarafından geliştirilmiş dillerden biridir. Microsoft tarafından geliştirilmiş olsa da European Computer Manufacturers Association (ECMA) ve International Organization for Standardization (ISO) standartları altındadır. C yazılım dilinde verilen bir sayısal değişkeni bir attırmak için sonuna ++ ekleri eklenir. C plus plus (C++) dili adını, C yazılım dilinde nesne yönelimli programlar yazabilmek için kütüphane eklentileri almıştır. Aynı benzerlikle C++ yazılım diline yeni kütüphaneler eklenerek C++ ve ++ oluşumundan bir sonraki uygulama olarak düşünülmüş ve bu defa tamamen nesne yönelimli yazılımlar için tasarlanmış C# yazılım dilinin adlandırılmasında, iki + karakterinin birleştirilmiş halini anımsatan bir müzik sembolü olan C#(majör) kullanılmıştır.



Şekil 4.1. Hazırlanan sözlük yazılımı

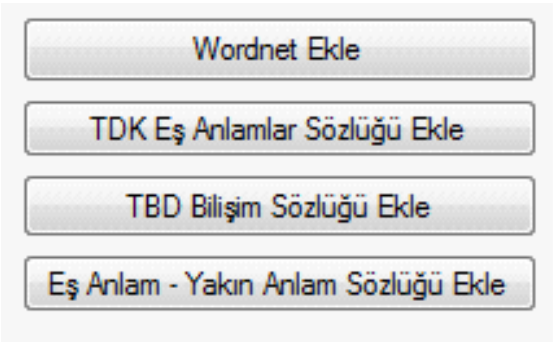
Geliştirilen yazılımın kelime arama kısmında; aranan kelime için öncelikle sözlükte birinci etiket olan kelime bilgisine bakılmakta, daha sonra ikinci ve üçüncü kısım olan eş anlam ve yakın anlam taraması yapılarak sonuç kullanıcıya iletilmektedir.

Örneğin; "network" kelimesi için arama yapıldığında öncelikle "network" olan kelimeler daha sonra "network" ile eş anlamlı ve birinci dereceden ilişkisi olan sözcükler listelenmektedir (Şekil 4.2).



Şekil 4.2. Sözlükte kelime arama

Ayrıca yazılıma farklı kaynaklardan ulaşılarak kullanılan sözlüklerin güncellenmesi halinde bu çalışmada oluşturulan sözlüğe uyarlanması için küçük bir güncelleme parçası eklenmiştir (Şekil 4.3.).



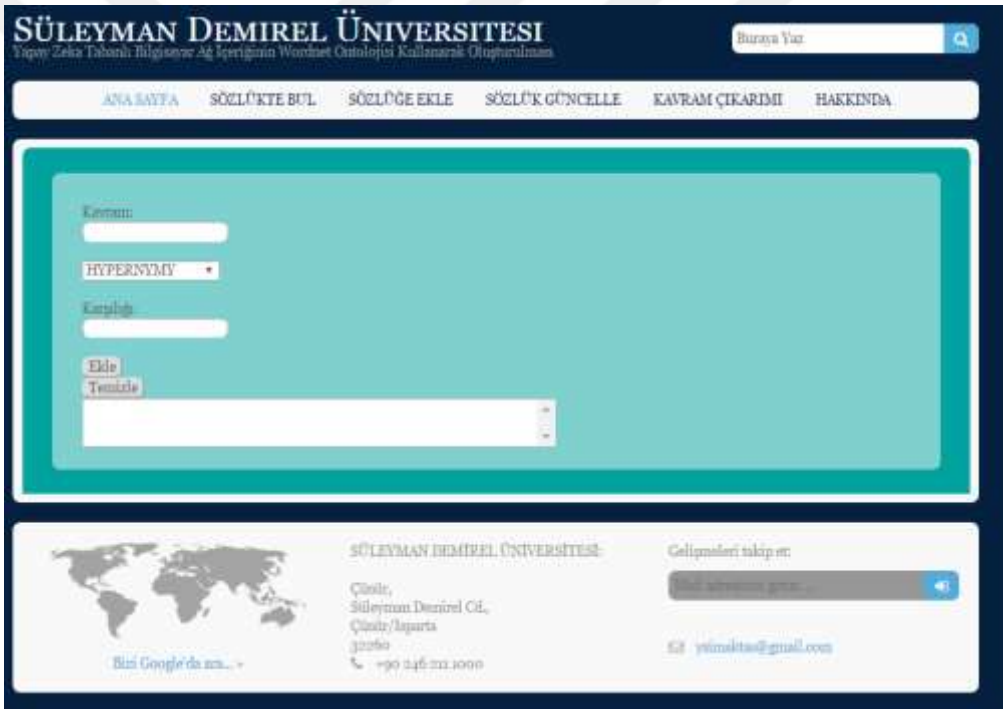
Şekil 4.3. Sözlük güncelleme eki

Oluşturulan C# Form Application sistemi, masaüstü ortamından, web ortamı uygulamasına dönüştürülmüştür (Şekil 4.4). Buradaki amaç, sistem kullanıcılarının işlemleri çevrimiçi yapmasına olanak sağlamaktır.

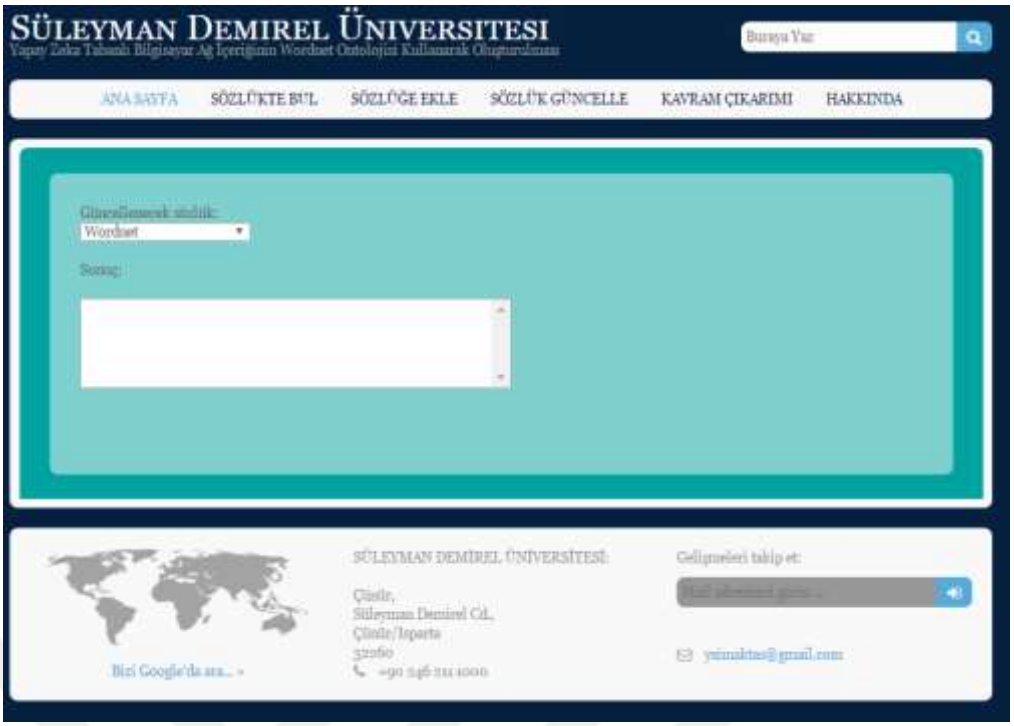
4.1.1. ASP.Net arayüzleri

Oluşturulan ASP.Net sistemi 4 ana parçadan oluşmaktadır. Bunlar:

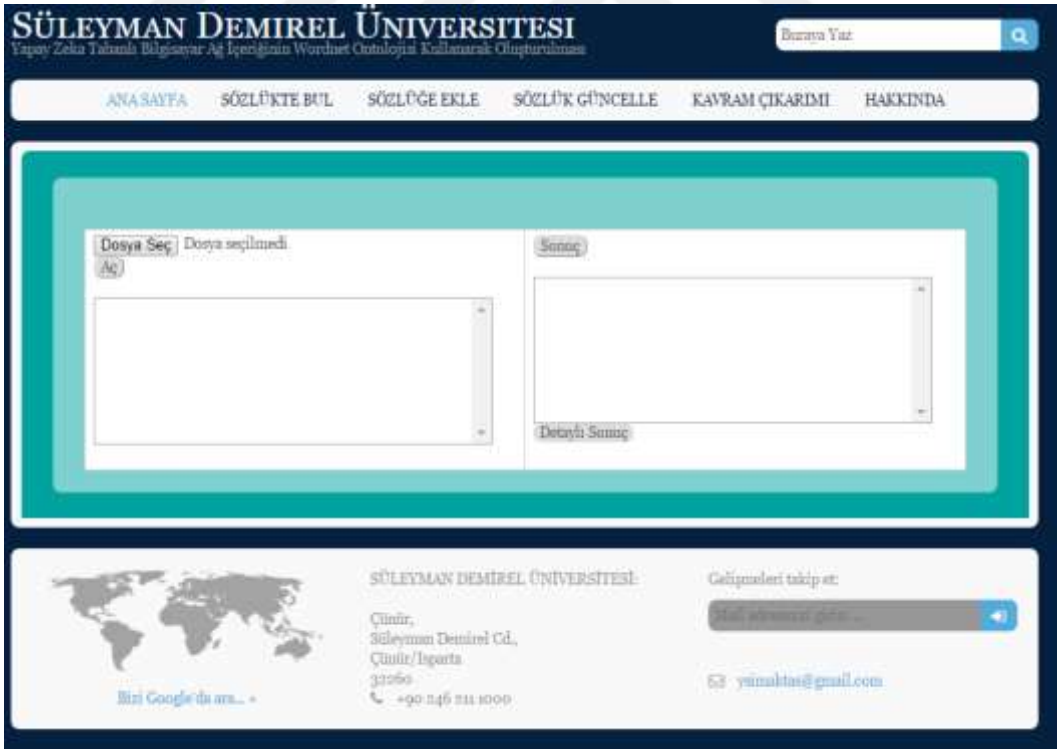
- Sözlükte Bul (Şekil 4.4)
- Sözlüğe Ekle (Şekil 4.5)
- Sözlük Güncelle (Şekil 4.6)
- Kavram Çıkarımı (Şekil 4.7)



