METİN DOSYALARI İÇİN ARAYICI

Orhan DEMİRSÖZ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

TEMMUZ 2009
ANKARA

Orhan DEMİRSÖZ tarafından hazırlanan METİN DOSYALARI İÇİN ARAYICI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Doğan ÇALIKOĞLU

Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Eğitimi Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr. Doğan ÇALIKOĞLU

Üye : Yrd. Doç. Dr. H. Şakir BİLGE

Üye : Yrd. Doç. Dr. Remzi YILDIRIM

Tarih : 09/07/2009

Bu tez, Gazi Üniversitesi Bilişim Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Orhan DEMİRSÖZ

METİN DOSYALARI İÇİN ARAYICI

(Yüksek Lisans Tezi)

Orhan DEMİRSÖZ

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Temmuz 2009

ÖZET

Bu çalışmada, kısaca BELAR denen yeni bir belge içi arama yazılımı tasarımlanmış ve gerçekleştirilmiştir. BELAR'ın özelliklerinin belirlenmesinde diğer metin düzenleyicilerinin arama özelliklerinde hissedilen noksanlıklar dikkate alınmıştır. Word belgeleri üzerinde işlem yapmak için geliştirilen BELAR, C# dilinde yazılmış olup, Microsoft Word programını bir alt program olarak kullanmaktadır; ancak BELAR'da ki arama imkanları Microsoft Word'deki imkanlara göre çok daha gelişkindir. BELAR'da aranan metin parçaları için üç farklı ölçü kullanılmaktadır. Bunlar cümle, paragraf ve bağlamdır. Burada kullanılan bağlam ölçüsü, cümle sayısı cinsindendir ve bu sayı kullanıcı tarafından belirlenebilmektedir. Belgenin tamamında, tercih edilen ölçüye göre düzenli ifade anahtarları ile arama yapılmakta ve bulunan sonuçlar listelenmektedir. Daha sonra bu liste içerisinden seçilen sonucun belge içindeki konumuna ulaşılabilmektedir. Böylece aramayı yapan kişi belge üzerinde dilediği işlemleri tamamlayabilir.

Bilim Kodu : **702.3.006**
Anahtar Kelime : **Gelişmiş arama, Metin arama, Düzenli ifadeler**
Sayfa Adedi : **75**
Tez Yöneticisi : **Prof. Dr. Doğan ÇALIKOĞLU**

SEARCHING FOR TEXT FILES

(M.Sc. Thesis)

Orhan DEMİRSÖZ

GAZI UNIVERSITY

INFORMATICS INSTITUTE

July 2009

ABSTRACT

In this study, a new software for performing search in documents, called BELAR, is designed and implemented. In determining the properties of BELAR, the short comings observed in the other text editors' searching capabilities were taken into consideration. BELAR, which is developed for processing Word documents, is written in the C# language and uses Microsoft Word program as a sub program; however, searching capabilities of BELAR are more comprehensive than that of Microsoft Word. Three different measures are used for searching text fragments in BELAR. These are sentence, paragraph and context. Here, the context measure is in terms of number of sentences and this number can be specified by the user. Searching is performed in the whole document using regular expression keys according to the searching measure and the results found are listed. It is possible later to reach the location of the results in the document, selected from the list. This way, the person performing the search can complete the operations he desires on the document.

Science Code : **702.3.006**
Key Words : **Advanced search, Text search, Regular expression**
Number of pages : **75**
Adviser : **Prof. Dr. Doğan ÇALIKOĞLU**

TEŞEKKÜR

Ülkemizde araştırmalara bir katkı olması amaçlanan bu arama programını kullanmak isteyenlerin danışmanımdan izin almaları yeterlidir. Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren danışmanım Prof. Dr. Doğan ÇALIKOĞLU'na, yine kıymetli tecrübelerinden faydalandığım hocam Yrd. Doç. Dr. Hasan Şakir BİLGE'ye, ayrıca çok değerli arkadaşım Gürhan USTALİ' ye ve bana yardımcı olan Elif ERDOĞAN ve tüm çalışma arkadaşlarıma, çalıştığım kurum yöneticilerine, manevi destekleriyle beni hiçbir zaman yalnız bırakmayan başta eşim olmak üzere, çocuklarıma ve ağabeyime teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELER

Çizelge **Sayfa**

RESİMLER

Resim **Sayfa**

1. GİRİŞ

Bu çalışmada .doc ve .docx uzantılı metin dosyaları üzerinde gelişkin arama yapabilen bir yazılım geliştirilmiştir. Günümüzde kullanılmakta olan metin işleyicilerinde bazı arayış seçenekleri varsa da bunlarda yetersizlikler bulunmaktadır. Bu çalışmada temel amaç, belge içi arayışta bu yetersizliklerin mümkün olduğu kadar giderildiği bir uygulama geliştirmek olmuştur.

Bilgisayar ortamında .doc ve .docx uzantısı ile saklanan Word belgeleri içerisinde kolay ve gelişmiş arama hedeflenmiştir. Programın kullanıcı arayüzünün anlaşılabilir olması ve kolay kullanılabilmesi için tercih kutuları yapılmıştır. Bu kutularla metin içerisinde aranan kelimelerin, "tam kelime", "ile başlar", "ile başlamaz", "ile biter", "ile bitmez", "içerir", "içermez" tercih kutularından birini işaretleyebilmesi sağlanmıştır.

Belge içindeki arayış işlemleri, aralarında mantıksal bağlaçlar (ve,veya) olan birden fazla tercih kullanılarak yapılabilmektedir. "veya" bağlacı ile tercihlerin herhangi birinin karşılandığı sonuçlara, "ve" bağlacı ile de tercihlerin her birinin aynı anda bulunduğu sonuçlara ulaşmak mümkün olmaktadır. Böylece belge içinde bulmayı istediğimiz doğru sonuçlara daha hızlı ulaşılmaktadır.

Metin arama işlemlerinde önem arz eden düzenli ifade (regular expression) anahtarları kullanılarak daha ayrıntılı arama yapılması sağlanmıştır. Düzenli ifade anahtarları ile belgede arzu edilen ayrıntılı içeriklere ulaşmak mümkün olacaktır.

Arama seçeneklerinde cümle, paragraf ve bağlam olmak üzere üç farklı ölçü kullanılmakta ve arama işlemlerinin hangisine göre yapılacağı kullanıcıya bırakılmaktadır. Tüm belge içinde arama tercihlerine uygun bulunan sonuçlar listelenmekte ve liste içerisinden tercih edilen sonucun gerçek belgede bulunduğu yere konumlanılabilmektedir. Herhangi bir veri tabanına ihtiyaç duymadan ve Türkçe metinlerde sorunsuz çalışması sağlanmıştır.

İnsanlar dünden bugüne öğrenme ihtiyacını gidermek için araştırmalar yapmaktadırlar. Bu araştırmalar ilk zamanlar direk bilginin kaynağı olan insanlara ulaşarak yapılırken, zamanla yazılı eserler üzerinden yapılmaya başlanmıştır. Yazılan eserlerden araştırmacıların faydalanabilmesi için kütüphaneler yapılmıştır. Belli bir dönem bu ihtiyacı karşılayan kütüphaneler, teknolojinin gelişmesi ile birlikte bilgisayar ortamlarına taşınmışlardır. Çünkü gerek bilginin üretimi, gerekse saklanması için bilgisayarlar hayatın vazgeçilmez bir parçası haline gelmiştir. İnternetin yaygınlaşması ile birlikte dünyanın farklı ülkelerinde yayınlanmakta olan bilgilere de ulaşılabilmektedir. Ancak zaman içerisinde elektronik ortamda artan kaynakların yoğunluğu, yapılan araştırmalarda istenen sonuçlara hızlı ulaşabilme ihtiyacını ortaya çıkarmıştır. Bu konuda farklı çalışmalar yapılmış, bilgiye hızlı erişim için yöntemler geliştirilmiştir.

Bilgisayar ortamında saklanan bu kayıtlara herhangi bir anda ulaşılma isteği veri madenciliğini ortaya çıkarmıştır. Veri madenciliğinde, veri herhangi bir veritabanında sorgulanabileceği gibi herhangi bir formata sahip metin içerikli dosyada da sorgulanabilir. Veriler herhangi bir veritabanından ya da metin dosyasından aranması işleminde arama algoritmalarına ihtiyaç duyulur. Arama algoritması bilgisayar biliminde en temel algoritmalardan biridir. Arama algoritması, bir probleme, bir çözüm uzayı içerisinden en uygun çözümü arayan algoritmadır. Arama işlemine sürekli ihtiyaç duyulduğu için aramanın hızlı olması da oldukça önemlidir. Bu sebepten dolayı arama algoritmaları elde olan bilgiyi en iyi şekilde kullanarak sonuca nasıl daha hızlı ulaşılacağı üzerinde yoğunlaşır.

Arama algoritmaları, algoritmanın yapısına uygun olarak seçilen bir başlangıç çözümünden ya da çözümlerinden başlar ve farklı yapılar kullanarak yeni çözümlere ulaşırlar. Bu çözümler amaç fonksiyonu (benzetim modeli) ile değerlendirilir. Bu akış belirli bir durdurma koşulu sağlanıncaya kadar devam eder ve bu koşula ulaşılınca en iyi / iyi çözüm bulunmuş olur [1].

Arama, en uygun cevabı vermek için bilgi alternatiflerini araştırmaktır. Arama bilginin kalitesini artırır. İstenmeyen bilgileri atarak kaynakları sınırlı kullanarak, gereksiz bilgi miktarını azaltır [2].

Elektronik ortamdaki bilgilere erişim konusunda iki husus vardır. Bunlardan birincisi internet üzerinden arama yöntemleri kullanarak web sayfaları üzerinden bilgiye erişim. İkincisi bilgisayarlarımızın disklerinde bulunan elektronik kitap, dergi, makale, tez v.b kaynaklar içerisinde bilgiye erişim. Bilimsel araştırmalarda en az birincisi kadar ikincisi de yaygın olarak kullanılmaktadır. Web sayfaları üzerinde gelişmiş arama seçenekleri sayesinde ayrıntılı aramalar yapılabilmekte, ancak bilgisayarımızda bulunan belgeler içerisinde aynı kolaylıkta aramalar yapılamamaktadır. Kullandığımız işletim sistemleri arama seçenekleri her geçen gün geliştirilmekte, ancak bunlar belge içi aramada istenen seviyede başarılı olamamaktadır. Metin belgelerimizi okumak ve yazmak için kullandığımız Wordpad, Microsoft Word v.b metin işleyicilerin arama seçenekleri ise esnek bir kullanıma sahip değildir. Google v.b arama motorlarında yapılabilen gelişmiş arama Microsoft Word' de yapılamamaktadır.

Bu kapsamda belge içi arama özelliklerine sahip benzer yazılımlar incelenmiş olup, tasarım aşamasında bizim projemize ışık tutacak taraflarından faydalanılmış, eksik yönleri görülerek farklı tarzda bir uygulama geliştirilmiştir. İncelenen benzer programların özelliklerine kısaca değinilmiştir.

Güncel Metin Editörlerin Arama Seçenekleri

Microsoft Word programı bul seçeneği;

Aranan kutusuna girilen ifadeyi metin içerisinde ararken büyük/küçük harf eşleştir, yalnızca tam sözcükleri bul ve Joker karakter kullan gibi seçenekler vermektedir. Burada kurulan yapı tek bir kelimeyi, ya da girilen tam ifadeyi bulmakta, ancak birden fazla sözcükle aramada başarısız olmaktadır. Kullanıcıların detaylı gelişmiş arama yapabilmesi için düşünülen joker karakterlerin kullanımı ise yine tek kelime üzerinde başarılı olabilmektedir. Ayrıca bu joker karakterlerin anlaşılması ve kullanılması kullanıcı açısından zor olduğundan, çok bilinmemekte ve tercih edilmemektedir.

Adobe Acrobat Reader programı search seçeneği;

Bilindiği üzere bu program Adobe firması tarafından üretilen ve kendine özgü dosya formatı olan pdf uzantılı belgeleri açmakta kullanılmaktadır. Bu editörün "use advanced search options" ile sunmuş olduğu seçeneklere baktığımız da yukarda bahsedilenlerden çok farklı olmadığı görülmektedir. Yine aramalarda kelime yada tam ifade ile arama yapılmakta, birden fazla kelimelere göre gelişmiş arama yaptıramamaktadır. Bu editörde bizimde çalışmamıza ışık tutan faklı bir yönü mevcuttur. Aranan ifadeye göre tüm belge içinde bulunan bölümler sayfa bazlı indekslenerek listelenmektedir. Kullanıcı belge içinde aradığı kelimenin geçtiği bütün cümleleri sıralı bir şekilde toplu halde görebilmekte, linke tıklayarak doküman içerisinde ilgili yere hızlıca ulaşabilmektedir. Buda kullanıcı için ekstra bir kolaylık sağlamaktadır.

Çalışmamıza Işık Tutan Yazılımlar

Yapmış olduğum araştırmalar neticesinde bulunan yazılımların Türkçe olmadığı, her birinin arama işlemini kendine özgü farklı tarzda yaptığı görülmüştür. Bu yazılımların isimleri kısaca özellikleri şöyledir.

File Search Assistant: İnternet üzerinden 30 günlük deneme sürümünü indirdiğim bu program beğendiğim programlardan biridir. Bu program disk üzerinde belirtilen klasörlerde hem dosya adında hem de dosya içinde arama yapabilmektedir. Birçok dosya türünü (txt,doc,xls,pdf) desteklemektedir. Arama sonucunda bulunanlar ekranın bir bölümünde listelenmekte, aynı zamanda listelenenler üzerinde hareket edilerek bir alt ekranda doküman içindeki yeri alınarak bir text editörde gösterilmektedir. Arama sırasında "use regular expression" seçeneği ile düzenli ifadeleri kullanarak arama yapabilmektedir. Ancak kullanıcıya bu aşamada detay seçenek sunmamaktadır. Ayrıca ilgili belgenin tümü üzerinde çalışma yapılacağı zaman ayrı bir editör açmak gerekmektedir.

Archivarius 3000 : Bilgisayarda, yerel ağda ve çıkartılabilir sürücülerde (CD,DVD ve diğerleri) belge ve e-posta aramak için kullanılabilen bir uygulamadır. Archivarius

3000 herhangi bir dildeki sorguları algılayabilir. Bir çok dil desteği olan Archivarius 3000'de (İngilizce, almanca, İspanyolca, Fransızca) Türkçe desteği de vardır. Türkçe menüleri olan bu program arama da oldukça başarılıdır. Birden fazla kelime ile arama yapılabilmektedir. "Tüm kelimeler ile", "kelimelerden herhangi biri ile", "tam sözcük ile", "şu kelimeler hariç", "kısmi kelimeler ile" gibi gelişmiş arama seçenekleri sunmaktadır. Ekran dikey olarak iki kısma ayrılmakta, birinci kısımda bulunan dosya ismi ve dosya içinden bir bölüm listelenmekte, ikinci kısımda ise yine programın kendine özgü bir metin editörü ile belge açılmaktadır. Aramada başarılı olarak kabul edebileceğimiz bu yazılım, belgeleri düz metin olarak açtığından bir dezavantaj yaşamaktadır. Ayrıca arama işleminden önce ilgili klasörlerin indekslenmesi gerekmektedir.

Super Text Search: Amerika da satışa sunulan bu program düzenli ifadeler kullanarak farklı biçimdeki metin dosyaları üzerinde arama yapabilmektedir. Yukarda anlatılanlara benzer mantıkta çalışmakta, detaylı arama yapabilmek için düzenli ifadeleri bilmek gerekmektedir. Ayrıca Türkçe dil desteği olmadığından, Türkçe belgelerde işe yaramamaktadır.

Windows Desktop Search: Microsoft tarafından ücretsiz olarak hizmete sunulan bu program belge içi arama yapma özelliğine sahip. Belirtilen klasör içerisinde dosya isimleri ve dosya içi arama yaparak bulunanları ekranın solunda listelemektedir. Arama kriterine uygun olarak bulunan sonuçlar üzerine gelindiğinde ise ekranın sağ tarafında belge içinden bulunan bölüm gösterilmektedir. Metin dosyası şeklinde gösterilen bu kısımda bulunan kelimenin nerede geçtiği görülememektedir.

Yukarıda bahsi geçen benzer programların eksik yönleri ve faydalı özellikleri dikkate alınarak BELAR tasarımlanarak gerçekleştirilmiştir. BELAR ile büyük Word dosyaları içerisinde aranılan bilgiye daha kolay ulaşılabileceğinden, vakit kayıpları önlenmiş olacaktır.

2. BELGE İÇİ ARAYICI ÖZELLİKLERİNİN BELİRLENMESİ

Bu bölüm üç kısma ayrılmıştır. Birinci kısımda bu konuda yapılmış olan bilimsel çalışmalar araştırılmıştır. İkinci kısımda ise çalışmamıza ışık tutan diğer yazılımlar detaylı olarak incelenmiştir. Üçüncü kısımda ise Word programı bul seçeneği kullanımı detaylı anlatılmıştır.

2.1. Metinlerde Aramaya Yönelik Yapılan Bilimsel Çalışmalar

Son yıllarda metin tabanlı belge içeriklerinde aramaya yönelik çalışmalar artmıştır. Cambazoğlu (2006), yüksek lisans tezinde, arama motorları teknolojisinde kullanılan paralel metin getirme ile ilgili çalışmıştır. Bir parallel metin getirme sistemi temel olarak üç bileşenden oluşmaktadır: tarayıcı, indeksleyici ve sorgu işleyici. Tarayıcı bileşeni Ağ'da bulunan sayfaları bulmayı, getirmeyi ve yerel bir metin ambarında saklamayı amaçlar. ˙Indeksleme bileşeni saklanmış olan düzensiz metinleri sorgulanabilir bir yapıya dönüştürür ki bu yapı bu coğu zaman bir ters dizindir. Sorgu işleme bileşeni ise indekslenmiş içerik üzerinde aramayı gerçekleştirir. Cambazoğlu tezinde, etkin Ağ tarama ve sorgu işleme için modeler ve algoritmalar önerilmiştir. Paralel Ağ tarama için, işlemciler arası iletişim miktarını en aza indiren ve işlemcilerin sayfa indirme isteklerinin sayısını ve saklama yüklerini dengeleyen karma bir model önerilmiştir. Ek olarak, metin ve kelime bazlı ters dizin bölümleme için modeller önerilmiştir. Metin bölümlemeye dayalı modelinde saklama yükü dengelenirken sorgu işleme sırasında karşılaşılacak disk erişim miktarı en aza indirilmektedir. Kelime bölümlemeye dayalı modelinde ise yine saklama yükü dengelenirken toplam iletişim hacmi en aza indirilmektedir. Bunlara ek olarak, Sıralamaya dayalı metin getirme sistemleri için çok sayıda sorgu işleme algoritması kullanmıştır [3].

Güldal (2005), yüksek lisans tezi çalışmasında, sunucudaki posta kutusunda saklanan elektronik postalar üzerinde, belirlenen sorgu ifadesine göre sınıflandırma yapmayı sağlayan bir yazılım gerçekleştirmiştir. Böylece, ilgi sahasına, amaca ve çalışma alanına uyan veya uymayan e-postaları ayırmanın mümkün olması sağlanmıştır. Yüksek lisans tezi çalışmasında, uzak posta sunucusuna bağlanıp POP3 protokolünü

kullanarak, bir kullanıcının hesabına gelen postalar üzerinde, belirlediği bir sorgu ifadesini uygulayıp, sorguya uyanlar ve uymayanlar olarak ayıran ve bunu kullanıcıya bildiren istemci bir uygulamanın geliştirilmesini amaçlamıştır. Çalışma, temel olarak iki ana parçadan oluşmaktadır. Birincisi, herhangi bir POP3 sunucusuyla iletişim kuran bir istemci ağ uygulamasıdır. Bu amaçla, Internet protokollerinin ve standartlarının yapısı, çalışma biçimleri, geliştirilme süreçleri incelenmiş, POP3 protokolü üzerinde detaylı bir şekilde durulmuştur. İkincisi ise, anahtar kelime tabanlı mantıksal sorgulara dayalı bir arama mekanizması kullanılmış olup, küçük bir "Bilgi Erişim" uygulamasına giriş olarak da değerlendirilebilir. Bu amaçla, bilgisayar bilimlerinin bu alanı üzerinde de çalışma yapılıp uygulaması gerçekleştirilmiştir [4].

Güven (2007), doktora tezinde "Belge bazlı veri yığınları içinden doğru belgelerin bulunması" üzerinde çalışma yapmıştır. Bilgisayar sistemlerinin ilk uygulama alanları veri toplama ve raporlama üzerinedir. Veri saklama kapasitelerinin ve bu verileri işleyecek bilgisayar işlemci gücünün artması ile daha fazla veriyi saklama ve inceleme imkanı doğmuştur.Böylece daha önce verilerden elde edilemeyen ilişkilerin, desenlerin ortaya çıkarılması mümkün hale gelmiştir. Geleneksel sorgulama yöntemlerinden farklı olan bu yöntemler veri madenciliği adı altında toplanmıştır.Veri madenciliği, verilerin içerisindeki desenlerin, ilişkilerin, değişimlerin, düzensizliklerin, kuralların ve istatistiksel olarak önemli olan yapıların yarı otomatik olarak keşfedilmesidir. Belge bazlı veri yığınları içinden doğru belgelerin bulunması, belgelerin birbirleri arasındaki ilişkilerin sorgulaması işlemleri için veri madenciliği alanındaki teknikler birebir uygulanabilir değildir. Bu nedenle belge madenciliği yapmak için farklı yöntemler geliştirilmiş ve bu alan metin madenciliği, belge madenciliği, yarı yapısal veri madenciliği gibi isimler altında toplanmıştır. Belge madenciliği çalışmalarında amaç belge içeriğinin, bir insan tarafından okunmuşçasına bilgisayar ortamında belirlenmesini içerir. Bu durumda belgelerin hangi dilde yazıldığı önem kazanmaktadır. Bu yönü itibariyle doğal dil işleme alanı ile belge madenciliği arasında sıkı bir ilişki doğmuştur. Hem belge madenciliği hem de doğal dil işleme çalışmaları uzun yıllardan beri İngilizce başta olmak üzere farklı diller üzerinde yapılmıştır. Güven (2007), tez çalışması Türkçe

belgeler üzerinde belge madenciliği yapmak amacıyla, Gizli Anlambilimsel Dizinleme (GAD) yöntemini kullanmakta ve kelimelere uygulanan n-gram yaklaşımını bu yöntemle birleştirmektedir. Önerdiği yöntem, Türkçe belgelerin madenciliği için kullanılmıştır. Bu amaçla bir Türkçe belge kümesi oluşturulmuştur. Ancak bu belge kümesi, uluslararası standart belge kümeleri gibi Türkçe için kabul edilmiş bir standart değildir. Bu nedenle elde edilen neticelerin değerlendirilmesinde, belge kümesinin yanlılığı gibi bir sebebe dayalı subjektiflikler olduğu iddia edilebilir [5].

Saraçoğlu (2007), metinsel belgelerin kullanışlı bir şekilde organize edilmesi, işlenmesi ve faydalı bilgiler çıkarılması gibi amaçları yerine getirilmesi için metin sınıflandırıcısı ve metinsel belge arama mekanizmaları üzerinde çalışmıştır. Bulanık kümeleme kullanmıştır. Günümüzde teknolojinin gelismesi ile birlikte her geçen gün büyük miktarlarda veriler ortaya çıkmaya ve depolanmaya baslanmıstır. Bu verilerdenfaydalanmanın yolu ise onların verimli bir sekilde organize edilmesi ve yararlıbilgilere dönüstürülmesinden geçmektedir. Bunu amaçlayan veri madenciliginin birçesidi ise metinsel veriler üzerinde çalısan metin madenciligidir. Metinsel belgelerin kullanıslı bir sekilde organize edilmesi, islenmesi ve faydalı bilgiler çıkarılması gibi amaçları yerine getirmek için gerekenlerin basında metin sınıflandırıcısı, metinsel belge arama mekanizmaları vb. araçlar gelmektedir.Bir metinsel belge arama islemini iki farklı yaklasımla ele almak mümkündür. Bunlardan biri genis bir alandaki belgeler üzerinde anahtar kelime seçilmesine dayalıolarak arama yapmaktır (internet arama motorları gibi). Bir digeri ise daha dar bir alanda metinin tüm kelimelerini kullanmak suretiyle daha ayrıntılı bir aramayapmaktır (bir kütüphanedeki kitaplar üzerinde yapılacak arama gibi). Şaraçoğlu (2007), bulanık kümeleme ve metinlerin tüm kelimelerini kullanarak birarama yaklasımı ortaya koymuştur. Bu yaklasım; önisleme, kümeleme/sınıflandırmave benzerlik ölçümü olmak üzere üç temel asamadan olusmaktadır. Önisleme asaması ile ilgili olarak terim agırlıklandırma yöntemleri üzerinde durulmustur. Bulanık kümeleme kullanıldıgından dolayı mevcut terim agırlıklandırma yöntemlerinin bulanık kümeleme ile birlikte kullanımları incelenmis ve performansları karşılaştırılmıştır. En iyi performansı gösteren yöntembelirlenerek daha sonraki asamalarda bu yöntem

kullanılmıstır. Benzerlik ölçümü asaması için ise mevcut benzerlik ölçümlerinin önerilen arama yaklasımındaki performansları incelenmistir. Yine bu asama için verinin boyutuna dayalı yeni bir benzerlik ölçümü önerilmistir. Bu önerilen yeni benzerlik ölçümünün süre ve verimlilik açılarından önceki yöntemlere göre daha iyi oldugu görülmüstür. Son olarak, bir test belgesinin birden fazla kategoriye ait olması seklinde özetlenebilecek olan çoklu kategori problemi ele alınmıstır. Bu problemin çözümü için önerilen arama yaklasımının kümeleme/sınıflandırma asaması gelistirilmeye çalısılmıstır. Bu amaçla hangi belgelerin birden fazla kategoriye ait olduklarını tespit etmek için mevcut sınıflandırma yöntemi probleme adapte edilmistir. Ayrıca, kategorilerin arasında bir iliski matrisi olusturularak, bir belge birden fazla kategoriye ait ise bunların hangi kategoriler oldukları tespit edilmeye çalısılmıstır. Önceki çalısmalarda pek yer verilmemis olan bu çoklu kategori probleminde önemli ölçüde bir başarı sağlanmıstır [6].

Dursun (2008), Türkçe metinlerin benzerlikleri ve eşleme üzerinde çalışmıştır. Benzerliklerinin hesaplanmasında sık karsılasılan bazı hata durumları modellenerek yeni bir benzerlik hesaplama yöntemi geliştirilmiştir. Bu yöntem özellikle yazım yanlıslarını algılayıp, metinlerin benzerliklerini daha tutarlı bir sekilde hesaplamaktadır. Metin olarak ifade edilen kavram, birkaç harften olusan bir kelime olabilecegi gibi yüzlerce kelimeden olusan paragraf gibi uzun bir metin parçası da olabilir [7].

Cringean, England, Manson ve Willet (1989) farklı dosya yapıları kullanan dökümanlarda hızlı, fonksiyonel ve paralel metin arama üzerinde çalışmışlardır. Arama işleminde çoklu işlemci kullanarak paralelliği sağlamışlardır [8].

Navarro (1998), doktora tezinde, hatalı olabilecek metinlerde en uygun arama teknikleri üzerinde; metindeki hata olasılığını dikkate alarak çalışma yapmıştır. Tipik metin arama yerine hatalarında göz önünde bulundurulması tezi bu alandaki diğer çalışmalardan daha değerli hale getirmiştir. Çalışma sonucunda değerli sonuçlar elde edilmiştir ve hata toleransından dolayı bu çalışma bir çok alanda uygulanma imkanı bulmuştur [9].

Metin aramaları, text editörlerinden moleküler biyolojiye kadar bir çok alanda kullanılmaktadır. Bir text dosyasında yapılan arama, DNA da herhangi bir parçanın aranmasına benzetilmektedir. Bir çok editör hatalı metin aramadaki parametre çokluğundan dolayı en uygun yada en yakın olanı aramak yerine klasik arama yöntemlerini kullanmaktadır. Wu ve Manber (1991) hızlı ve esnek olabilecek ve bir çok parametreyi dikkate alacak bir algoritma üzerinde çalışmış ve başarılı sonuçlar elde etmişlerdir [10].

Benzer (2007), arama algoritmalarının performansını kıyaslanması üzerine çalışma yapmıştır. Arama islemi yapay zeka alanın önemli bir parçasıdır. Bir problemin çözümü için aramak ve en uygun çözümü bulmak gerekir. Genel olarak arama algoritmaları iki ana baslık altında toplanmıstır. Bunlar; uninformed ve informed aramalardır. Uninformed aramalara kör aramalar da denmistir. Bunun sebebi arama yaparken herhangi bir bilgi kullanmamasıdır. Informed algoritmaları ise arama yaparken daha basarılı olmaktadır. Bunun sebebi arama yaparken bazı bilgileri kullanmasıdır. Bu arama kategorisine sezgisel (heuristic) aramalar da denmektedir. Arama algoritmalarının çalısmalarını incelemek için birbirleriyle kıyaslama islemi yapılmıştır [2].