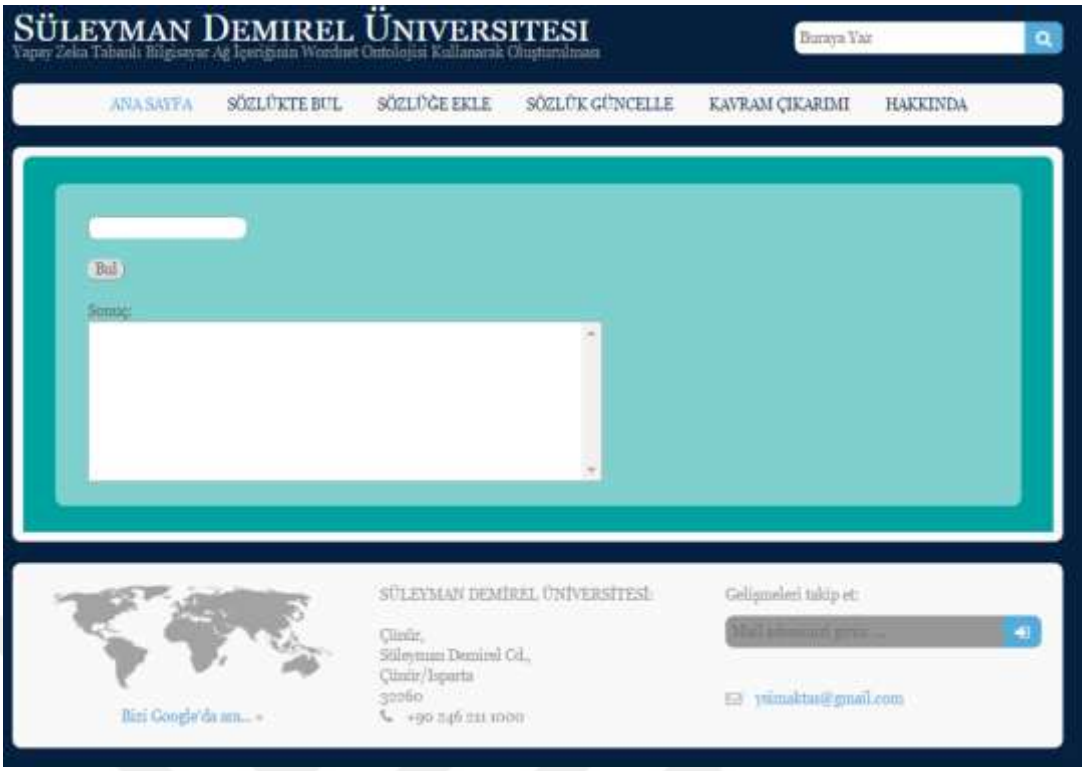
Şekil 4.4. Sözlükte bul



Şekil 4.5. Sözlüğe ekle



Şekil 4.6. Sözlükte güncelle



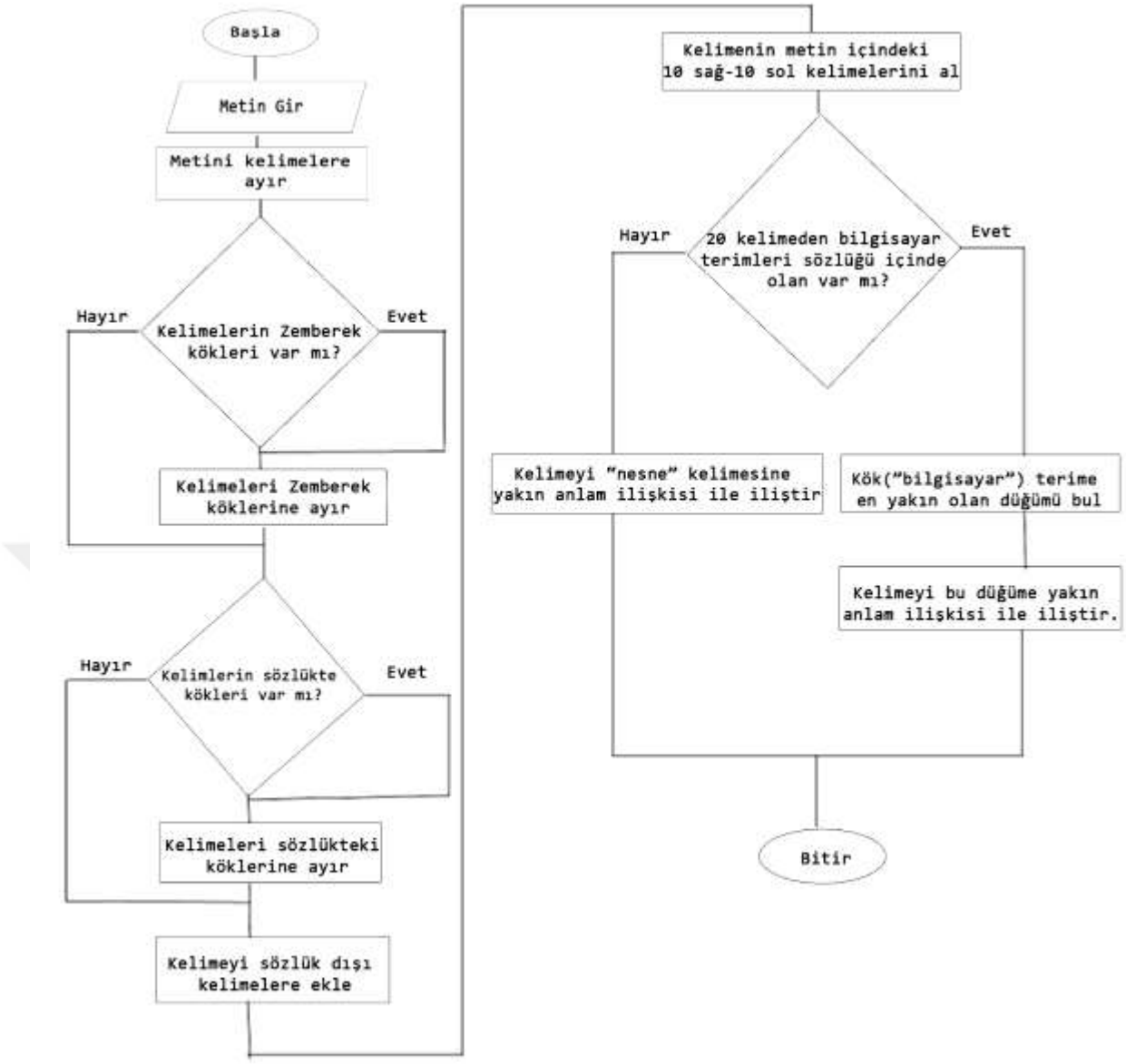
Şekil 4.7. Kavram çıkarımı

4.2. Algoritmanın Oluşturulması

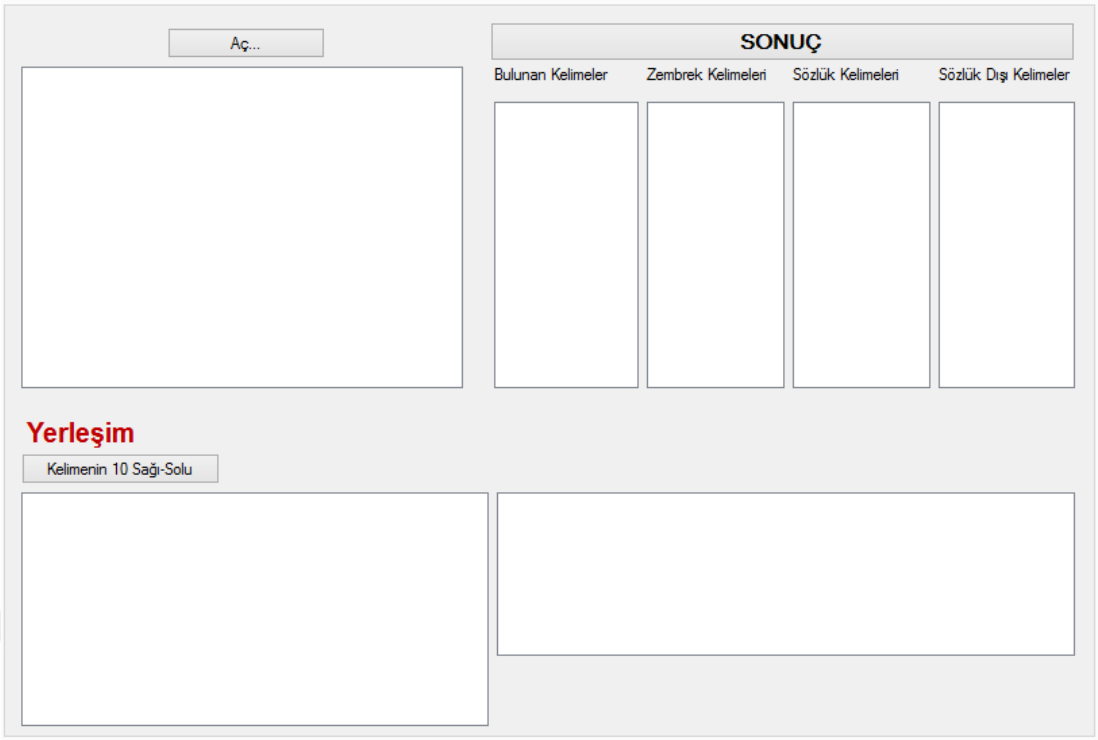
Geliştirilen algoritma iki aşamada hazırlanmıştır. İlk aşama kelimelerin ayrıştırılması kısmıdır. Yazılım .Net uygulamasına göre daha hızlı olan C# form application uygulaması üzerinde oluşturulmuştur.

Kelimenin algoritma kullanılarak kavram haritası üzerinde yerleştirilmesi için paragrafın en az 21 kelimeden oluşması ve kelimenin her iki yönünden birinde en az 10 kelime bulunması şartı vardır.

Geliştirilen algoritma yazılımının akış şeması Şekil 4.8'de, uygulama çalışması Şekil 4.9'da görülmektedir.



Şekil 4.8. Geliştirilen yazılımın Akış Şeması



Şekil 4.9. Algoritma uygulaması

TXT dosyası içerisine alınan paragraf, programa dahil edilmektedir. Paragraf ilk olarak kelimelere ayrıştırılmıştır. Ayrıştırma sonrasında yanlış yazılan kelimeler Zemberek hata düzeltme algoritmasına göre doğrularıyla değiştirilecek şekilde düzenlenmiştir. Bulunan kelimeler liste kutularına aktarılmıştır.

Bulunan kelimeler köksüz ve köklü halinin aynı olma durumu nedeniyle Zemberek ek-kök kütüphaneleri (Zemberek, 2007) yardımıyla köklerine ayrıştırılmış, bulunan kelime kökleri ayrı bir liste kutusuna eklenmiştir.

Zemberek tarafından bulunamayan sözcükler oluşturulan sözlükte aranmış ve bulunanlar bir diğer liste kutusuna eklenmiştir. Zemberek ve sözlük tarafından ayrıştırılamayan son sözcükler son liste kutusuna eklenmiş ve algoritmaya girecek sözlükleri oluşturmuşlardır.

Kelime ayrıştırma işlemlerine örnek Şekil 4.10'da verilmiştir.

Şekil 4.10. Kelime ayrıştırma örneği

Kelimeler ayrıştırıldıktan sonra her kelime için kelimenin 10 sağ ve 10 solunda olmak üzere bulunan kelimelerin birbirlerine uzaklıklarının ortasına yerleştirilmesi algoritması, bu çalışma için bilgisayar terimlerinde kavram haritasının en üst dalı ile en alt dalı arasında 6 uzaklık bulunmasından dolayı, 20 sözcük içinde bilişim sözlüğü içinde bulunan kelimelerde ontoloji ağaç yapısının kök terimine en yakın olan kavramla yakın anlam ilişkisi olacak şekilde iliştirilmiştir. Birbirine yakın anlamlı kelimelerden alt kavram-üst kavram şeklinde oluşan bağıntı arasında beşten fazla uzaklık olmayacak şekilde kavram haritası üzerine eklenmesi şeklinde geliştirilmiştir. Tüm kelimeler ile birlikte algoritmaya giren fakat herhangi bir dala yerleştirilemeyen kelimeler “nesne” kelimesi altına iliştirilmektedir.

Yakın anlam ve eş anlam etiketi ile sözlüğe aktarılmış örnekler Şekil 4.11’de verilmiştir.

```
tcp/ip ağ komutları|(null)|arp|(null)|(null)|
sunucu|server|(null)|(null)|(null)|
```

Şekil 4.11. Sözlüğe eklenmiş kelimeler

Kelimenin yan anlam etiketi, kavram haritasının alt dalını oluşturmaktadır. Örnekte verildiği gibi “arp”, “tcp/ip ağ komutlarının” alt dalıdır.

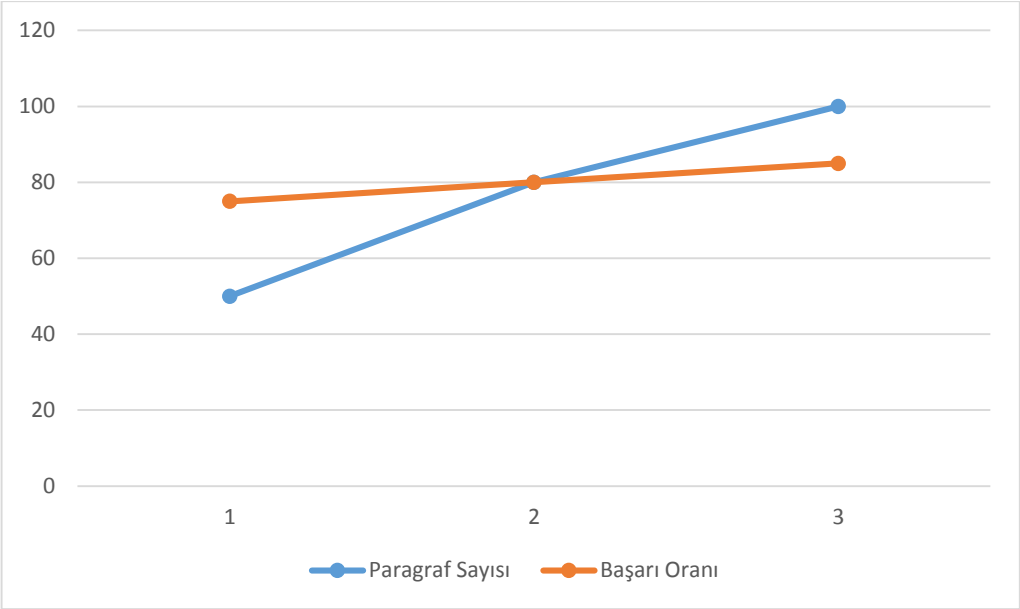
Yapay zekâ, muğlaklık uygulanan projenin sonucunun değerini alabilmek için başarı testleri uygulanmıştır. Başarı testi mevcut doğal dil işleme projelerinde kullanılan doğruluk hesabı şeklindedir. Geliştirilen algoritmanın başarı testi Şekil 4.12’deki formül ile hesaplanmıştır.

$$\boldsymbol{Doğruluk} = \frac{\boldsymbol{Doğru\ Bulunan} + \boldsymbol{Doğru\ Olarak\ Bulumayan}}{\boldsymbol{Doğru\ bulunan} + \boldsymbol{Bulunamayan} + \boldsymbol{Yanlış\ Bulunan} + \boldsymbol{Doğru\ Olarak\ Bulunmayan}}$$

Şekil 4.12. Başarı formülü

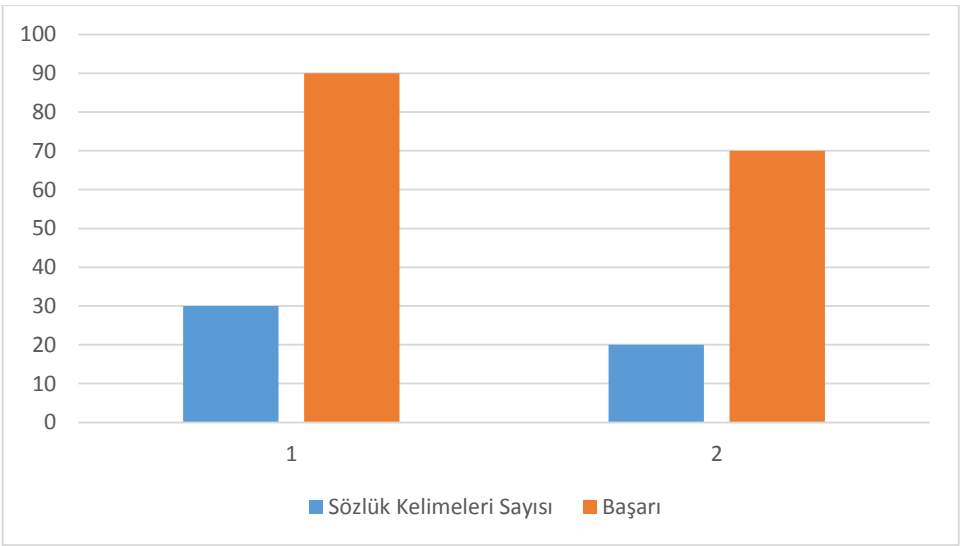
4.2.1. Algoritma sonuçları

Yapay zeka tabanlı bilgisayar ağ içeriğinin wordnet ontolojisi yardımıyla oluşturulması algoritmaları, wordnet ontolojisi dışında, 3 büyük sözlük ve kitaptan edinilen kavramlarla oluşturulan sözlük sayesinde, yeni bir ontoloji tabanı elde edilerek oluşturulmuştur. MEGEP ağ temelleri dökümanları ve ağ-haberleşme çeşitli dökümanları üzerinden alınan farklı paragraflar üzerinde yapılan incelemeler sonucu, kavram haritası üzerinde yerleştirilen kelimelerin doğruluk değerlerinin değişimi grafikler halinde verilmiştir. Elde edilen sonuçlara göre paragrafta kelime sayısı arttıkça ontolojik sözlüğe eklenme sayısı artmakta ve buna bağlı olarak başarı artış göstermektedir (Şekil 4.13). Fakat paragraf sayısındaki artış oranı, içerikteki kelimelerin ek ve köklerine ayrılması durumuna göre değişmektedir. Örneğin köklerine ayrılmamış kelimeler kök olarak kabul edilmekte ve ontolojik tabanda eklerine ayrılmadan yerleşim yapılmaktadır. Bunların yanında bilgisayar ağ kavramlarının çokluğu, kavramın sözlük üzerine eklenmesinde performans açısından büyük önem arz etmektedir. Bilgisayar ağ terimleri sayısı arttıkça, uzaklık ilişkisi bakımından daha sağlam ilişki kurulmakta ve yanlış yerleştirme oranı düşmektedir.



Şekil 4.13. Paragraf sayısına göre başarı oranı değişimi

Şekil 4.14'te, paragrafta bulunan kelimelerin sözlükte bulunma sayısı ve doğruluk değişimi gösterilmiştir. Grafikten görüleceği üzere kelime sayısı arttıkça kelimelerin ontolojik sözlük üzerine yerleştirilme başarıları da artmıştır. En büyük artış oranının bilgisayar kelimelerinin en çok bulunduğu paragraf olduğu tespit edilmiştir. Bu artış oranını, kelimelerin ek ve köklerine ayrılma durumları izlemiştir. Başarıdaki artışının fazla olmasının nedeni, bilgisayar kelimelerinin ve köklerine ayrılmış kelimelerin fazla olmasıdır.



Şekil 4.14. Sözlükte bulunan kelimelere göre başarı değişimi

Sonuç olarak 230 paragraf üzerinde yapılan çalışmalar sonucu, bilinmeyen kelimelerin sözlük üzerine eklenmesinde kullanılan başarı formülü ile %80 başarı elde edilmiştir. Yine aynı formül sonucunda, kelimelerin sözlük üzerinde sözlük kelime sayısının artmasıyla, kelimelerin bulunma oranında artış gözlenmiştir. Örneğin "IP" kelimesi cümle içinde "TCP" ve "katman" ile verilmişse, kelimenin doğru yerleştirilme oranı artmıştır. Buna paralel olarak köklerine ayrılmayan sözcüklerin paragraf içinde bulunması doğru yerleştirme verimini düşürmektedir.