Alpdoğan (2007), yüksek lisans tezinde dökümanları içeriklerine göre otomatik olarak Sınıflandırılması üzerinde çalışmıştır. İnternet üzerinde web sayfalarının sayısı, büyük bir hızla artmakta ve otomatik arama motorları, arama sorgularına isabetli cevaplar vermekte yetersiz kalmaktadırlar. Dizin siteleri, bütün web sayfalarını değerlendirmeye yetisememektedir, dolayısıyla dizinlerin kalitesi ve kapsamı azalmaktadır. Ayrıca, bağlantılar güncelliğini kaybetmektedir. Öte yandan, bilgisayarlarda saklanan dokümanların sayısı ve hiyerarsisi de artmaktadır. Sonuç olarak web sayfalarının ve dokümanların otomatik olarak sınıflandırılması daha fazla önem kazanmaktadır. Alpdoğan (2007), çalışmasında özellikle yüksek boyutlu verilerde basarılı olan ve danısmansız öğrenme özelliğine sahip Kendinden Düzenlenen Haritalar (SOM) algoritması kullanılarak bir sınıflandırma sistemi gelistirmiştir. Kendinden düzenlenen haritalar algoritması ile elde edilen sonuçlar etkin bir sınıflandırma yöntemi olan hiyerarsik sınıflandırma ile karşılaştırmıştır. Her

iki algoritmada da dokümanı ayırt edici kelimelerin ön plana çıkarılması için uygun bir etiketleme yöntemi uygulamıştır. Sınıflandırma isleminden önce dokümanlardaki durak kelimelerinin temizlenmesi, çok ve az tekrar eden kelimelerin temizlenmesi, kelimelerin indekslenmesi, ağırlık vektörlerinin bulunması, ağırlık vektörlerinin aynıboyuta getirilmesi, normalizasyon islemleri yapılmıştır [11].

Türkeeş (2007), yüksek lisans tezinde, Türkçe için doğal dil işleme destekli bir bilgi erişim sistemi tasarımı yapmıştır. Bu bilgi erişim sistemi, Türkçenin özelliklerini ve yapısını göz önüne alarak, bilgi erişimindeki basarısını arttırmayı amaçlamaktadır. Tasarlanmıs olan sistemde Türkçenin biçimsel ve sözdizimsel farklılıkları ele alınmış ve bu farklılıkların giderilmesinin bilgi erisimi üzerindeki etkileri irdelenmiştir. Birimsel farklılıklar biçimbirimsel çözümlemeye dayanan gövdeleme, sözdizimsel farklılıklar ise tamlama analizi ile giderilmeye çalışılmıştır. Tamlama analizinde hem istatistiksel hem de dilbilimsel yaklaşımlar incelenmiş ve dilbilimsel tamlama analizi gerçeklenmiştir. İtatistiksel yaklaşımda, birbirini belirli sıklıklarla izleyen ve belge genelinde sıkça geçen sözcük öbekleri tamlama olarak kabul edilmiş, dilbilimsel yaklaşımda ise isim ve sıfat tamlamaları bulunmaya çalışılmıstır [12].

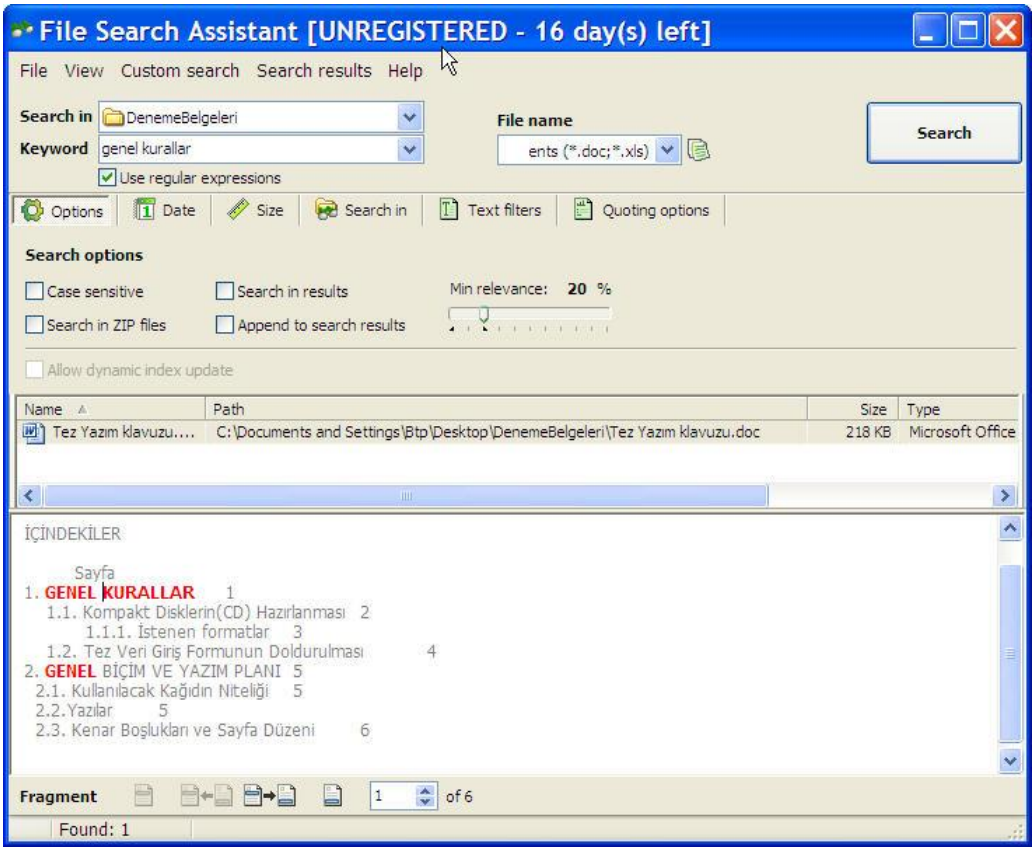
Okur (2007), yüksek lisans tezi kapsamında, Türkçe dilinin özelliklerini kullanarak yapılacak olan dizinleme işlemi sırasında kullanılabilecek olan gövdeleme algoritmaları doğru sonuç döndürme özelliklerine göre kıyaslanmıştır. Bilgi erişim sistemleri, bilgi içeren belge kümelerini dizinleyerek, sorgu yapılmasına olanak tanıyan, doğru ve hızlı bir şekilde istenilen bilgilere erişilmesine olanak sağlayan sistemlerdir. Bu sistemlerdeki en öne çıkan gereksinim, doğru ve hızlı bir Çekilde belgelerin dizinlenmesidir. Microsoft.NET platformunda Visual Basic.NET programlama dili kullanılarak bir arama sistemi geliştirilmiştir [13].

2.2. Diğer Metin Arayıcılardaki Tasarımlar

Bizim buradaki metin arayıcısının, tasarımını en akılcı bir şekilde yapabilmek için benzer çalışmalar yakından incelenmiştir ve kayda değer olanlar bu bölümde yeterince ayrıntılı olarak ele alınmıştır.

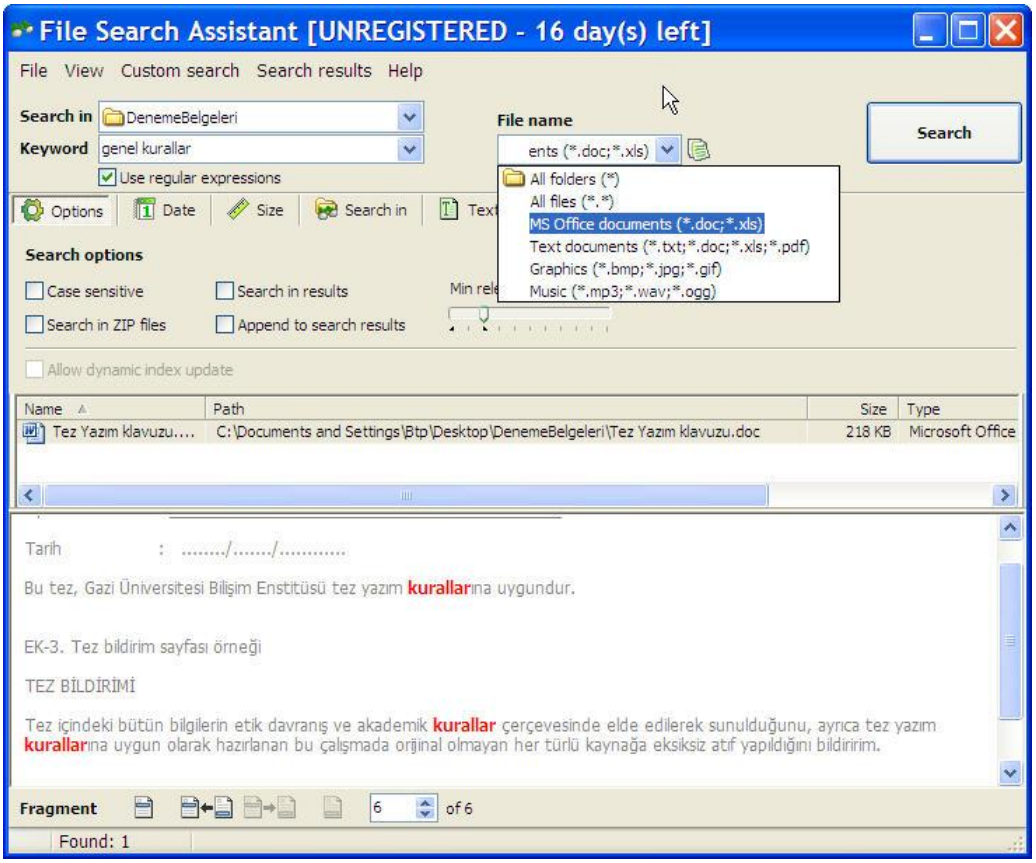
2.2.1. File Search Assistant

Bu program kısaca FSA olarak anılmaktadır. Aşağıdaki şekillerde FSA ara yüzleri sunulmuştur. Windows ortamında çalışan bu program Resim 2.1 de görüleceği üzere “Search in" kutusunda seçilen klasör içerisinde Keyword kutusuna girilen kelime veya kelimelere göre Search butonu ile arama yapmaktadır. Arama sonuçları ekranda iki bölümde gösterilmektedir. Birinci bölümde bulunan dosya isimleri, diskte bulunduğu yer, boyutu, tipi şeklinde listelenmektedir. İkinci bölümde ise listelene dosyalar içerisinden seçili dosyada bulunan bölümler işaretlenmiş olarak gösterilmektedir. Aynı dosya içerisinde arama kriterine uygun bulunan sonuç birden fazla ise bunların sayıları görülebilmekte ve Next Fragment, previos fragment, first fragment, last fragment butonları ile parçalar arasında geçişler yapılabilmektedir.



Resim 2.1. File search assistant arayüzü

Resim 2.2' de görüleceği üzere arama yapabileceği birçok dosya türü mevcuttur. Microsoft Office (*.doc,*.xls), Text documents (*.txt,*.doc,*.xls,*.pdf) seçenekler görülmektedir. Bu yönü ile avantajlı olan bu program, arama seçeneklerinde ise aynı başarıyı gösterememiştir. Search options (arama seçenekleri) kısmında görüldüğü üzere büyük-küçük harf duyarlı (Case sensitive), sıkıştırılmış dosyalarda ara (Search zip files), sonuçlarda ara (search in result), sonuçlara ekle (Append to search result) seçenekleri mevcuttur. Hemen Keyword kutusu altında görülen Use regular expression seçeneği seçime bırakılmıştır.

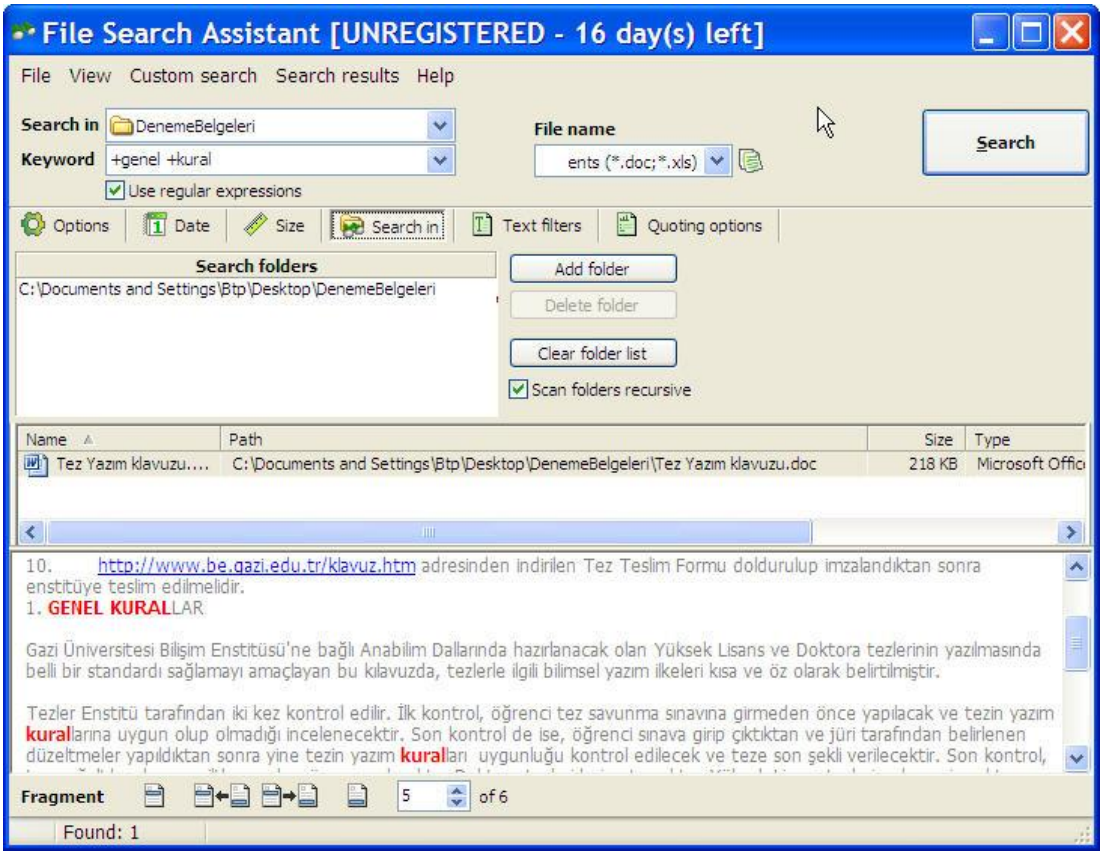


Resim 2.2. File Search Assistant Ara yüzü-2

Bizim uygulamamızda kullandığımız ve önemsediğimiz regular expression ile arama yönteminden bahsedecek olursak Resim 2.3' de görüleceği üzere "+" ve "-", " " karakterleri kullanılarak birden fazla kelime ile arama yapılabilmektedir.

Regular expressions ile ara seçeneklerine örnekler şunlardır.

1. +kelime1 +kelime2 +kelime3 : Kelimelerden herhangi biri anlamındadır. "+" karakteri VEYA mantıksal bağlacı işlevi görmektedir.
2. +kelime1 –kelime2 : Kelime1 geçen ancak kelime2 geçmeyen anlamındadır. "-" karakteri DEĞİL mantıksal bağlacı işlevi görmektedir.
3. +kelime1 +"kelime2 and kelime3" : Kelime1 veya kelime2 ve kelime3 ün birlikte geçtiği anlamındadır.



Resim 2.3. FSA da regular expression seçenekleri

Programın yardım kısmında anlatılanların bir kısmının çalışmadığı, özellikle "–" ve "and " işleçlerinin çalışmadığı görülmüştür.

Bu programın güçlü ve zayıf tarafları aşağıdaki belirtilmiştir.