Tüm paragraflarda, kelime sayısı arttıkça, başarı da artmıştır. Teorik hesaplamalarda tüm denemelerin ortak sonucu, çalışmanın başarısı olarak kabul edilmektedir. Fakat deneysel sonuçlara göre paragraf sayısına doğru orantılı olarak başarı artıp, azalmaktadır. Kelimelerin sözlükte bulunma azlığı durumunda çok kelimeli parafraflarda sonuç daha az kelimeli paragraflara göre düşük çıkmıştır.

5. SONUÇ VE ÖNERİLER

Yapılan çalışmaların birçoğu sadece Türkçe dilinin genel kavramları alanında yapılmış olup, bilgisayar kavramları ile yapılan çalışmaların birçoğunda daha düşük başarı elde edilmiştir. Tüm bu analizler ışığında doğal dil işleme tabanlı bilgisayar ağ sözlüğündeki muğlaklığın giderilmesi çalışmaları sonucu sırayla verilmiştir. % 20 başarısızlığa rağmen, algoritmanın denenmiş mevcut algoritmalara oranla daha verimli olduğu gözlenmiştir.

Doğal dil işleme tabanlı ontolojik bilgisayar ağları sözlüğüne veri girişlerinde, kelime sayısı arttıkça başarı artmış, bulanıklık azalmıştır. Düşük veri girişlerinde algoritma düşük hızda çalışırken, kavramların Zemberek ve sözlük içerisinde kök halinde bulunması, algoritmanın uygulanabilirlik düzeyini arttırmıştır. Bilişim kavramlarının otomatik hale getirilmesi konusunda yüksek verim sağlanmıştır.

Algoritmanın ilk uygulanma sırasında, Aydın (2014)'ın çalışmasından baz alınan paragrafta en çok hangi kelime geçiyorsa yazı onunla ilgilidir tezi uygulanmış, bilgisayar kavramları için değiştirilmesi gerektiğine karar verilmiştir. Aydın çalışmasında, Türkçe genel kavramları üzerine odaklanırken, bilgisayar ağ terimleri konusundaki uzaklık bağlantısının, oluşturulan sözlük üzerinde düşük verimde çalıştığı gözlenmiştir.

Geliştirilen algoritma uzaklık ilişkileri eklendiği için DDİ çalışmalarında kullanılması ve geliştirilmesi beklenmektedir. Uzaklık bağlantılarından başka, bilişim içeriğinin İngilizce olması, Türkçe-İngilizce alanında geliştirilecek DDİ algoritmalarıyla birleştirilerek performans daha çok arttırılabilir.

Sonuç olarak DDİ algoritmalarının sözlükler üzerinde yüksek başarı sağlamamasının, kavram madenciliği üzerinde olumsuz etki olduğu düşünülmektedir. Bu etkiler, bu tip sistemlerde algoritma verimlerinin iyileştirilmesiyle azaltılabilir. Bu amaç için, algoritmaları çok daha verimli olarak düzenlemek çok önemlidir. Uzaklık hesaplı algoritmalar verim açısından en uygun yöntemlerin başında gelmektedir. Böyle bir algoritmayla büyük oranda

verim sağlanabilir. Bu çalışma, bilgisayar ağ kavramlarının, ontolojik sözlük oluşturma ve belirsiz kavramların otomatik olarak sözlüğe eklenmesiyle, diğer DDİ çalışmalarına büyük ölçüde katkı sağlayacaktır. Bu çalışmadaki analizlerin, bu incelemelere ışık tutması beklenmektedir. Ayrıca oluşturulan sözlük bilişim ve içerdiği sözlükler bakımından Türkçe oluşturulmuş en büyük sözlüktür.

KAYNAKLAR

Açar, B., 2007. Öğrencilerin Kuvvet Konusundaki Başarılarının Kavram Haritası İle Ölçülmesi, Gazi Üniversitesi Eğitim Bilimleri Enstitüsü, Yüksek Lisans Tezi, 90 s., Ankara.

Adalı, Ş. 2009. An Integrated Architecture for Information Extraction from Documents in Turkish. İstanbul Technical University, Institute of Science and Technology, Doctorate Thesis, 109, İstanbul.

Amasyalı, M., F., 2005. Türkçe Wordnet'in Otomatik Oluşturulması, Bilgisayar Mühendisliği, Yıldız Teknik Üniversitesi, İstanbul.

Anderson-Inman, L., Ditson, L., 1999. Computer-BasedConceptMapping: A ToolForNegotiatingMeaning. Learning AndLeadingWithTechnology, 26(8), 6-13.

Aydın, C.R., Erkan, A., Güngör, T., Takçı, H., 2014. Sözlük Kullanarak Türkçe için Kavram Madenciliği Metotları Geliştirme, Akademik Bilişim, Mersin.

Aykanat, F., Doğru, M., Kalender, S., 2005. Bilgisayar Destekli Kavram Haritaları Yöntemiyle Fen Öğretiminin Öğrenci Başarısına Etkisi, Kastamonu Eğitim Dergisi, 13(12), 391-400.

Bilgin, O., Cetinoğlu, O., & Oflazer, K., 2004. Building a WordNet or Turkish, Romanian Journal of Information Science and Technology, 7.1-2, 163-172.

Bilişim Sözlüğü, Türkiye Bilişim Derneği. Erişim Tarihi:24.04.2017, http://www.tbd.org.tr/index.php?sayfa=sozluk.

Bloehdorn, S., Hotho, A., 2006. Boosting for Text Classification with Semantic Fetures. AdvancesIn Web MiningAnd Web Usage Analysis, Proceedings of the 6th internationalconference on Knowledge Discovery on the Web, BerliniHeidelberg.

Can, Ö., Ünalır, M.O., 2010. Ontoloji Tabanlı Erişim Denetimi. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 197-206, Denizli.

Cebiroğlu, G., 2002. Sözlüksüz Köke Ulaşma Yöntemi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 86s, İstanbul.

Chang, K., Sung, Y., 2001. Learning Thorough Computer Based Concept Mapping With Scaffolding Aid. Journal of Computer Assisted Learning, 17, 21-33.

Chua, S., Kulathuramaiyer, N., 2004. SemanticFeatureSelection Using Wordnet. Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, 4, 166-172.

Cortes, C., Vapnik, V., 1995. Support-Vector Networks. Machine Learning, 20(3), 273-297.

Çelik, K., 2012. A Comprehensive Analysis Of Using Wordnet, Part-Og-Speech Tagging, And World Sense Disambiguation In Text Categorization, Boğaziçi Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, 94s., İstanbul.

Çölkesen, R., Örencik, B., 2003. Bilgisayar Haberleşmesi ve Ağ Teknolojileri, Papatya Yayıncılık, 1448s, İstanbul.

Delibaş, A., 2008. Doğal Dil İşleme İle Türkçe Yazım Hatalarının Denetlenmesi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi.

Dilsiz, S., 2005. Bulanık Mantık ve Yapay Sinir Ağları İle Türkçe Yazım Denetleyicisi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi.

Diri, B., Doğal Dil İşleme (DDĐ) Natural Language Processing (NLP), Eirişim Tarihi: 24.04.2017, https://www.ce.yildiz.edu.tr/personal/banud/file/2625/Course+Overview-1.pdf.

Düzağaç, R., 2014. Improving Search Engine PerformanceWith Context Extraction Using Lucene, DBpedia-Sportlight, AndWordnet, Işık Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 57s., İstanbul.

Eryiğit, G., Adalı, E., Oflazer, K., Türkçe Cümlelerin Kural Tabanlı Bağlılık Analizi, Erişim Tarihi: 24.04.2017, http://research.sabanciuniv.edu/1210/1/3011800000624.pdf.

Eryiğit, G., Oflazer, K., 2006. Statistical dependency parsing of Turkish. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, Trento, Italya.

Fellbaum, C., 1998, WordNet: An electronic lexical database. , Cambridge, MA: MIT Press,423.s.

Fırat, Y., 2017. Modellenmiş Kavram Kafesleriyle Bilgisayarlı Framenet Çalışması, e, Journal of Engineering Sciences and Design, 5(1), 317-329.

Forman, G., 2003. An ExtensiveEmpiricalStudy of FeatureSelectionMetricsforTextClassification. Journal of Machine Learning Research, 3, 12891305.

Gruber, T.R., A Translation Approach To Portable Ontology Specifications, Knowl. Acquis. 5, 199-220, 1993.

Gültepe, Y., Memiş, E. K., 2014. Kavram Haritalarının Ontoloji Tabanlı Oluşturulması: Kuvvet Konusu Uygulama Örneği, Journal of Instructional Technologies & Teacher Education, 24-33.

Güner, E.S., 2005. Makine Çevirisinde Yeni Bir Bilgisayımsal Yaklaşım, Yayımlanmış doktora tezi, Trakya Üniversitesi, Edirne.

Güngör, O., Güngör, T., 2007. Türkçe Bir Sözlükteki Tanımlardan Kavramlar Arasındaki ÜstKavram İlişkilerinin Çıkarılması, Akademik Bilişim Konferansı Bildirileri, 31 Ocak-2 Şubat 2007, Kütahya.

Güven, A., 2007. Türkçe Belgelerin Anlam Tabanlı Yöntemlerle Madenciliği, Doktora Tezi, Yıldız Teknik Üniversitesi, İstanbul.

Hakkani, D.Z., Tür, G., Oflazer, K., 2002. Statistical morphological disambiguation for agglutinative languages. Computers and the Humanities 36(4), 381—410.

Hearts, M., A., 1998. Automated discovery of Wordnet Relations, In C. Felbaum, editor, Wordnet: an Electronic Lexical Database, MIT Press.

Hung, C., Wermter, S., 2004. Neural Network-based Document Clustering Using WordNet Ontologies, International Journal of Hybrid Intelligent Systems.

Jonessen, D.H., Garabowski, B.L., 1993. HandbookFfIndivitualDiferences: Learning andIntruction. Hove. LEA.

Kabaca, T., 2002. Ortaöğretim Matematik Eğitiminde Kavram Haritalanması Tekniğinin Kullanımı. Marmara Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 82 s., İstanbul.

Kapucu, N., 2008. Bilgisayar Destekli Kavram Haritasının Kullanımının, Öğrencilerin Bilişsel Senaryo Oluşturma Becerileri, Erişi Öğrenmelerinin Kalıcılığı Ve Derse Yönelik Tutumları Üzerindeki Etkileri, Muğla Üniversitesi Sosyal Bilimler Enstitüsü, Yüksek Lisans Tezi, Muğla.

Karadeniz, Z.İ., 2007. Türkçe için Biçimbilimsel Belirsizlik Giderici. İstanbul Teknik Üniversitesi. Fen Bilimleri Enstitüsü. Yüksek Lisans Tezi, 67, İstanbul.

Lenat, D.B., 1979. On automated scientific theory formation: A case study using the AM program. In J. Hayes, D. Michie and L.I. Mikulich (Eds.) Machine Intellligence 9, (251-283). New York: Halstead.

Li, J., Zhao, Y., Liu, B., 2009. Fully Automatic Text Categorization by Exploiting Wordnet. Proceedings of th 5th Asia Information Retrieval Symposium on Information Retrieval Technology, Berlin, Heildberg.

Mahcup, T., Adalı, E., 2010. Türkçede Eklerin Kazandırdığı Anlamlar, Bilgisayar Bilimleri ve Mühendisliği Dergisi.

McGıwen, M.,Tall, D., 1999, Concept Maps And Schmatic Diagrams As Devices For Dpcumenting The Growth of Matematical Knowledge. Mathematic Education, 34, 717-733.

Nabdel, L., Karataş, A.S., Oğuztüzün, H., Doğru, A., 2011. A Feture Model Markup Language In Numerical Analysis And Applied Mathematics International Conference on Numerical Analysis And Applied Mathematics, 2011, AIP Publishing, 841-844.

Oflazer, K., 2003. Dependency parsing with an extended finite-state approach. Computational Linguistics 29(4).

Oflazer, K., Say, B., Hakkani, D.Z., Tür, G., 2003. Building a Turkish treebank. In Abeille, A., ed.: Building and Exploiting Syntactically-annotated Corpora. Kluwer Academic Publishers.

Per, M.U., 2011. Developing A Concept Extraction System for Turkish. The Middle East Technical University. The Institute of Informatics, Master Thesis, 59, Ankara.

Savaşçı, A., 2010. Türkçe Bitişkenlik Derecesinin İstatiksel Verilerle Belirenmesi. Ege Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 58, İzmir.

Sebastiani, F., 2002. Machine Learning in AutomatedTextCategorizatio. ACM Computing Surveys, 34(1), 1-47.

Soysal, E., 2010. Ontology Based Information Extraction on Free Text Radiological Reports Using Natural Language Processing Approach. The Middle East Technical University, The Institute of Informatics, Doctorate Thesis, 110, Ankra.

Şahin, F., 2001. Öğretmen Adaylarının Kavram Haritası Yapma Ve Uygulama Hakkındaki Görüşleri. Pamukkale Üniversitesi Eğitim Fakültesi Dergisi, 10, 12-24.

Takçı, H., Soğukpınar, İ., 2004. Centroid-Based Language Identification Using Letter Feature Set, International Conference on Intelligent Text Processing and Computational Linguistics, 5th International Conference, 640-648s, Kore.

Tufis, D., Cristea, D., & Stamou S., 2004. BalkaNet: Aims, Methods, Results and Perspectives: A General Overview, Romanian Journal of Information science and technology, 7.1-2, 9-43.

Türk Dil Bilgisi, 2016. Kök Nedir? İsim Kökleri ve Fiil Kökleri, Erişim Tarihi: 24.04.2017, http://www.turkdilbilgisi.com/sozcukte-yapi/kok-nedir-isim-kokleri-fiil-kokleri.html.

Türk Dil Kurumu Ana Sayfası, Türk Dil Kurumu. Erişim Tarihi: 24.04.2017, http://www. tdk.gov.tr/.

Türkçe Eş Anlamlar Sözlüğü, Erişim Tarihi: 20.08.2016, https://github.com/maidis/mythes-tr.

Vossen, P., 1998. EuroWordNet: A Multilingual Database with Lexical Semantic Networks, Computational Linguistics, Vol. 25, 427–430.

XML Tutorial, 2011. Erişim Tarihi: 24.04.2017. http://www.w3schools.com/xml/default.asp.

Yaşar, C., 2007. Wordnet Üzerinde Türkçe Bilişim Ontolojisinin Oluşturulması, Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 68s, Çanakkale.

Yavanoğlu, U., Sağıroğlu, Ş. 2010. Web Tabanlı Otomatik Dil Tanıma ve Çevirme Sistemi, Gazi Üniversitesi Mğhendislik Fakültesi Dergisi, 483-494.

Yılmaz İnce, E., 2016. Web Ortamındaki Yazılı Sınavların Doğal Dil İşleme Yöntemleri İle Değerlendirilmesi, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, 117s, Isparta.

Yılmaz, E., Tamer, S.L., Koç, M., 2009. Öğretmen Adaylarının Kavram Haritalarının Arayüz Tasarımındaki Görsel Tercihleri. SDU International Journal of TechnologicalScience, 1(1), 4157.

Zemberek,2007. Erişim Tarihi: 24.04.2017, https://code.google.com/archive/p/nzemberek/downloads.

EKLER

EK A. Metin Girişi

EK B. Kelimelerin Ayrılması

EK C. Kelimelerin Zemberek Köklerine Ayrılması

EK D. Kelimelerin Sözlük Köklerine Ayrılması

EK E. Bulunmayan Kelimelerin Listelenmesi

EK F. Bulunmayan Kelimeler Üzerinde Algoritmanın Uygulanması

EK G. ASP.Net Kodları

EK H. Yazılım Üzerinde Denenmiş Metinler

EK I. Fotoğraflar

EK A. Metin Girişi

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using net.zemberek.erisim;
using net.zemberek.tr.yapi;
using net.zemberek.yapi;
using net.zemberek.yapi.ek;
using net.zemberek.yapi.kok;

namespace tdk_15_kavram
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            temizle();
            dosyaAc();
        }



        public void temizle()
        {
            listBox1.Items.Clear();
            listBox2.Items.Clear();
            listBox3.Items.Clear();
            listBox4.Items.Clear();
            listBox5.Items.Clear();
        }
        public void dosyaAc()
        {
            OpenFileDialog op = new OpenFileDialog();

            if (op.ShowDialog() == DialogResult.OK)
            {

                StreamReader oku = new StreamReader(op.FileName,
Encoding.GetEncoding(1254));
                string veri = oku.ReadLine();
                while (veri != null)
                {
                    listBox1.Items.Add(veri);
                    veri = oku.ReadLine();

                }
            }
        }
```

EK B. Kelimelerin Ayrılması

```
public void kelimeleriListele()
        {
            string[] karakterler = { "?", "!", "_", "-", "+", "*", "/", ".",
",", ";", ":" ,"(",")"};
            if (listBox1.Items.Count != 0)
            {
                string kelime;
                for (int i = 0; i < listBox1.Items.Count; i++)
                {
                    kelime = listBox1.Items[i].ToString();
                    string[] parcala = kelime.Split(' ');
                    for (int y = 0; y < parcala.Count(); y++)
                    {
                        for (int z = 0; z < karakterler.Count(); z++)
                        {
                            parcala[y]=parcala[y].Replace(karakterler[z], "");
                        }

                        parcala[y] = parcala[y].Replace("I", "i");
                        parcala[y] = parcala[y].ToLower();
                        listBox2.Items.Add(parcala[y]);

                    }
                }

            }

        }

        public void ozelBul()
        {
            string kelime1 = "";
                for (int i = 0; i < listBox2.Items.Count; i++)
                {
                    kelime += listBox2.Items[i].ToString() + " ";
                }


                string kelimeninSag = "";
                string kelimeninSol = "";

                string[] parcala = kelime.Split(' ');
                for (int y = 0; y < parcala.Count(); y++)
                {
                    if (parcala[y].StartsWith(kelime1))
                    {
                        kelimeninSag = "";
                        kelimeninSol = "";
                        for (int a = 1; a <= 5; a++)
                        {
                        kelimeninSag += parcala[y + a] + "||";
                        sagKelimeYanAnlam(parcala[y + a]);
                        kelimeninSol += parcala[y - a] + "||";
                        sagKelimeYanAnlam(parcala[y - a]);
                    }
                        listBox6.Items.Add(kelimeninSol + "-->" + kelime1 +
"<--" + kelimeninSag);
```

```
                    }

                }

        }

        public void asilKelimeleriListele()
        {
           //Listbox1'in satır satır okunması
           for (int i = 0; i < listBox1.Items.Count; i++)
                       {
                          string kelime = "";
                         // Her satırın uzunluğu kadar döngü oluşturulması
                           for (int y = 0; y <
listBox1.Items[i].ToString().Length; y++)
                           {
                               if (listBox1.Items[i].ToString().Substring(i,
1) != " ")
                               {
                                   kelime +=
listBox1.Items[i].ToString().Substring(i, 1).ToString();
                               }
                               else
                               {
                                   listBox2.Items.Add(kelime);
                                   kelime = "";
                               }

                           }
                   }
        }

        public void basarim()
        {


        }
```

EK C. Kelimelerin Zemberek Köklerine Ayrılması

```
        public void zemberekAyır()
        {
            //______..::Zemberek Köklerine
Ayrıldı::..__________________________

            if (listBox2.Items.Count != 0)
            {
                string cozulecek;
                for (int i = 0; i < listBox2.Items.Count; i++)
                {
                    cozulecek = listBox2.Items[i].ToString();
                    Zemberek zbr = new Zemberek(new TurkiyeTurkcesi());
                    Kelime[] kelimeler = zbr.kelimeCozumle(cozulecek);
                    foreach (Kelime klm in kelimeler)
                    {
                        string[] kokBol = klm.kok().ToString().Split(' ');
                        listBox3.Items.Add(kokBol[0].ToString());
                        break;
                    }
                }