Güçlü tarafları;

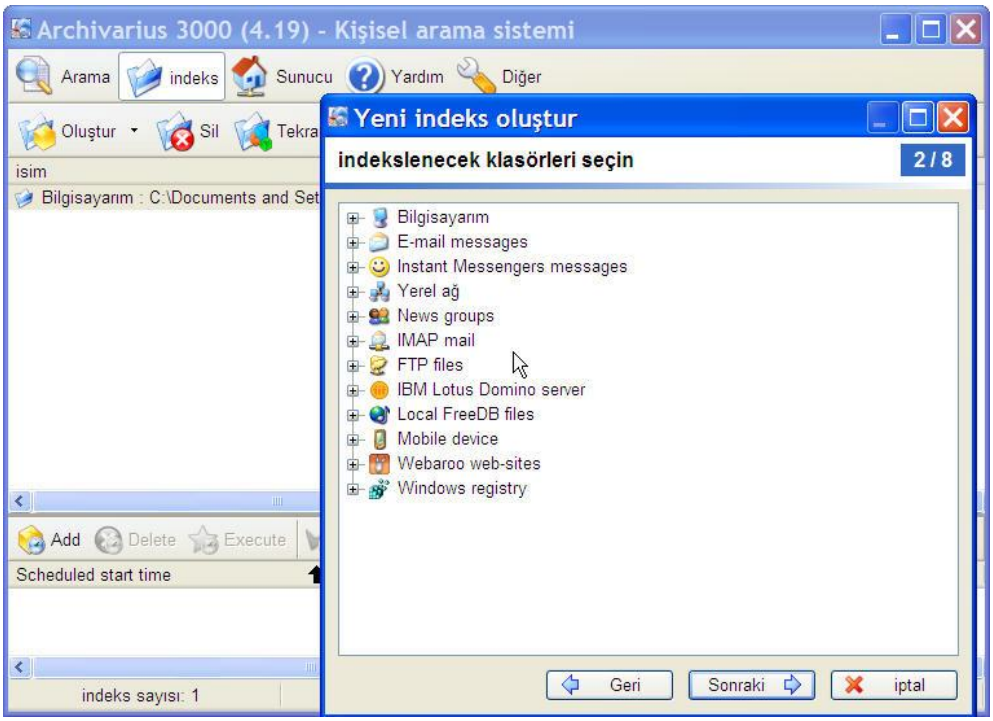
- Çok yaygın olarak kullanılan metin dosyalarında arama yapabilmesi
- Kriterlere uygun olarak bulunan dosyaların listelenmesi
- Türkçe kelimelerde problem yaşanmaması
- Kriterlere uygun bulunan metin parçalarının aynı ekranda görüntülenmesi
- Düzenli ifadelerin kullanılarak çoklu kelimelere göre arama yapabilmesi

Zayıf tarafları;

- Düzenli ifadeler (regular expression) kullanılarak yapılan arama seçenekleri tam çalışmadığından bu yönü eksiktir.
- Kriterlere uygun bulunan sonuçlar programın kendi editöründe parça metin şeklinde gösterildiğinden orijinal belgenin tümünde çalışma yapılamamaktadır.
- Bulunan sonuçlar text editöründe gösterildiğinden orijinal belge içindeki resim, şekil, tablo v.s görülememektedir.

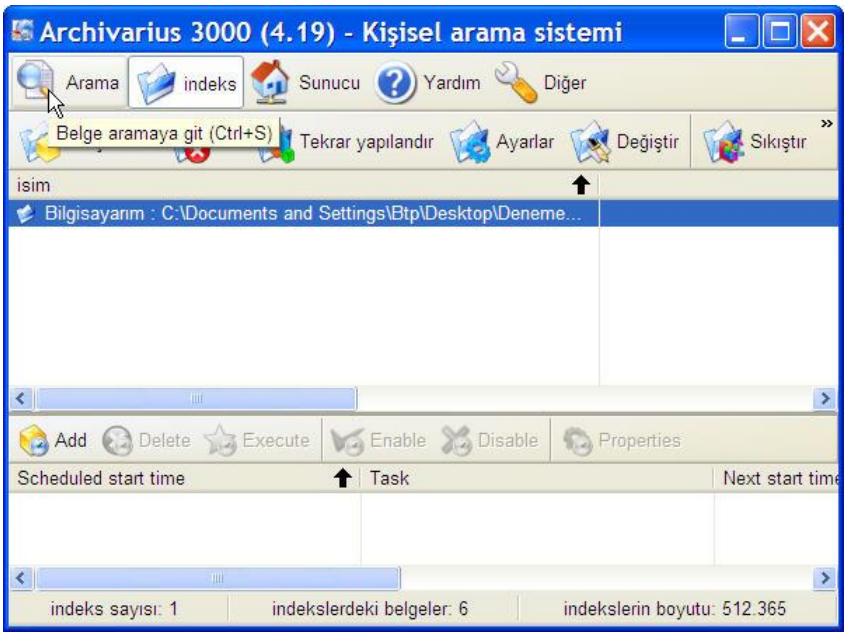
2.2.2. Archivarius 3000

Windows ortamında çalışan bu program Resim 2.4'de görüleceği üzere öncelikle arama yapılacak konumu indeksleyerek işe başlamaktadır. Seçilen konumdaki dosyaları kendi içinde bir algoritma ile indekslemekte, daha sonra aramaları bu indeks üzerinden gerçekleştirmektedir. Oldukça çok amaca yönelik hazırlanmış olan bu programın biz metin dosyaları tarafıyla ilgilendiğimizden, diğer kısımlarına değinmeyeceğiz.



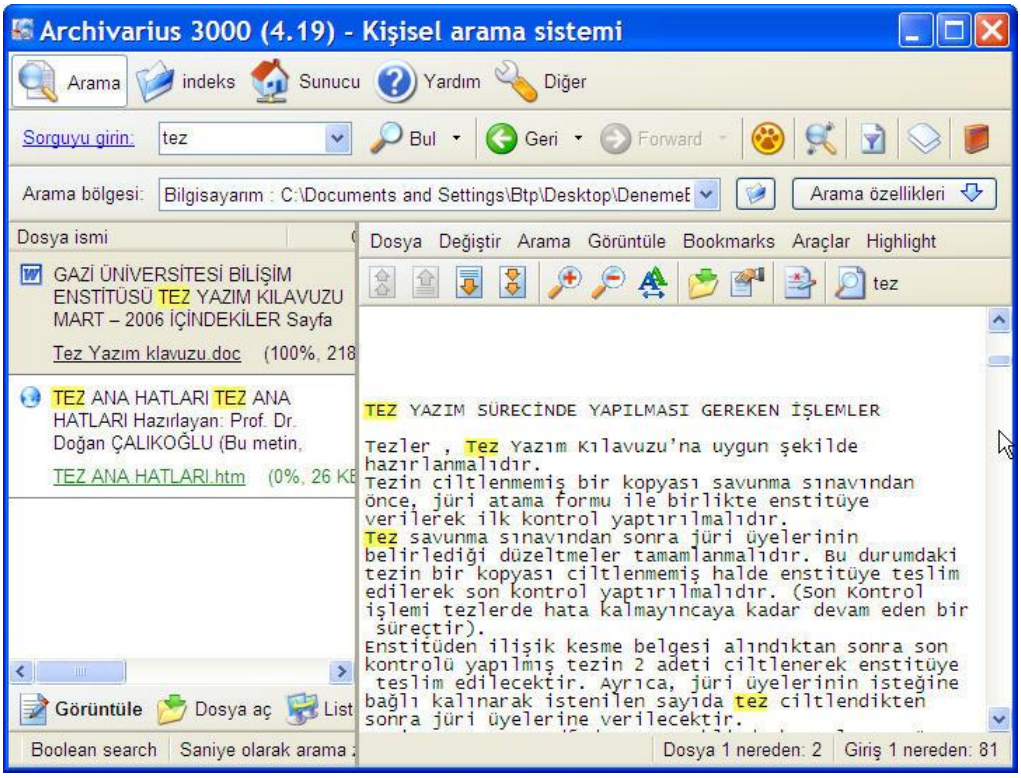
Resim 2.4. Archivarius 3000 Arayüz-1

Resim 2.5'de görüleceği üzere deneme klasörü altında 6 adet dosya indekslenmiştir. Buradan sonra anlatacaklarımız arama sekmesi altındaki ekran görüntüsü üzerinden gerçekleşecektir.



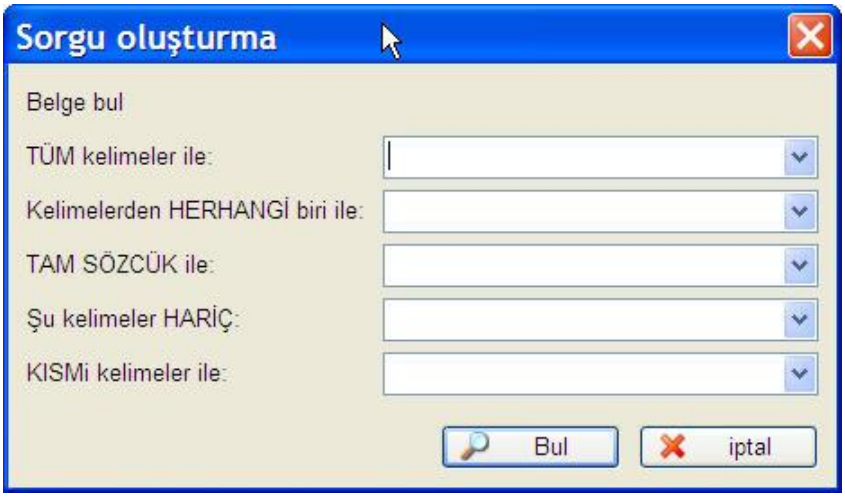
Resim 2.5. Archivarius 3000 arayüz-2

Arama butonu aktif durumda iken Resim 2.6'da görüldüğü üzere ekran dikey olarak iki kısma ayrılmıştır. Menülerin Türkçe olduğu kullanımı kolay bir programdır. "Sorguyu girin" kutusuna yazılan kelime yada kelimelere göre belge içinde arama yapmaktadır. Kriterlere uygun olarak bulunan dosyalar ekranın sol tarafında listelenmektedir. Ekranın sağ tarafında ise ilgili dosyanın tümü bir metin editörü ile açılmaktadır. Bulunan kelimeler burada renkli olarak işaretlenmiştir. Editör üzerinde ileri, geri butonları ile ilgili kelimeler üzerine gidilebilmektedir. Editör içinde ayrıca arama menüsü vardır. Ancak burası basit arama yapabilmekte, tercih sunamamaktadır.



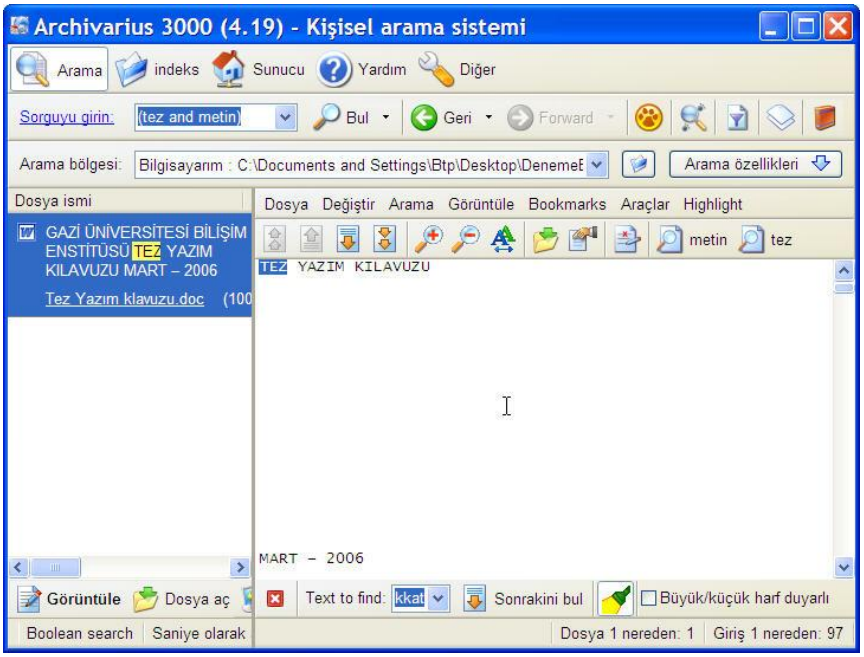
Resim 2.6. Archivarius 3000 Ara Yüz-3

Bu programı öne çıkaran sorguyu girin seçenekleridir. Resim 2.7'de görüleceği üzere "tüm kelimeler ile", "Kelimelerden herhangi biri ile", "tam sözcük ile", "şu kelimeler hariç", "kısmi kelimeler ile" seçenekler mevcuttur.



Resim 2.7. Sorgu seçenekleri

Resim 2.8'de görüleceği üzere buraya girilen ifadeler kod kısmında yapılan bir işlem ile "VE", "VEYA" mantıksal bağlaçlara dönüştürülmektedir.



Resim 2.8. Sorguya göre arama sonucu

Bu programın güçlü ve zayıf tarafları aşağıda belirtilmiştir.

Güçlü tarafları;

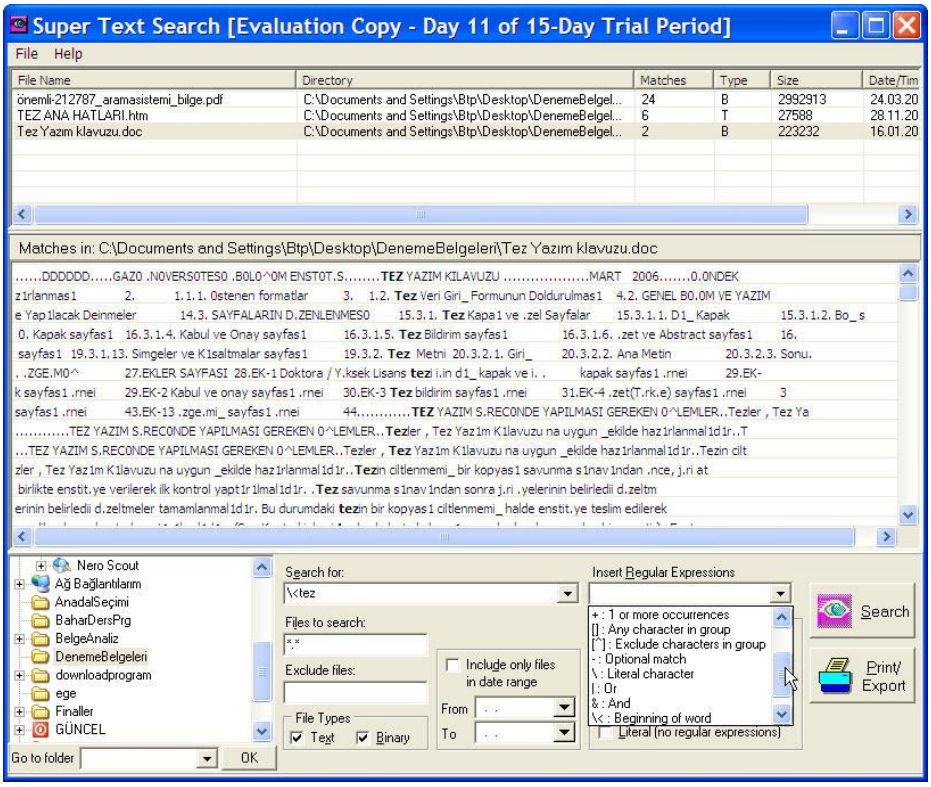
- Menülerinin Türkçe olabilmesi
- Çok yaygın olarak kullanılan birçok dosya türünde arama yapabilmesi
- Kriterlere uygun olarak bulunan dosyaların listelenmesi
- Türkçe kelimelerde problem yaşanmaması
- Kriterlere uygun bulunan metin parçalarının aynı ekranda görüntülenmesi
- Arama seçeneklerinin gelişmişliği sayesinde çoklu kelimelerle arama yapabilmesi
- Kritere uygun bulunan belgenin tümünü kendi içinde metin göstericisi ile açıyor olması.
- Editör içinde bulunan kelimeler üzerinde ileri, geri rahatlıkla hareket ediliyor olması.

Zayıf tarafları;

- Gelişmiş arama seçeneklerinin belge içinde kullanılamıyor olması.
- Gelişmiş arama seçeneklerinin tam manasıyla çalışmamakta. Hatalı sonuçlar döndürmektedir.
- Program içindeki tüm belgenin gösterildiği metin düzenleyicisinin sadece düz metinleri göstermesi. Orijinal belge içindeki resim, şekil, tablo v.s görülememektedir.

2.2.3. Super Text Search

Resim 2.9'da görüleceği üzere seçilen klasör içerisinde bulunan metin dosyaları ve binary dosyalar üzerinde belge içi arama yapabilmektedir. Ekran yatay olarak ikiye ayrılmış olup, birinci kısımda dosya isimleri listelenirken, ikinci kısımda ilgili belge içeriği gösterilmektedir. Belge içinde bulunan kelimeler koyu olarak işaretlenmektedir.



Resim 2.9. Super text search ara yüzü

Bu programı diğerlerinden ayıran en belirgin tarafı düzenli ifade (regular expression) seçeneklerinin detaylandırılmış olması ve kullanıcının kullanımına bırakılmış olmasıdır. Yapılan denemeler sonucunda bu seçeneklerin doğru çalıştığı gözlenmiştir. Birden fazla kelime ile detaylı arama yapılabilmektedir.

Bu programın güçlü ve zayıf tarafları aşağıda belirtilmiştir.

Güçlü tarafları;

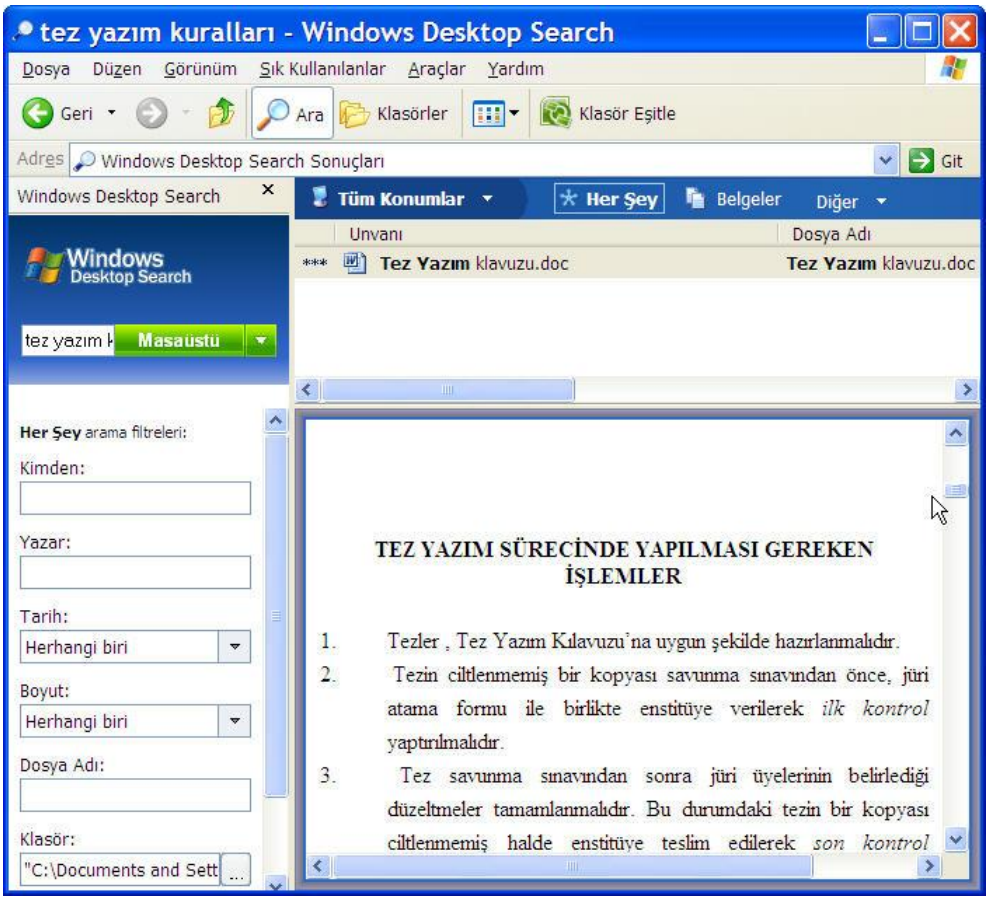
- Çok yaygın olarak kullanılan birçok dosya türünde arama yapabilmesi
- Kriterlere uygun olarak bulunan dosyaların listelenmesi
- Kriterlere uygun bulunan metin parçalarının aynı ekranda görüntülenmesi
- Arama seçeneklerinin gelişmişliği sayesinde çoklu kelimelerle detaylı arama yapabilmesi
- Düzenli ifade seçeneklerinin kullanıcının seçimine bırakılmış olması.
- Kritere uygun bulunan belgenin tümünü kendi içinde metin göstericisi ile açıyor olması.
- Editör içinde bulunan kelimeler üzerinde ileri, geri rahatlıkla hareket ediliyor olması.

Zayıf tarafları;

- Menülerinin Türkçe olmaması
- Türkçe dil desteğinin olmaması yüzünden Türkçe karakterlerde problem yaşanması (Ç,İ,Ğ,Ö,Ş,Ü)
- Düzenli ifadelerle arama seçeneklerinin kullanımının zorluğu
- Program içindeki tüm belgenin gösterildiği metin düzenleyicisinin sadece düz metinleri göstermesi. Orijinal belge içindeki resim, şekil, tablo v.s görülememektedir.

2.2.4. Windows Desktop Search

Microsoft ürünü olan ve ücretsiz kullanıma sunulan bu program belge içi arama yapabilmektedir. Seçilen klasör içerisinde birçok dosya türünde arama yapabilen bu program Resim 2.10 da görüleceği üzere bulunan dosyalar aynı ekranda listelenirken, alt tarafta ilgili belge içeriğin biçimi bozulmayacak şekilde bir metin gösterici ile gösterilmektedir. Bu kısım diğer bahsedilen programlarda olmayan bir özelliktir. Bu kısım bir avantaj sağlarken, metin gösterici içerisinde bulunan kelimelerin işaretlenmemiş olması ise çok büyük bir eksikliktir. Yani sadece aranan ifade bu belgede geçiyor ise belgenin tümünü burada göstermektedir. Belge içinde nerelerde geçtiği görülememektedir. Arama seçenekleri açısından da çok basit kalmaktadır.

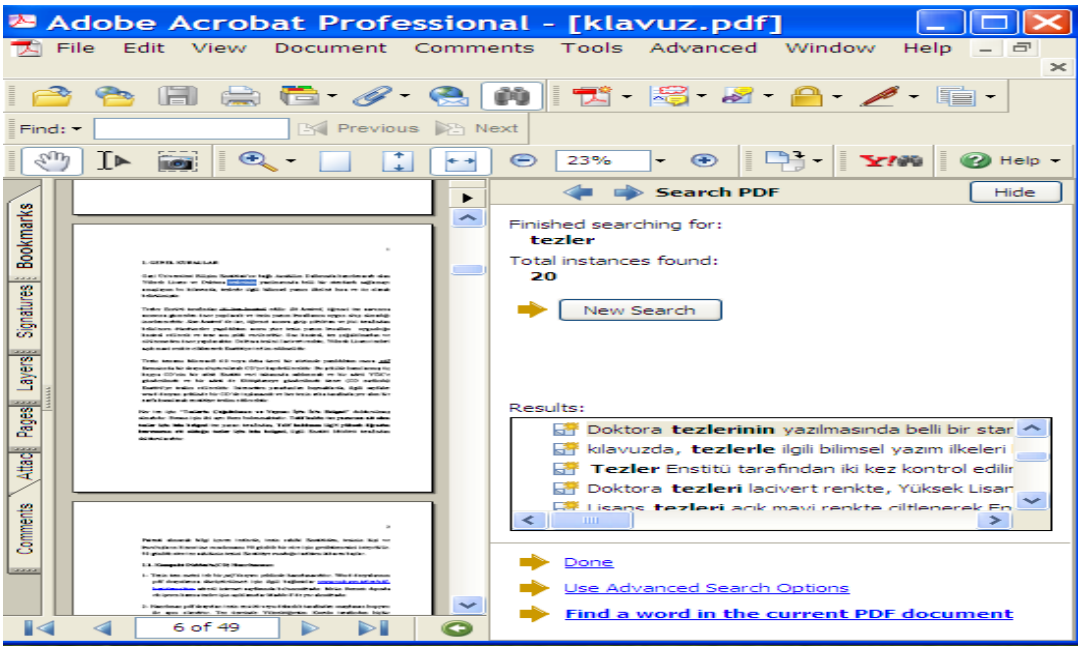


Resim 2.10. Windows desktop search ara yüz

2.2.5. Adobe Acrobat Profesional Programı

Adobe firması tarafından üretilen ve kendine özgü dosya formatı olan pdf uzantılı belgeleri açmakta kullanılmaktadır. Resim 2.11' de görüleceği üzere bu editörün "search" butonu ile arama yapılabilmektedir. Bu arama işleminin "find" butonundan farkı vardır. "Find" işleminde belge içerisine ilk bulduğu yere konumlanıp, ileri geri butonları ile hareket etmektedir. "search" seçeneğinde ise tüm belge kritere uygun olarak taranmakta, bulunanlar sağ tarafta sayfa numarası indeksi ile birlikte listelenmektedir. Liste içerisinde istenen istenen cümle üzerinde çift tıklandığında direk belge içerisinde ilgili yere konumlanabilmektedir. Bu özellik çalışma anında kolaylık sağladığından, BELAR ın tasarımında da buradan faydalanılmıştır. Ancak gelişmiş arama seçenekleri çoklu aramayı desteklemediğinden birden fazla kelime ile arama yapamamaktadır.

Örneğin "tez" ve "teslim" kelimelerinin birlikte geçtiği cümleleri aramak istiyorum. Aranacak ifade olarak "tez teslim" yazdığımda belge içerisinde aynen girildiği gibi geçen cümlelere ulaşabilmektedir. Oysa tez ve teslim kelimeleri aynı cümlede geçebilir, ancak peş peşe gelmeyebilir.



Resim 2.11. Adobe Acrobat Profesional Arayüz

2.3. Microsoft Word Programı Bul Seçeneği ve Joker Karakterler

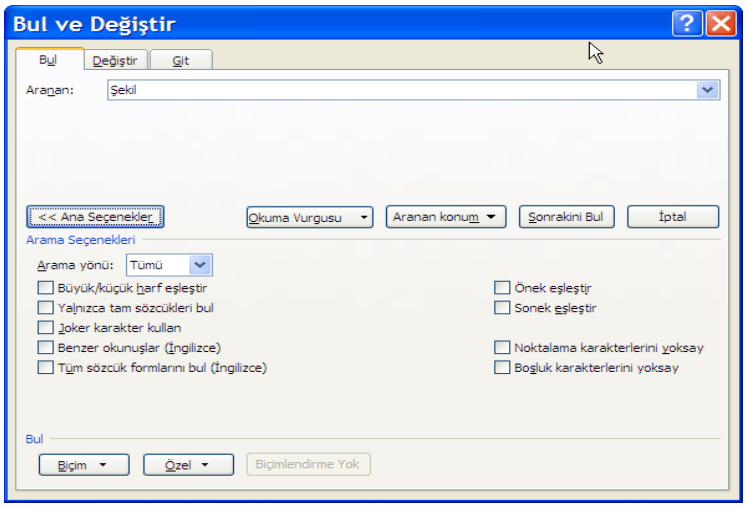
Bu bölümde Microsoft Word programı bul seçeneğinin nasıl çalıştığı ve joker karakterlerin kullanımı anlatılmıştır.

Metin bulma

Belirli bir sözcük veya deyimi arayabilirsiniz.

- Giriş sekmesinin Düzenleme grubunda Bul'u tıklatın.
- Aranan iletişim kutusuna aramak istediğiniz ifadeyi yazın.
- Aşağıdakilerden birini yapın:
  - Sözcük veya tümceciğin her oluşumunu bulmak için Sonrakini Bul seçeneğini tıklatın.
  - Belirli bir sözcük veya tümceciğin tüm oluşumlarını bir kerede bulmak için, sırasıyla Tümünü Bul ve Ana Belge seçeneklerini tıklatın.

Resim 2.12' de görüleceği üzere Aranan kısma girilen ifade "Büyük/Küçük harf eşleştir", "Yalnızca tam sözcükleri bul", Seçeneklerinin dışında "Joker karakter kullan" seçeneği ile de arama yapılabilmektedir.

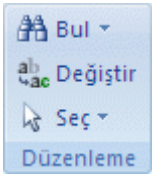


Resim 2.12. Microsoft word programı bul ara yüzü

Belirli harfleri bulmak için joker karakterler kullanarak arama yapma

Metinde arama yapmak için joker karakterler kullanabilirsiniz. Örneğin, yıldız işareti (*) joker karakterini kullanarak bir karakter dizisini arayabilirsiniz ("s*t" "sat" ve "saadet"i bulur).

- *Giriş* sekmesinin Düzenleme grubunda Bul'u veya Değiştir'i tıklanır.



Resim 2.13. Bul ve Değiştir

- Joker karakterlerini kullan onay kutusu seçilir.

  Joker karakterlerini kullan onay kutusunu göremiyorsanız, Diğer'i tıklatın.

- Aşağıdakilerden birini yapın:
  - Listeden bir joker karakter seçmek için Özel'i tıklatın, bir joker karakteri tıklatın ve sonra Aranan iletişim kutusuna ek metni yazın. Daha fazla bilgi için Bulmak ve değiştirmek istediğiniz öğeler için joker karakterler tablosuna bakın.
  - Joker karakteri doğrudan Aranan kutusuna yazın. Daha fazla bilgi için Bulmak ve değiştirmek istediğiniz öğeler için joker karakterler tablosuna bakın.
- Öğeyi değiştirmek istiyorsanız, Değiştir sekmesini tıklatın ve yapılacak değişikliği Yeni değer kutusuna yazın.
- Sonrakini Bul, Tümünü Bul, Değiştir veya Tümünü Değiştir seçeneğini tıklatın.