            }



            int a = 0;
            for (int i = 0; i < listBox2.Items.Count; i++)
            {
              a = 0;
            //    //bilgisayar-donanım
               for (int y = 0; y < listBox3.Items.Count; y++)
               {
            //        //bilgisayar
                    if (listBox3.Items[y] == listBox2.Items[i])
                   {
                        a++;
                        break;
                    }
                }
                if (a == 0)
                {
                    listBox4.Items.Add(listBox2.Items[i].ToString());
                }
            }
        }
```

EK D. Kelimelerin Sözlük Köklerine Ayrılması

```
public void bulunanKelimeler()
        {
            //_____..::Sözlükte Bulunanlar Listelendi::.._____________________

            if (listBox3.Items.Count != 0)
            {
                for (int i = 0; i < listBox3.Items.Count; i++)
                {
                    StreamReader ac = new
StreamReader("sozlukler/hepsiii.txt", Encoding.GetEncoding(1254));
                    string kelime = ac.ReadLine();
                    while (kelime != null)
                    {
                        string[] sozlukBol = kelime.Split('|');
                        if (sozlukBol[0] == listBox3.Items[i].ToString() ||
sozlukBol[1] == listBox3.Items[i].ToString())
                        {
                            listBox4.Items.Add(listBox3.Items[i].ToString());
                            break;
                        }
                        kelime = ac.ReadLine();
                    }
                    ac.Close();
                }
            }


        }
```

EK E. Bulunmayan Kelimelerin listelenmesi

```
public void bulunmayanKelimeler()
{
    int sayac = 0;
    for (int i = 0; i < listBox3.Items.Count; i++)
    {
        sayac = 0;
        for (int y = 0; y < listBox4.Items.Count; y++)
        {
            if (listBox3.Items[i] == listBox4.Items[y])
            {
                sayac++;
            }

        }
        if (sayac == 0)
        {
            listBox5.Items.Add(listBox3.Items[i]);
        }
    }

}
```

EK F. Bulunmayan Kelimeler Üzerinde Algoritmanın Uygulanması

```
        public void yerlestirme()
        {
            int[] sagDizi =[10];
            int[] solDizi =[10];

            for (int i = 0; i < listBox4.Items.Count; i++)
            {
                for (int y = 0; y < 2; y++)
                {
                    for (int x = 0; x < listBox1.Items.Count; x++)
                    {
                        if (listBox1.Items[x] == listBox4.Items[i])
                        {

                        }
                    }
                }
            }
        }


        public void kelimenin15sagSol()
        {
            string kelime1 = "";

            for (int yy = 0; yy < listBox5.Items.Count; yy++)
            {
                kelime1 = listBox5.Items[yy].ToString();

                string kelime=
            for(int i = 0; i < listBox2.Items.Count; i++)
            {
                kelime += listBox2.Items[i].ToString()+" ";
            }

            string kelimeninSag = "";
            string kelimeninSol = "";

            string[] parcala = kelime.Split(' ');
            for (int y = 0; y < parcala.Count(); y++)
            {
                if (parcala[y].StartsWith(kelime1))
                {
                    kelimeninSag = "";
                    kelimeninSol = "";
                    for (int a = 1; a <= 1; a++)
                        {
                            kelimeninSag += parcala[y + a] + "||";

                            sagKelimeYanAnlam(parcala[y + a]);
                            kelimeninSol += parcala[y - a] + "||";

                            sagKelimeYanAnlam(parcala[y - a]);
                        }
                    listBox6.Items.Add(kelimeninSol + "-->" + kelime1 + "<--"
+ kelimeninSag);

                }
            }
```

```
            }
        }

        public void sagKelimeYanAnlam(string kelime)
        {
                    StreamReader ac = new StreamReader("sozlukler/tbdx.txt",
Encoding.GetEncoding(1254));
                    string aa = ac.ReadLine();

                    while (aa != null)
                    {
                        string[] sozlukBol = aa.Split('|');
                        if (sozlukBol[0] == kelime || sozlukBol[1] ==kelime)
                        {
                            listBox7.Items.Add(sozlukBol[0]+"-
"+sozlukBol[1]+"-"+sozlukBol[2]);
                        }

                        aa = ac.ReadLine();
                    }
                    ac.Close();


        }




        public void zemberekHataDuzeltme()
        {
            //_____________________________________________________________
            // Zemberek hata düzetlme
            //_____________________________________________________________

            string duzelecek;
            for (int i = 0; i < listBox2.Items.Count; i++)
            {
                duzelecek = listBox2.Items[i].ToString();
                Zemberek zbr = new Zemberek(new TurkiyeTurkcesi());
                string[] oneriler = zbr.oner(duzelecek);
                for (int y = 0; y < oneriler.Count(); y++)
                {
                    listBox6.Items.Add(duzelecek + "--->" + oneriler[y]);
                }

            }

        }

        public void zeberekEkKok()
        {
            //_____________________________________________________________
            // Zembere Kök-Ek Çözümlemesi
            //_____________________________________________________________

            string cozulecek;
            for (int i = 0; i < listBox2.Items.Count; i++)
            {
```

```
                cozulecek = listBox2.Items[i].ToString();
                Zemberek zbr = new Zemberek(new TurkiyeTurkcesi());
                Kelime[] kelimeler = zbr.kelimeCozumle(cozulecek);
                foreach (Kelime klm in kelimeler)
                {
                    string[] kokBol = klm.kok().ToString().Split(' ');
                    listBox3.Items.Add(cozulecek+ "Kökü:
"+klm.kok().ToString());
                    listBox3.Items.Add(cozulecek + "Boyu: " +
klm.boy().ToString());
                    listBox3.Items.Add(cozulecek + " Ekler: ");
                    foreach (Ek ekk in klm.ekler())
                    {
                        listBox3.Items.Add(cozulecek + " --> " + ekk.ad());
                    }
                    listBox3.Items.Add("____________________");