Bulmak ve değiştirmek istediğiniz öğeler için joker karakterler

Joker karakter kullan onay kutusu seçiliyse, Word yalnızca belirttiğiniz tam metni bulur. Büyük/küçük harf eşleştir ve Yalnızca tam sözcükleri bul onay kutularının, bu seçeneklerin otomatik olarak etkinleştirilmiş olduğunu belirtecek şekilde kullanılamaz (soluk) olduğuna dikkat edin. Bu seçenekleri devre dışı bırakamazsınız.

Joker karakter olarak tanımlanmış bir karakteri aramak için, bir karakterden önce ters eğik çizgi (\) yazın. Örneğin, bir soru işareti bulmak için \? yazın.

Joker karakterleri ve metni gruplandırmak ve değerlendirme sırasını belirtmek parantez için kullanabilirsiniz. Örneğin, "önizleme" ve "önemseme" sözcüklerini bulmak için <(ön)*(me)> yazın.

\n joker karakterini bir ifadeyi aramak ve onu yeniden düzenlenmiş bir ifadeyle değiştirmek için kullanabilirsiniz. Örneğin, Aranan iletişim kutusuna (Mersin) (Sibel) yazın, Yeni değer kutusuna da, \2 \1 yazın. Word Mersin Sibel ifadesini bulacak ve Sibel Mersin ile değiştirecektir.

Word içerisinde kullanılabilecek joker karakterler bir tablo şeklinde Çizelge 2.1’ de verilmiştir.

Çizelge 2.1. Joker karakterler

Bulunacak olan	**Yazılacak olan**	**Örnek**
Tek bir karakter	**?**	s?t sat ve set bulacaktır.
Karakter dizisi	*****	s*t set ve saadet bulacaktır.
Sözcüğün başı	<	<(kül) küllük ve külüstür sözcüklerini bulabilir, ancak dökülen sözcüğünü bulmaz.
Sözcüğün sonu	>	(in)> derin ve serin sözcüklerini bulabilir, ancak ince sözcüğünü bulmaz.
Belirlenen karakterlerden biri	**[ ]**	s[eü]t set ve süt sözcüklerini bulur.

Bu aralıktaki herhangi bir karakter	**[-]**	[s-y]on son ve ton sözcüklerini bulur. Aralıklar artan düzende olmalıdır.
Köşeli ayracın içindeki aralıkta bulunanlar dışında herhangi bir karakter	**[!x-z]**	a[!a-k]ın alın ve atın sözcüğünü bulabilir, ancak akın sözcüğünü bulamaz.
Önceki karakterin veya ifadenin *n* kez yinelendiği yerler	**{n}**	sa{2}t saat sözcüğünü bulur, ancak sat sözcüğünü bulmaz.
Önceki karakterin veya ifadenin en az *n* kez yinelendiği yerler	**{n,}**	sa{1,}t sat ve saat sözcüklerini bulur.
Önceki karakterin veya ifadenin *n* ile *m* arasında yinelendiği yerler	**{n,m}**	10{1,3} 10, 100 ve 1000 bulur.
Önceki karakterin bir veya daha fazla yinelendiği yerler	**@**	fi@l fil ve fiil sözcüklerini bulur.

Bölüm 2.2’ de anlatılan programların her birinin kendine özgü güçlü tarafları olmakla birlikte, belge içi aramada çoklu ve detaylı aramada yetersiz kalmıştır. Türkçe metinler üzerinde problem çıkarmadan, web sayfaları arama motorlarında arama yapmaya benzer tarzda, diskte bulunan dosyalarda, belge içi arama yapacak bir program ihtiyacı tespit edilmiştir.

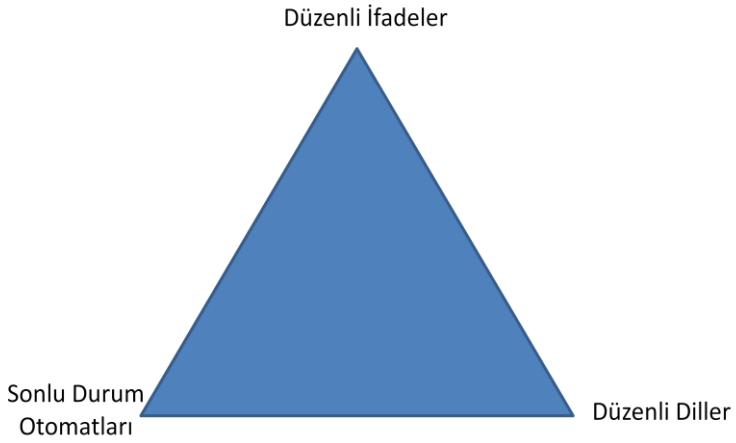
Yapılan incelemeler sonucu BELAR ın tasarımında bunlardan yararlanılmıştır. Bundan hemen sonra gelen bölümümüzde Belge içi arayıcı programımız (BELAR) anlatılmıştır.

2.4. Düzenli İfadeler (Regular Expressions)

Diller, doğal diller ve resmi diller olarak ikiye ayrılır.

Doğal Diller: İnsanların doğduktan sonra ana dil olarak toplumlarından öğrendikleri ve bireyler arası iletişim kurmakta yararlandıkları ifade sistemleri (Türkçe, İngilizce). Bunlar resmi tanıma bağlı değildir. Doğar, gelişir ve hatta Latince dilinde olduğu gibi ölür [21].

Resmi Diller: Kavramsal varlıkları matematiksel (resmi) tanımla küme kavramı kullanılarak ortaya konur. Üyelik kuralları (bir dilde hangi cümlelerin olduğu) sabittir. Yani bir elemanın bir resmi dile ait olup olmadığı sabit bir kuralla ifade edilir [14].



Resim 2.14. Tip-3 Gramerler [15]

Düzenli Diller

- Σ sonlu bir alfabeyi,
- ∅ boş kümeyi,
- ε, {ε} kümesini göstermek üzere;
- Σ üzerinde tanımlanabilen *düzenli dillerin* formel tanımı şöyledir:
  - ∅ düzenli bir dildir.
  - $\forall a \in \sum \cup \varepsilon$, {a} düzenli bir dildir.

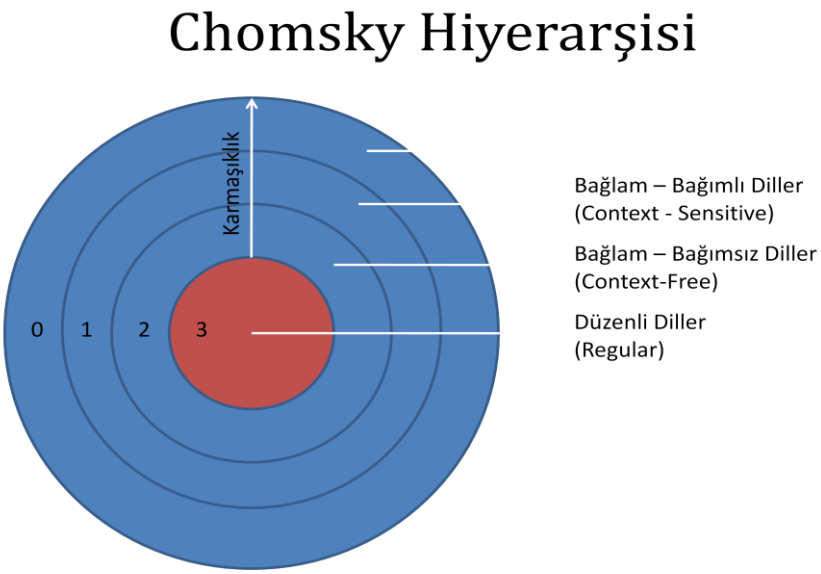
– Eğer L_1 ve L_2 dilleri düzenli diller ise:

a) $L_1 * L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$ *ekleme işlemi (concatenation),*

b) $L_1 \cup L_2$ *birleşim işlemi (union, disjunction, alternation),*

c) L_1^* ise *Kleene sonlandırması (Kleene closure)* ile tanımlanan diller de düzenli dillerdir.

Noam Chomsky'nin dilbilimine getirdikleri Noam Chomsky doğal dilleri anlamına göre sıralamaya sokmuştur. Yaptığı yenilikse, özel dil açılımlarını meta dilinin yardımıyla yeniden adlandırmasıdır. Bu meta dillerden gelişen gramer sınıfları bölümlere ayrılabilir. Bu ayrımın adına da Chomsky Hiyerarşisi denir ve dilbilimin temel yapıtaşlarından birisini bu konu oluşturur. Resmi diller ve Chomsky Hiyerarşi'si, özellikle de karmaşa teorisi ve derleyici yapıların oluşumunda ve bilgi teknolojisinin gelişiminde önemli bir rol oynamıştır. Pinker gibi modern araştırmacılar kendi tezlerini Chomsky üzerine kurmuşlardır [16].



Resim 2.15. Chomsky Hiyerarşisi [16]

Düzenli İfadeler

- Karakter dizileri içinde belirli örüntüleri (pattern) aramayı sağlayan bir dildir.
- İlk olarak 1956 yılında *Stephen C. Kleene* tarafından formel bir model olarak sunulmuştur.
- *Formel tanım*: Karakter katarı kümelerinin cebirsel olarak tanımlanmasını sağlayan bir gösterimdir.
- Sonlu durum otomatları ile tanınan dilleri ifade edebilme gücüne sahiptirler.

Düzenli ifadeler (regular expressions) değişken sayıda karakter dizilerinden oluşulabilecek ancak belirli koşulları sağlayabilen ifadelerdir. Düzenli ifadeler programdaki ihtiyaca göre düzenlenir. Diyelim ki bir text dosyası içinde @ karakteri geçen bütün satırları elde etmek istiyoruz. Burada satırdaki karakterin uzunluğu ve ne olduğu önemli değil; yeter ki @ karakteri olsun. Belirtilen bu satırları elde etmenin çeşitli yolları olabilir. Ancak şartlarımız arttıkça işlemi koda dökmek zorlaşacaktır. Örneğin milyonlarca e-mail adresi olabilir. Ama bir tane e-mail adrsi formatı vardır. Her e-mail adresi mutlaka @ karakteri ve en az bir '.' Karakteri içermelidir. Eğer birden fazla nokta varsa, noktalardan biri mutlaka @ karakterden sonra olmalıdır. gördüğünüz gibi bir karakter dizisinin gerçek bir e-mail adresi olup olmadığını test etmek bir hayli zor. Bu yüzden C# ta bu tür düzenli ifadeleri temsil etmek için Regex sınıfı geliştirilmiştir. Regex sınıfı System.Text.RegularExpressions isim alanında bulunmaktadır. Bir karakter dizisinin, oluşturulan düzenli ifadeye uyup uymadığını belirlemek için ise yine aynı isim alanında bulunan Match isimli sınıftan faydalanılır [17].

2.4.1. Düzenli İfade Kuralları

Basit Düzenli İfadeler

En basit düzenli ifadeler karakterlerin sıralı biçimde dizilmesiyle oluşur [18].

Düzenli ifadeler büyük-küçük harf duyarlıdır [19].

- /bilgisayar/ Arama bilgisayar biliminde en temel işlemlerden biridir.
- /bilgi/ Arama bilgisayar biliminde en temel işlemlerden biridir.
- /ar/ Arama bilgisayar biliminde en temel işlemlerden biridir.
- / en temel/ Arama bilgisayar biliminde en temel işlemlerden biridir.

Karakter Seçimi ve Aralıklar

Kare parantezler '[ ]' arasına yazılan karakterler 'veya' işlemine tabi olurlar [20].

'-' işareti ile kare parantez içinde aralık (range) belirtilebilir.

- /[Bi]lgi/ → Bilgi veya bilgi
- /[klm]/ → k, l veya m
- /[1234567890]/ → herhangi bir rakam
- /[A-Z]/ → Herhangi bir büyük harf
- /[a-z]/ → Herhangi bir küçük harf
- /[0-9]/ → Herhangi bir rakam

'*' karakteri kendinden önce gelen karakterin 0 veya daha fazla kere ardışık olarak tekrarlandığını belirtir [21].

- /kl*m/→ kl, klm, kllm, klllm, ….
- /[0-9][0-9]*/ → bir veya daha fazla sayıda ardışık rakam

'+' karakteri kendinden önce gelen karakterin 1 veya daha fazla kere ardışık olarak tekrarlandığını belirtir.

- /sa+t/ → bir veya daha fazla sayıda a harfi

‘?’ karakteri kendinden önce gelen karakterin seçimlik olduğunu belirtir.

- /bilgi?/ → bilg veya bilgi
- /colou?r/ → color veya colour

‘.’ karakteri tekil herhangi bir karakterin yerine geçebilir.

- /bilg./ → bilge veya bilgi gibi

‘|’ Bu karakter örüntüler arasında ‘veya’ işlevi görür.

- /bilgi|belge/ bilgi veya belge

‘()’ şeklindeki normal parantezler içine alınan ifadeler tek bir karaktermiş gibi işlenir.

- /yüz(er|erler)/ → yüzer veya yüzerler

\b özel karakteri, kullanıldığı yere göre, aranan ifadenin önünde veya arkasında sınırlayıcı (boşluk gibi) karakterleri sınır olarak kabul eder.

- /\btez\b/ → önünde ve arkasında boşluk olan ‘tez’ ifadesini bulur

\B karakteri sınırlandırma olmayan durumu belirtir.

^ karakteri 3 farklı şekilde kullanılabilir.

1. Aralık için olumsuzlama

    • /[^A-Z]/ → büyük harf harici karakter
    • /[^Ss]/ → S veya s harici karakter
    • /[^\.]/ → nokta harici karakter

2. Satır başına bağlama

   • /^Asya/ → satır başında ‘Asya‘ olan durum

3. Normal karakter olarak kullanım

   • /[e^]/ → e veya ^
   • /a^b/ → a^b örüntüsü

$ karakteri ise normal karakter olarak kullanımı haricinde satır sonuna bağlama içinde kullanılır.

- /geldiler\.$/ → satır sonunda ‘geldiler’ olan durum

Sayaçlar

Herhangi bir karakterin ne miktarda tekrarlanabileceğini belirten ifadelerdir [22-23-24].

- {n} → kendinden önceki karakter n defa ardışık olmalıdır
- {n, m} → kendinden önceki karakter n ile m aralığında ardışık olmalıdır
- {n,} → kendinden önceki karakter en az n kadar ardışık olmalıdır

Operatörlerin Öncelik Sırası

En yüksekten en düşük öncelikli operatöre doğru sıralama şu şekildedir:

1. Parantez →()
2. Sayaçlar →* + ? {}
3. Seriler veya bağlayıcılar → evler ^Yarın gelecek$
4. Veya → | (pipe)

Özel Operatörler

- \d → herhangi bir rakam
- \D → rakam olmayan bir karakter
- \w → alfanümerik veya boşluk karakteri
- \W → alfanümerik olmayan karakter
- \s → boşluk
- \S → boşluk olmayan karakter
- \. → nokta karakteri
- * → * (asterisk) karakteri
- \? → ? Karakteri
- \n → newline karakteri
- \t → tab karakteri

3. BELGE İÇİ ARAYICI (BELAR)

Bu bölümde BELAR ismiyle yazmış olduğumuz belge içi gelişmiş ve kolay arama yapabilen bir yazılım anlatılacaktır. Belge içi arayıcı (BELAR) ismiyle geliştirdiğimiz bu uygulamada amaç Microsoft Word belgelerinin belge içi arayıcımızla açılıp, detaylı arama yapılabilmesini sağlamaktır.

3.1. Kullanılan Araçlar

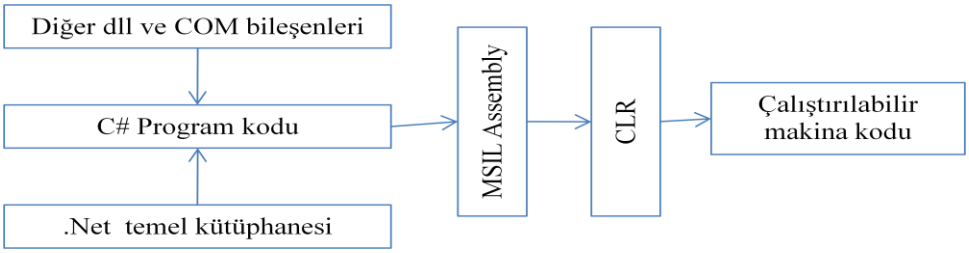
BELAR ın geliştirilmesi sırasında uygulama geliştirme aracı Visual Studio 2008 ve programlama dili olarak C#.NET kullanılmıştır. Framework olarak da .NET 2.0 Framework tercih edilmiştir.

Microsoft .Net ortamında çalışan C# dili tercih etmemizdeki nedenlerden bahsedecek olursak;

- .NET in nesne yönelimli bir dil olması ve çok sayıda sınıfın programcının hizmetine sunulması güzel bir geliştirme ortamı sunuyor.
- C# görsel bir dil olması yönüyle kullanımı ve öğrenilmesi kolaydır.
- XML desteği vardır.
- Güncel bir programlama dilidir.

.NET framework, çok dağıtık (highly-distributed), bileşen yönelimli uygulamaların geliştirilmesini ve yürütülmesini destekleyen bir ortam sunar. .Net framework, farklılık gösteren bilgisayar dillerinin birlikte çalışmasını mümkün kılar ve Windows platformu için güvenlik, taşınabilirlik (programlar açısından) ve ortak bir programlama modeli sunar. [25]

.Net platformunun en önemli taraflarından biride platformdan bağımsız olmasıdır. C# program kodları derleme esnasında öncelikle ara dil (IL) çevrilir. IL kodu çalıştırılmak istendiği zaman ortak dil çalışma platformu .NET CLR (Common Language Runtime) ile makine diline çevrilir. Bu sayede yazılan kodlar CLR ile farklı platformlarda ve işletim sistemlerinde çalışır. Resim 3.1.

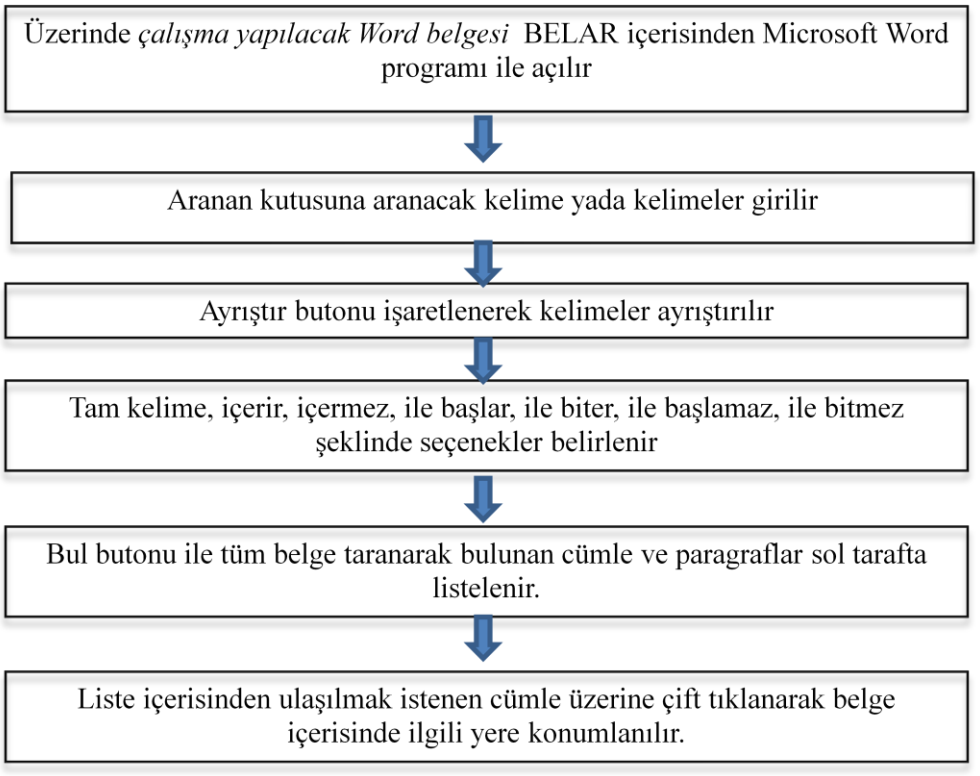


Resim 3.1. .NET programlarının derlenip çalıştırılması [17]

Ortak Dil Çalışma Platformu(CLR), .NET altyapısında programların çalışmasını kontrol eden ve işletim sistemi ile programımız arasında yer alan arabirimdir. Normalde yazdığımız programlar derlenirken makine diline çevrilirdi ve program bu şekilde işletim sistemi ile direk bağlantı kurarak çalışırdı [17].

Resim 3.1' de Belar ın genel tasarımında görüleceği üzere .doc veya .docx uzantılı word belgeleri BELAR içerisinden Microsoft Word ile açılabilmektedir. Dolayısıyla bilgisayarımızda Microsoft Word'ün kurulmuş olması bir ön şarttır. Sonrasında yapılan işlemlerde herhangi bir veri tabanına ihtiyaç duymadan, direk word içerisindeki cümleler elde edilmekte ve bunlar üzerinde arama işlemi yaptırılmaktadır. Bu tür işlemlerde çoğunlukta veri tabanı kullanılmaktadır. Bunun nedeni veri tabanı sorgu komutlarının özellikleridir. Biz ise burada veri tabanı kullanmadan düzenli ifadeleri kullanarak aynı başarıyı gösterdiğimize inanıyoruz.

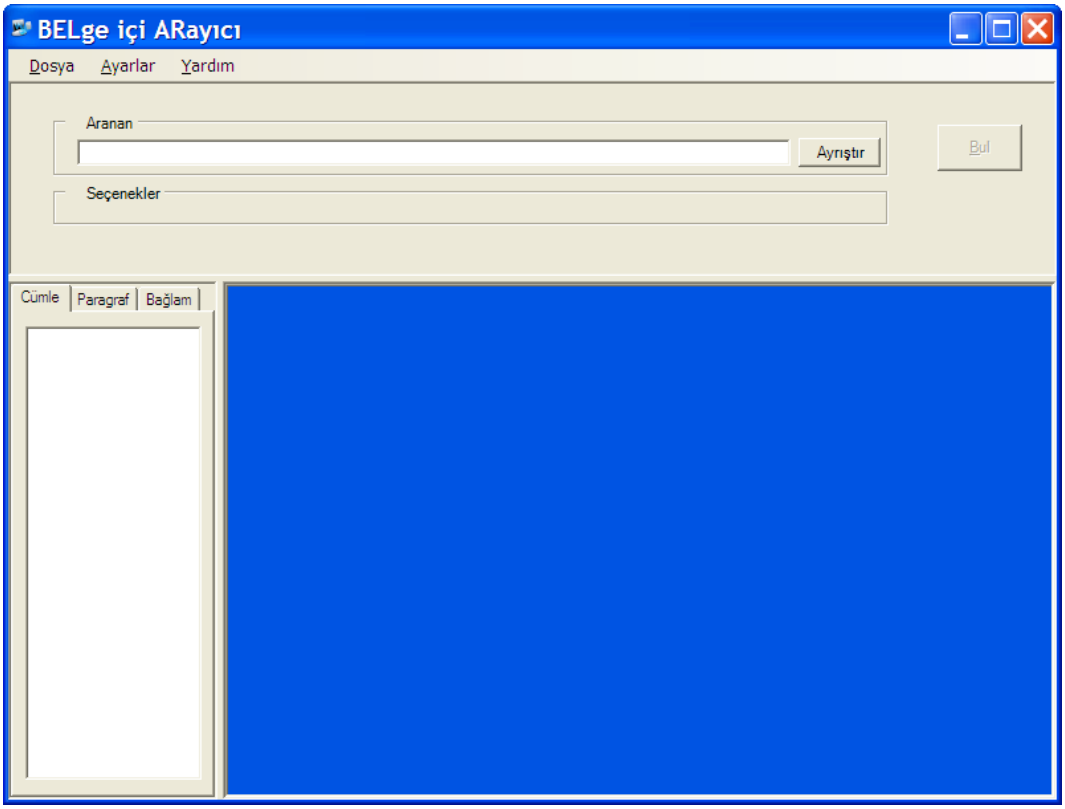
Ayrıca çalışılan belge üzerinde direk konumlanabildiğimiz için aynı anda hem Word programının özelliklerini, hem de BELAR ı aynı anda kullanma imkanı oluyor.



Resim 3.2. BELAR'ın Genel Tasarımı

3.2. Kullanıcı Ara Yüzü

Bu bölümde geliştirdiğimiz uygulamanın (BELAR) kullanıcı ara yüzleri sunulmuştur. Bir Windows uygulaması olarak geliştirilmiş olan BELAR, ilk çalıştırıldığında Resim 3.3. de görülen ekran görüntüsü gelmektedir. Sade bir görünüme sahip olan bu ara yüzü inceleyecek olursak; Dosya ve yardım menülerini, "Aranan" kutusunu , "Ayrıştır" butonunu, pasif durumda olan "Bul" butonunu, sol tarafta ise "Cümle", "Paragraf" ve "Bağlam" sekmelerini görmekteyiz. Temelde ekranımız üçe ayrılarak kullanılmıştır. Arama tercihlerinin belirlendiği yatay bölüm. Bulunan cümle, paragraf ve bağlamların listelendiği dikey bölüm ve açılan belgemizin görüntülendiği mavi renkli geniş bölümdür.



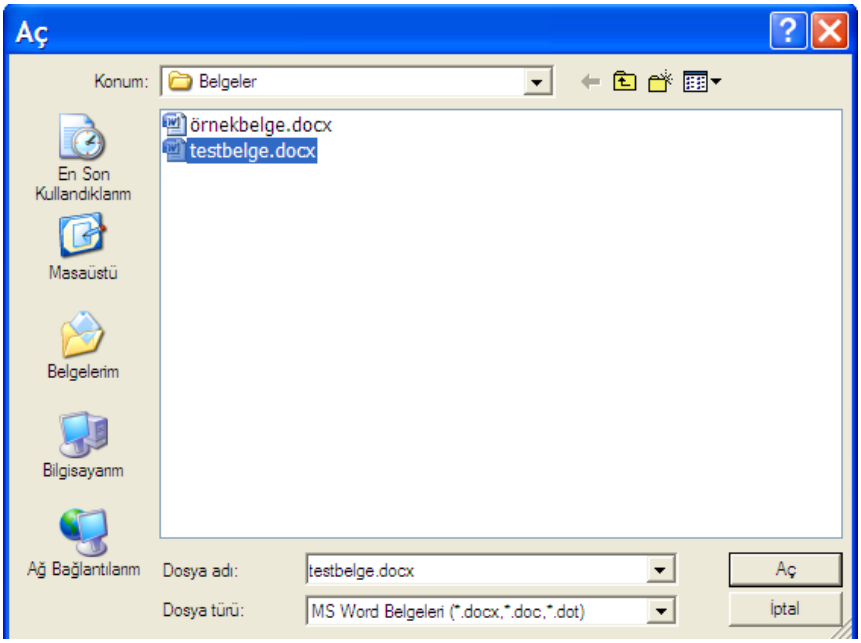
Resim 3.3. BELAR ilk arayüz

BELAR ı kullanabilmek için üzerinde çalışılacak belgenin öncelikle açılması gerekir. Belge açılmadan "Aranan" kutusuna aranacak olan ifade yazılıp "Ayrıştır" butonuna tıklandığında "Bul" butonunun aktif olduğu görülecektir. Ancak "Bul" butonuna tıklandığında ise Resim 3.4' de görülen Hata mesajı ile karşılaşılır.

Resim 3.4. Hata mesajı-1

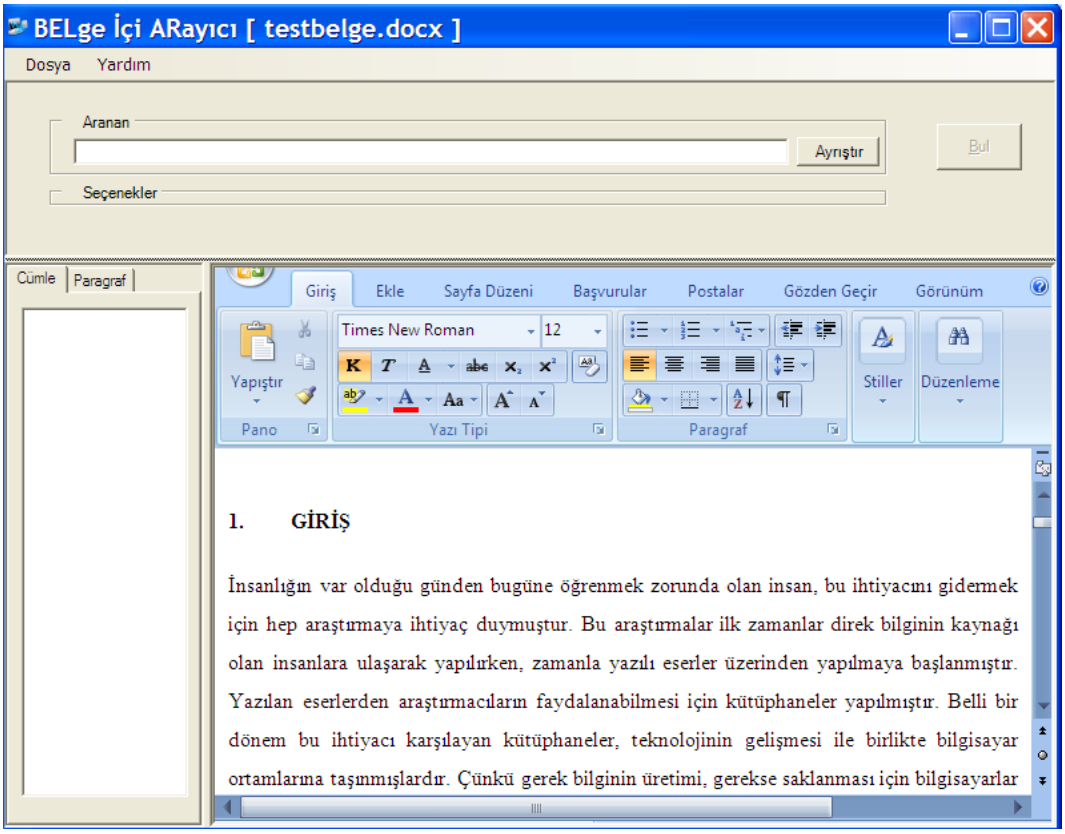
Resim 3.5. Dosya Aç

Üzerinde çalışmak istediğimiz word belgesi Resim 3.5'de görüleceği üzere "Dosya Aç" menüsü veya "Ctrl+O" tuşları ile açılır. Resim 3.6 da görülen ara yüzdende anlaşılacağı üzere .doc, .docx, .dot uzantılı word belgeleri açılabilmektedir.