                }
            }


        }

        private void button2_Click(object sender, EventArgs e)
        {
            kelimeleriListele();
            zemberekAyır();
            bulunanKelimeler();
            bulunmayanKelimeler();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ozelBul();
        }
    }
}
```

EK G. ASP.Net Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Collections;


#region copyright
//Yeşim AKTAŞ © 2015
#endregion

namespace SözlükYazılım
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            Listele();
        }
        public void Listele()
        {
            listBox1.Items.Clear();
            StreamReader oku = new StreamReader("bil.txt",
Encoding.GetEncoding("utf-8"), false);
            string yazi;
            string[] p;
            int i = 0;
            ArrayList dizi = new ArrayList();
            while ((yazi = oku.ReadLine()) != null)
            {
                p = yazi.Split('|');
                dizi.Add(p[0].ToString());
                i++;
            }
            oku.Close();
            for (int x = 0; x < dizi.Count; x++)
            {
                if (x == 0)
                {
                    listBox1.Items.Add(dizi[x].ToString());

                }
                if (x != 0 && dizi[x - 1] != dizi[x])
                {
                    listBox1.Items.Add(dizi[x].ToString());

                }
```

```
            }
            listBox1.Sorted = true;
            listBox1.Refresh();
            int indeks;
            int sayi1 = listBox1.Items.Count;
            if (sayi1 > 1)
            {
                string sonitem = listBox1.Items[sayi1 - 1].ToString();
                for (indeks = sayi1 - 2; indeks >= 0; indeks += -1)
                {
                    if (listBox1.Items[indeks].ToString() == sonitem)
                    {
                        listBox1.Items.RemoveAt(indeks);
                    }
                    else
                    {
                        sonitem = listBox1.Items[indeks].ToString();

                    }

                }
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            Sil();
        }
        public void Sil()
        {
            var file = new
List<string>(System.IO.File.ReadAllLines("hepsi.txt"));
            file.RemoveAt(listBox1.SelectedIndex);
            File.WriteAllLines("hepsi.txt", file.ToArray());
        }

        public void Bul()
        {
            listBox2.Items.Clear();
            StreamReader oku = new StreamReader("hepsiii.txt",
Encoding.GetEncoding("UTF-8"), false);
            string kel = textBox3.Text.ToLower();
            string yazi;
            string[] p;
            while ((yazi = oku.ReadLine()) != null)
            {
                if (yazi.StartsWith(kel))
                {
                    p = yazi.Split('|');
                    if (p[1].ToString() != "(null)")
                    {
                        p[1] = "--->yakın anlam:" + p[1].ToString();
                    }
                    else
                        p[1] = "";
                    if (p[2].ToString() != "(null)")
                    {
                        p[2] = "--->eş anlam:" + p[2].ToString();
                    }
                    else
                        p[2] = "";
                    if (p[3].ToString() != "(null)")
```

```
                    {
                        p[3] = "--->zıt anlam:" + p[3].ToString();
                    }
                    else
                        p[3] = "";
                    if (p[4].ToString() != "(null)")
                    {
                        p[4] = "--->tanım:" + p[4].ToString();
                    }
                    else
                        p[4] = "";
                    listBox2.Items.Add(p[0].ToString() + " " + p[1].ToString()
+ " " + p[2].ToString() + " " + p[3].ToString() + " " + p[4].ToString());
                }
            }
            oku.Close();

            StreamReader oku2 = new StreamReader("hepsiii.txt",
Encoding.GetEncoding("UTF-8"), false);
            string kel2 = textBox3.Text.ToLower();
            string yazi2;
            string[] p2;
            while ((yazi2 = oku2.ReadLine()) != null)
            {
                p2 = yazi2.Split('|');
                if (p2[2].StartsWith(kel2))
                {
                    listBox2.Items.Add(p2[0].ToString() + "--->eş anlam:" +
p2[2].ToString());
                    if (p2[1].ToString() != "(null)")
                    listBox2.Items.Add(p2[2].ToString() + "--->yakın anlam:" +
p2[1].ToString());
                }
                if (p2[1].StartsWith(kel2))
                {
                    listBox2.Items.Add(p2[0].ToString() + "--->yakın anlam:" +
p2[1].ToString());
                    if (p2[2].ToString() != "(null)")
                        listBox2.Items.Add(p2[1].ToString() + "--->eş anlam:"
+ p2[2].ToString());
                }
            }
            oku2.Close();




            listBox2.Sorted = true;
            listBox2.Refresh();
            int indeks;
            int sayi1 = listBox2.Items.Count;
            if (sayi1 > 1)
            {
                string sonitem = listBox2.Items[sayi1 - 1].ToString();
                for (indeks = sayi1 - 2; indeks >= 0; indeks += -1)
                {
                    if (listBox2.Items[indeks].ToString() == sonitem)
                    {
                        listBox2.Items.RemoveAt(indeks);
                    }
                    else
```

```
                    {
                        sonitem = listBox2.Items[indeks].ToString();

                    }

                }
            }
        }

        public void Ekle()
        {
            if (textBox1.Text != "" && textBox2.Text != "" && comboBox1.Text
!= "")
            {

                StreamWriter Dosya = File.AppendText("bil.txt");
                if (comboBox1.Text == "HYPERNYM")
                {
                    string satir = textBox1.Text.ToLower() + "|" +
textBox2.Text.ToLower()+ "|(null)|(null)|(null)|";
                    Dosya.Write("\r\n" + satir);
                }
                else if (comboBox1.Text == "SYNONYM")
                {
                    string satir = textBox1.Text.ToLower()+ "|(null)|" +
textBox2.Text.ToLower() + "|(null)|(null)|";
                    Dosya.Write("\r\n" + satir);
                }
                else if (comboBox1.Text == "ANTONYM")
                {
                    string satir = textBox1.Text.ToLower() + "|(null)|(null)|"
+ textBox2.Text.ToLower() + "|(null)|";
                    Dosya.Write("\r\n" + satir);
                }
                else
                {
                    string satir = textBox1.Text.ToLower() +
"|(null)|(null)|(null)|" + textBox2.Text.ToLower() + "|";
                    Dosya.Write("\r\n" + satir);
                }
                Dosya.Close();
            }
            else
            {
                MessageBox.Show("Yoo dostum boş geçemezsin!!");
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Bul();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Ekle();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            textBox1.Clear();
            textBox2.Clear();
```

```
            textBox3.Clear();
            listBox2.Items.Clear();
        }

        private void listBox1_DoubleClick(object sender, EventArgs e)
        {
            Bul();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            Form2 frm = new Form2();
            frm.ShowDialog();
        }
    }
}

//Yeşim AKTAŞ ©


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Text;

namespace deneme1
{
    //class

    public partial class conclusion : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            ngr();

        }

        //nGram hesaplama fonksiyonu
        public void ngr()//yazı, kaç gram olacağı
        {
            string[] ngrsonuc = new string[5000000];
            var liste = new List<string>();


            int sonucsayac = 0;
            //listbox satır sayısı kadar döngü
            for (int i = 0; i < ListBox1.Items.Count; i++)
            {
                //her satır boşluklara göre diziye aktarıldı
                string[] dizi = ListBox1.Items[i].ToString().Split(' ');
                //kaç kelimelikse o kadar döngü kuruldu
                for(int y = 0; y < dizi.Length; y++)
                {
```

```
                    //her bir kelimenin uzunluğu kadar döngü oluşturuldu
                    for (int x = 0; x < dizi[y].Length; x++)
                    {
                        //kelimenin bitimine en az 2 haf kala
                        if (x != dizi[y].Length - 1)
                        {
                            //ikili harf kümelerini listbox'a ekle
                            ListBox2.Items.Add(dizi[y].Substring(x, 2));
                            //tekrar eden 2li gramları bulmak için diziye ata
                            ngrsonuc[sonucsayac] = dizi[y].Substring(x, 2);
                            //dizinin eleman sayısını arttır
                            sonucsayac++;
                        }


                    }
                }
            }

            ListBox2.Items.Add("_____________________");




            int sayacsayi = 0;
            //oluşan 2li gramların adedi kadar döngü oluştur
            for (int i = 0; i < ngrsonuc.Length; i++)
            {
                if (ngrsonuc[i] != null)
                {
                    //birbiriyle karşılaştırma yapmak için aynı döngüden
oluştur.
                    for (int y = 0; y < ngrsonuc.Length; i++)
                    {
                        //bakılan dizi elemanları boş değilde
                        if (ngrsonuc[y] != null && ngrsonuc[y] == ngrsonuc[i])
                        {
                            //adedi arttır
                            sayacsayi++;
                        }
                        else // elemanlardan biri boşsa, iç döngüyü bitir
                            break;
                    }
                    //listbox2'ye karşılaştırılan ilk dizi elemanını ve tekrar
adedini yaz.
                    ListBox2.Items.Add(ngrsonuc[i] + "-->" + sayacsayi);
                    //sayacı tekrar 1'e eşitle.
                    sayacsayi = 0;

                }
                else
                    break;
            }
        }
```

```
        protected void Button4_Click(object sender, EventArgs e)
        {
            if (FileUpload1.HasFile)
            {
                StreamReader oku =
File.OpenText(Server.MapPath(FileUpload1.FileName));
                    string veri = oku.ReadLine();
                    while (veri != null)
                    {
                    ListBox1.Items.Add(veri);
                        veri = oku.ReadLine();
                    }
                    oku.Close();
            }
        }
    }
}
```

EK H. Yazılım Üzerinde Denenmiş Metinler

Bu metinler MEGEP ve değişik kaynaklar üzerinden, ağ temelleri hakkında yazılmış paragraflardan oluşmaktadır. Paragraflar üzerinde değişiklik yapılmadan programa sokulmuştur.

Bu paragraflar:

Seri veri iletiminde, bir kerede bir karakterin sadece bir biti iletilir. Alıcı makine, doğru haberleşme için karakter uzunluğunu, başla-bitir (start-stop) bitlerini ve iletim hızını bilmek zorundadır. Paralel veri iletiminde, bir karakterin tüm bitleri aynı anda iletildiği için başla-bitir bitlerine ihtiyaç yoktur. Dolayısıyla doğruluğu daha yüksektir. Paralel veri iletimi, bilginin tüm bitlerinin aynı anda iletimi sebebiyle çok hızlıdır.

Herhangi bir zamanda veri gönderilebilir. Veri gönderilmediği zaman hat boşta kalır. Senkron seri iletişimden daha yavaştır. Her veri grubu ayrı olarak gönderilir. Gönderilen veri bir anda bir karakter olacak şekilde hatta bırakılır. Karakterin başına başlangıç ve sonunda hata sezmek için başka bir bit eklenir. Başlangıç için başla biti (0), veri iletişimini sonlandırmak için ise dur biti (1) kullanılır.

Senkron iletişim alıcı ve vericinin eş zamanlı çalışması anlamına gelir. Önce gönderici taraf belirli bir karakter gönderir. Bu her iki tarafça bilinen iletişime başlama karakteridir. Alıcı taraf bu karakteri okursa iletişim kurulur. Verici bilgileri gönderir. Transfer işlemi veri bloku tamamlanana ya da alıcı verici arasındaki eşleme kaybolunca kadar devam eder

---

Topoloji bilgisayarların birbirine nasıl bağlandıklarını tanımlayan genel bir terimdir. Topolojinin bir kısmı kablolama arabirimlerinden bahseden fiziksel

topoloji kısmıdır. Diğer kısmı ise medyanın veri gönderiminde nasıl kullanıldığından bahseden mantıksal topoloji kısmıdır.

Bütün terminaller tek bir doğrusal kablo ile birbirlerine bağlanmışlardır. Burada hatta gönderilen sinyal tüm terminallere gider. Sinyal bir hedefe ulaşana ya da bir sonlandırıcıya gelene kadar hatta dolaşır. Hattaki bilgi akışı çift yönlüdür. Kaynak istasyon bilgiyi hatta bırakır. Bilgi her iki yönde ilerleyerek hatta yayılır. Ancak bu topolojide birden fazla istasyonun bilgi göndermesi durumunda ağ trafiğinde aksamalar meydana gelir. Bunu önlemek için hat paylaşımını düzenleyen ağ protokolleri kullanılmalıdır.

Bus topolojisi kullanılarak kurulan ağlarda koaksiyel kablo kullanılır, ağdaki her istasyona ise T-konnektör takılır. Bus topolojisinde verileri sonlandırmak için mutlaka kablonun iki ucuna sonlandırıcı (terminatör) adı verilen ağı sonlandıran parçalar takılmalıdır.

En yaygın kullanılan topoloji tipidir. Bu topolojisinde her bilgisayar ağ iletişiminin gerçekleşmesi için merkezi birim (switch, hub, vs) dediğimiz cihazlara bağlanır. Hatta gönderilen sinyal önce merkezi birime ulaşır, buradan hedefe yönlendirilir.

Genellikle yıldız topolojisindeki ağları birbirine bağlamak için kullanılır. Böylece ağlar büyütülebilir. Bir ağacın dalları farklı topolojilerdeki ağları temsil eder, ağacın gövdesi ile de bunlar birbirine bağlanabilir

---

Mantıksal olarak bir daire şeklinde tüm düğümlerin birbirine bağlandığı topoloji çeşididir. Hatta gönderilen sinyal hedefe ulaşıncaya kadar tüm terminallere

uğrar. Düğümlerden herhangi birindeki hatanın ya da kablodaki bir sorunun tüm sistemi etkilemesi bu topolojinin en önemli dezavantajıdır.

Kablolu bağlantı, ağdaki cihazların birbirlerine kablo vasıtası ile bağlandıkları yapıdır. Kablolu bağlantıda kablo uzunluğunun artması iletişim performansını olumsuz etkilemektedir. Bu bağlantı türünde kullanılan kablo türüne göre bağlantı hızında değişiklikler olabilmektedir.

İnternete bağlı her sistemin kendisine ait özel bir adresi vardır. Bunlar IP (internet protokol) adresi olarak adlandırılır ve bilgisayarlar arasında iletişim yapılırken veri paketlerinin adreslenmesinde kullanılır. Tipik bir IP adresi noktalarla ayrılan dört rakamdan oluşur; örneğin, 192.168.2.1. Bu adres 32 bitlik bir sayıdır dolaysıyla ağ üzerine 2 32 tane, yani yaklaşık 4 milyar tane bilgisayar bağlanabilir

---

İnternete bağlanmak amacıyla kullanılan modemler IP numaralarını otomatik olarak kendi havuzlarından dağıtır. Bilgisayarınız bu havuzdan IP numarası alarak internet ağına dâhil olur. Ancak bazı durumlarda bilgisayarınız otomatik IP alamayabilir, böyle durumlarda bilgisayarınızın IP alma işlemini el ile yapılandırmanız gerekmektedir.

DHCP (dynamic host configuration protocol / dinamik istemci ayarlama protokolü), bir TCP/IP ağındaki makinelere IP adresi, ağ geçidi veya DNS sunucusu gibi ayarların otomatik olarak yapılması için kullanılır. Günümüzde neredeyse tüm ev ve halka açık ağlarda kullanılmaktadır, ofis veya daha kontrollü bir bağlantı sağlanan yerlerde ise statik IP adresi tercih edilir.

IP adresleri IPv4 (32bit) ve IPv6 (128bit) olmak üzere iki çeşittir. Günümüzde yaygın olarak IPv4 (32bit) adresleme mekanizması kullanılmaktadır. Güvenilir ve sınırlı sayıda kullanıcıya hizmet etmek için dizayn edilen IPv4'ün, geniş kitleler ve çok fazla uygulama tarafından kullanılır hâle gelmesiyle iletişim güvenliği konusunda ciddi zaaflar ortaya çıkmıştır. Bu zaafların üstesinden gelmek için IETF (internet engineering task force / internet mühendisliği görev gücü) tarafından yürütülen çalışmalar sonucunda 128 bitlik yapısı ile yeni nesil IP protokolü (IPv6) ortaya çıkmıştır.

IP sayısındaki artışın yanında sade başlık yapısı, geliştirilmiş seçenekler bölümü ve içerdiği güvenlik uygulaması ile IPv6 birçok yenilik getirmektedir. Beklenildiği hızla yaygınlaşmasa da özelikle IPv4 sayısında sıkıntı çeken ülkelerde yoğun olarak kullanımına başlanılan IPv6'yı birçok cihaz üreticisi de desteklemeye başlamıştır. Uygulama bazında sıkıntılar yaşanılsa da IPv6'ya tüm internette kademe kademe geçileceği tahmin edilmektedir.

IPv4 32 bit adresleri kullanır. IPv4 adresleme kullanılarak 4 milyar 294 milyon 967 bin 196 tane bilgisayar adreslenebilir (232). Adresler birbirinden nokta ile ayrılmış dört adet sekiz bitlik parçalardan oluşur. Bu sayılar 0 ile 255 arasında bir değer olabilir. Örnek bir IPv4 adresi: 192.168.2.1’dir.

---

32 bitlik bir adres yapısına sahip olan IPv4 adreslemede ciddi sıkıntılar meydana getirmektedir. IPv4 oluşturulmaya başlandığında internetin bu kadar ilerleyeceği hesap edilmemişti. Şimdi adresleme sıkıntısı oluşunca 128 bitlik adres yapısı olan IPv6 'ya geçilmesi kaçınılmaz olmuştur.

A sınıfı adresler: IBMNET, MILNET gibi büyük ağlar bu ağ sınıfını kullanır. İlk oktet 0 ile 127 arasındadır. İlk oktet ağ numarasını belirtir (toplam 126 tane) .Geri kalan 3 oktet ise bilgisayar numarasıdır. Örnek olarak 122.113.45.67 IP sini ele alacak olursak bu IP, 122 numaralı A sınıfı ağ içerisindeki 113.45.67 nu.lı bilgisayarı belirtir. 127.0.0.1 IP' si A sınıfı IP olmasına karşın yerel localhost IP' si olarak kullanıldığı için ağ adreslemede kullanılmaz.

TCP/IP'de iki cihaz aynı ağda olup olmadıklarını birbirlerinin IP adreslerinin ilk birkaç basamağına bakarak anlarlar. Bu basamağa ağ maskesi (subnet mask) denir. Bir bilgisayar ancak aynı ağda bulunan bir bilgisayarla doğrudan iletişime geçebilir. Aynı ağda değillerse dolaylı olarak iletişime geçer. Aynı ağda olup olmadığını IP adreslerini kullanarak anlarsınız. IP adresinin bir bölümü ağı, diğer bölümü de bilgisayarın ağ içindeki adresini tanımlar. Hangi bölümü ile ağı hangi bölümü ile bilgisayarı tanımladığını bilmek için alt ağ maskesi kullanılır.

Bütün terminaller tek bir doğrusal kablo ile birbirlerine bağlanmışlardır. Burada hatta gönderilen sinyal tüm terminallere gider. Sinyal bir hedefe ulaşana ya da bir sonlandırıcıya gelene kadar hatta dolaşır. Hattaki bilgi akışı çift yönlüdür. Kaynak istasyon bilgiyi hatta bırakır. Bilgi her iki yönde ilerleyerek hatta yayılır. Ancak bu topolojide birden fazla istasyonun bilgi göndermesi durumunda ağ trafiğinde aksamalar meydana gelir. Bunu önlemek için hat paylaşımını düzenleyen ağ protokolleri kullanılmalıdır.

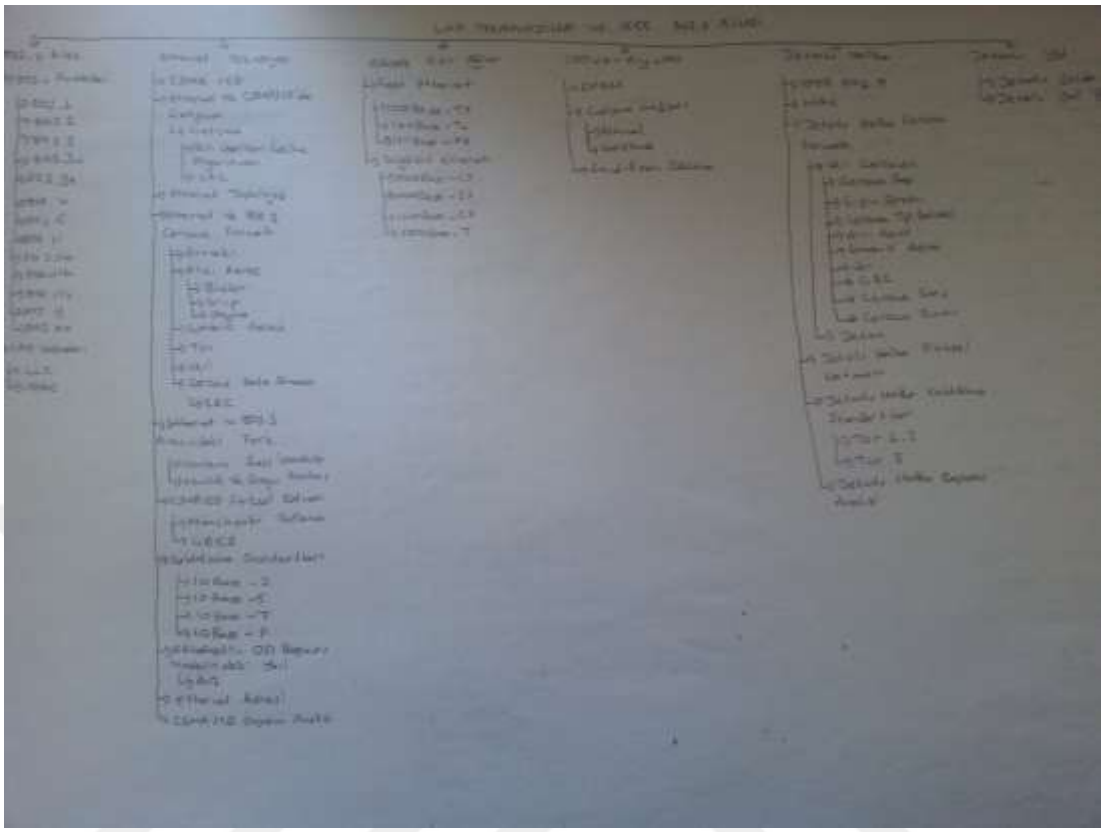
Bus topolojisi kullanılarak kurulan ağlarda koaksiyel kablo kullanılır, ağdaki her istasyona ise T-konnektör takılır. Bus topolojisinde verileri sonlandırmak için mutlaka kablonun iki ucuna sonlandırıcı (terminatör) adı verilen ağı sonlandıran parçalar takılmalıdır.

Yönlendirici (router) temel olarak yönlendirme görevi yapar. LAN-LAN ya da LANWAN arasında bağlantı kurmak amacıyla kullanılır. Üzerinde LAN ve WAN bağlantıları için ayrı port bulunur ve şaseli olarak da üretilebilir. Bu portlara gerektiğinde LAN veya WAN portları eklenebilir.

Günümüzde kullanılan kablosuz modemlerden birisi de 3G mobil modemlerdir. 3G mobil internet ile GPRS/EDGE destekli 3G uyumlu 3G mobil modeminizle kablosuz, kolay ve hızlı bir şekilde her yerden internete bağlanabilirsiniz. 3G mobil modem ile epostalarınıza hareket hâlindeyken ulaşabilir, SMS gönderebilir, telefonunuzu meşgul etmeden kablosuz, kolay ve hızlı bir şekilde her yerden internete bağlanabilirsiniz.

Extranet (dış ağ) ise buna benzer olmakla birlikte müşteriler, iş ortakları veya şirket dışından herhangi birileri tarafından erişilebilir olma özelliğine sahiptir. Extranet tasarım, şirketinizin dışındaki insanlarla irtibat hâlinde olabilmenizi sağlar ve şirketinizin, belge yönetimi, dosya ve fotoğraf değişimi, posta, duyurular, bilgi edinme talebi, müşteri proje güncelleme, etkinlik takvimi, çevrimiçi katalog, fiyatlandırma, irtibat yönetimi, müşteri geribildirimi gibi birçok işinin gerçekleştirilmesine imkân tanır.

EK I. Fotoğraflar



EK B.1. Kitaptan Oluşturulan Sözlük Kavramları Sayfa Örneği

ÖZGEÇMİŞ

Adı Soyadı : Yeşim AKTAŞ

Doğum Yeri ve Yılı : Kumluca, 1992

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : ysimaktas@gmail.com

Eğitim Durumu

Lise : Kumluca Anadolu Meslek Lisesi, 2009

Lisans : SDÜ, Teknik Eğitim Fakültesi, Bilgisayar ve Elektronik Eğitimi Bölümü Bilgisayar Sistemleri Öğretmenliği, 2013

Erasmus - Master : Fachhochschule Schmalkalden, Informatik, 2014-2015.

Mesleki Deneyim

SDÜ Yalvaç TBMYO: 2016- …….. (Devam Ediyor)