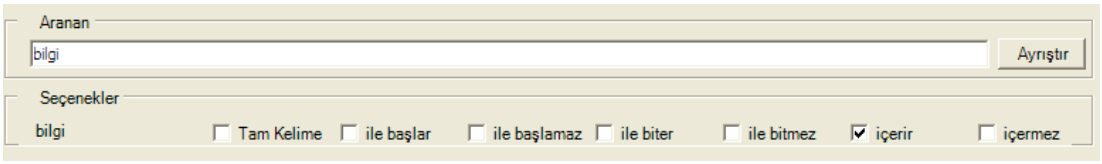
Resim 3.6. Word Belgesi Aç

"Testbelge.docx" isimli MS Word belgesi açıldıktan sonra ekran görüntüsü Resim 3.7 gibi olacaktır. Belge BELAR ın çalışmakta olduğu bilgisayarda kurulu olan Microsoft Word programı ile açılacağından, belge içerisinde BELAR ile arama yapmanın dışında MS Word ün diğer tüm özellikleri kullanılabilecektir. Burada dikkat edilmesi gereken husus, belgenin mutlaka BELAR "Dosya Aç" seçeneği ile açılması gerekmektedir. Eğer MS Word programı içerisinden dosya açılacak olursa, program etkin olarak kullanılamayacaktır.



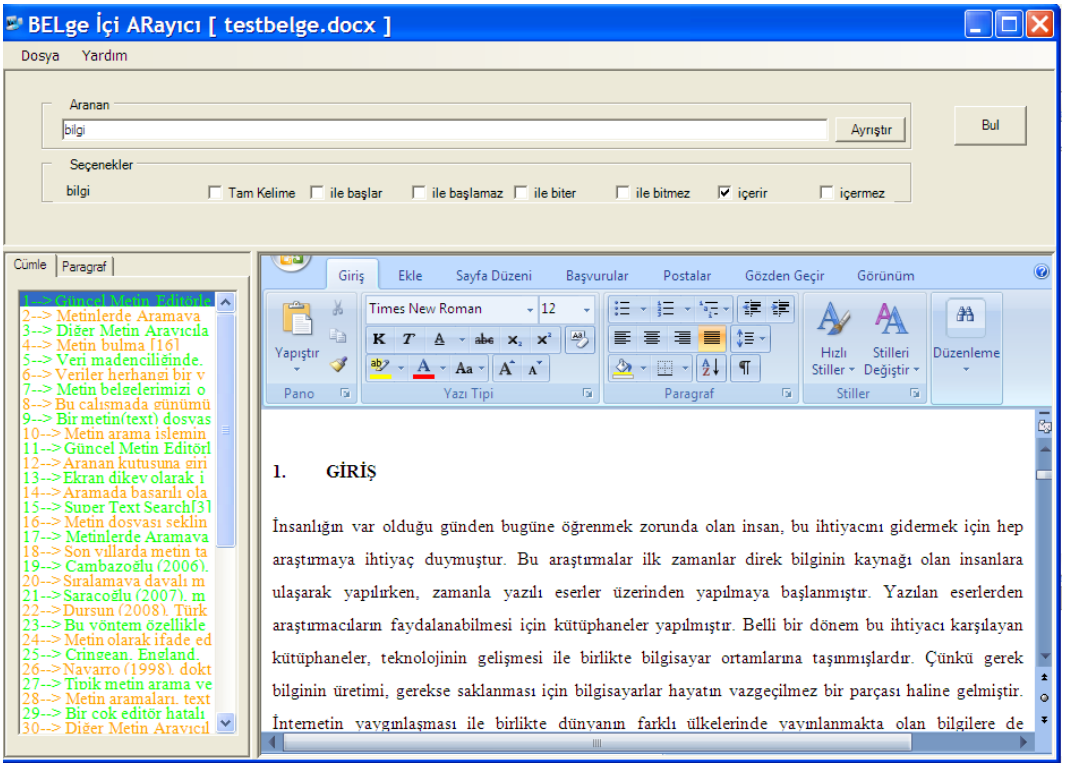
Resim 3.7. Belge Açılmış İlk Ara Yüz

BELAR ile açmış olduğumuz “testbelge.docx” isimli belge içerisinde arama yapmak istediğimiz “bilgi” kelimesini “Aranan” kutusuna yazıp “Ayrıştır” butonuna basıldığında Resim 3.8 de görülen Seçeneklerin geldiği görülecektir.



Resim 3.8 Seçenekler

Seçeneklerde görüldüğü üzere “içerir” kutusu seçili olarak gelmektedir. “Bul” butonuna basıldığında belge içerisinde “bilgi” kelimesi içeren cümleler başlarına sıra numarası verilerek listelenir (Resim 3.9).



Resim 3.9. Sonuçlar ara yüzü

Kullanıcının seçebileceği tercihlerin açıklamaları şunlardır:

Tam Kelime : Sadece tam kelimeleri bulur. Örneğin aranan “bilgi” ise “bilgisayarı” bulmaz.

İle başlar : Aranan ifade kelime başında geçiyor ise bulur. Örneğin aranan “bilgi” ise “bilgisayarı” bulur, ancak “ebilgi” yi bulmaz.

İle başlamaz : Aranan ifade ile başlamayanları bulur. Tek kelime ile aramada çok anlamlı olmayan bu seçenek, birden fazla kelime ile aramada anlam kazanır.

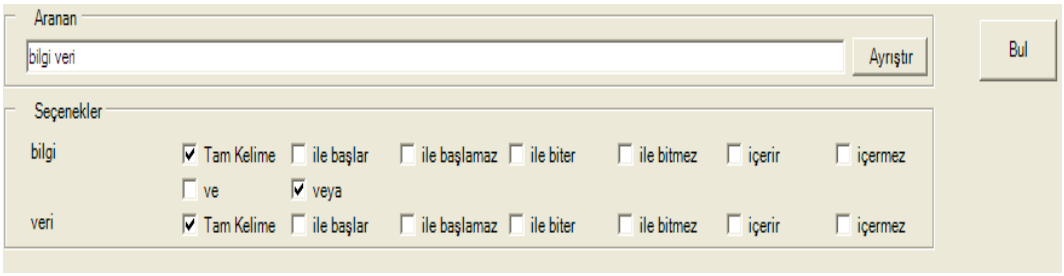
İle biter : Aranan ifade ile bitenleri bulur. Örneğin aranan “bilgi” ise “ebilgi” yi bulur.

İle bitmez : Aranan ifade ile bitmeyenleri bulur. Birden fazla kelime ile aramada anlam kazanır.

İçerir : Aranan ifade kelimenin herhangi bir yerinde geçmesi yerlidir. Örneğin aranan “bilgi” ise “ebilgi”, “bilgisayar”, “bilgi” kelimelerinin hepsini de bulur.

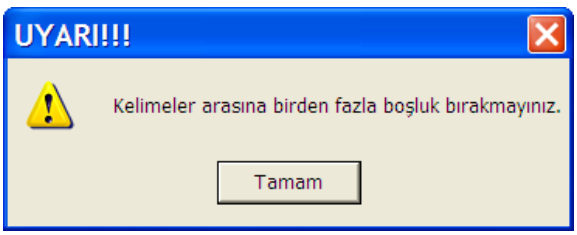
İçermez : Aranan ifadenin geçmediği cümleleri veya paragrafları bulur. Bu seçenekte birden fazla kelime ile aramada anlamlı olmaktadır.

Arama işlemi birden fazla kelime ile yapıldığında Resim 3.10’ da görüleceği üzere kelimeler alt alta ayrıştırılmakta ve kelime aralarına “ve”, “veya” bağlaçları gelmektedir. Burada bir örnek verecek olursak; bilgi ve veri kelimeleri tam kelime olarak aranmaktadır. Aradaki bağlaç “veya” olduğundan “bilgi” yada “veri” kelimelerini içeren cümleler listelenecektir. Testbelge isimli belge içerisinde bu şekilde arama yaptığımızda on iki adet cümleye ulaşmaktadır. Bağlaç “ve” yapıldığında ise cümle sayısı yedi olmaktadır. Çünkü aynı cümlede hem “veri” hem de “bilgi” kelimesinin geçmesi gerekmektedir.



Resim 3.10. Çoklu Arayış

Aranan kutusuna girilen kelimeler arasında birden fazla boşluk olmamalıdır. Kullanıcı yanlışlıkla girmiş ise program bunu kontrol edecek ve Resim 3.11’ de ki uyarı mesajını verecektir. Ayrıca aranan kutusunda yapılacak herhangi bir değişiklik sonrası mutlaka ayrıştır butonuna basılmalı ve sonrasında “bul” işlemi yapılmalıdır.



Resim 3.11. Uyarı mesajı

“ve” ile “veya” bağlaçları art arda geldiğinde işlem sırası diziliş sırasına uygun olmaktadır. Yani A “veya” B “ve” C “veya” D “ve” E ifadesi ((((A “veya” B) “ve” C) “veya” D) “ve” E) şeklinde parentezlendirilmiş olarak kabul edilmektedir. Dolayısıyla A “ve” (B “veya” C) şeklinde bir ifade B “veya” C “ve” A dizilişiyle elde edilebilmektedir.

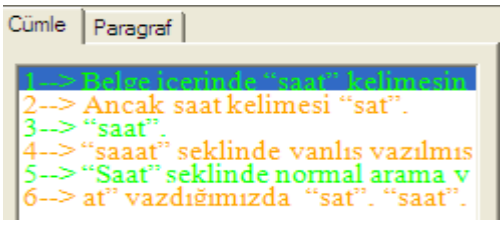
Arama işlemlerinde kriterlere uygun sonuçlar cümlede farklı, paragrafta farklı, bağlamda farklı olabilir. İki kelime aynı anda bir cümlede olmayabilir, ancak bir paragrafta olması mümkündür.

Arama işlemlerinde üzerinde durulması gereken bir özellik daha vardır ki; BELAR a ayrıcalık katmaktadır. Buda Aranan kutusuna kullanıcı düzenli ifadeleri (regular expression) kullanarak daha detaylı arama yapabilir. Bölüm 3.4’ de detaylı olarak değinilecek olan düzenli ifadelere bir örnek verecek olursak;

Belge içerinde “saat” kelimesini aramak istiyoruz. Ancak saat kelimesi “saaat” şeklinde yanlış yazılmış olabilir. “Saat” şeklinde normal arama yapıldığında sadece “saat” geçen cümleleri bulur. Oysa diğer hatalı olanı da bulmasını isteyebiliriz. Bu durumda aranan kutusuna “sa+at” yazdığımızda “saat” ve “saaat” geçenlerin hepsini de bulabilir (Resim 3.12).

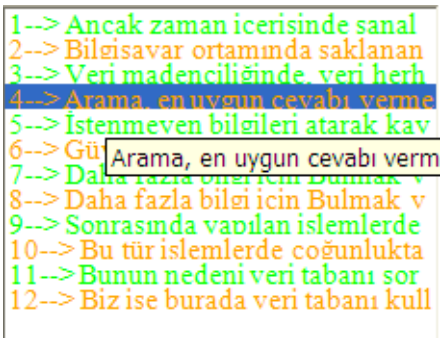
Hem düzenli ifadeler, hem de çoklu arama birlikte kullanılabildiğinden kullanıcın kabiliyetine göre çok gelişmiş aramalar yapmak mümkündür. Yine bu konuda örnekler tezimizin deneysel sonuçlar bölümünde gösterilecektir.

Burada önemli olan kullanıcının düzenli ifade ile arama yöntemini bilmesidir. BELAR ın yardım menüsünde düzenli ifade kurallarına yer verilecektir.



Resim 3.12. Düzenli İfade ile Arayış

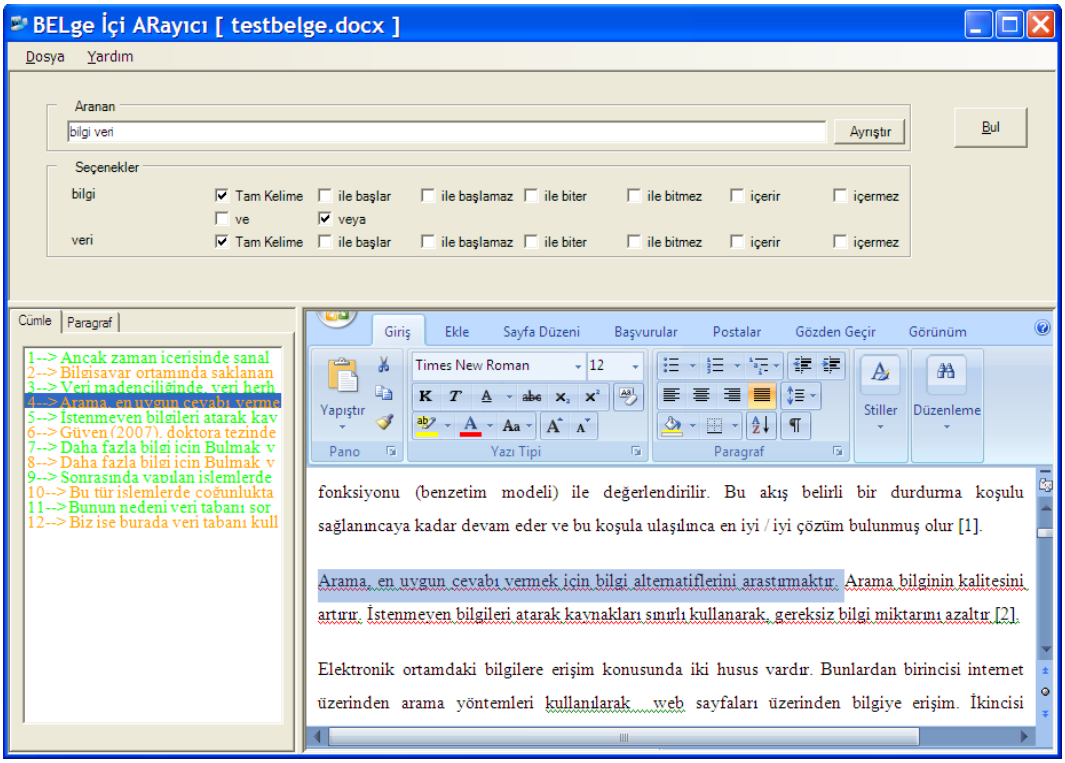
Arama sonuçlarının gösterildiği bölümde Resim 3.13'de görüleceği üzere bulunan sonuçlar sıralanmakta, cümleler arasında yön tuşları ile hareket edilebilmekte, istenilen cümle üzerine fare ile gidilerek cümle veya paragrafın tamamı okunabilmektedir.



Resim 3.13. Sonuçlar

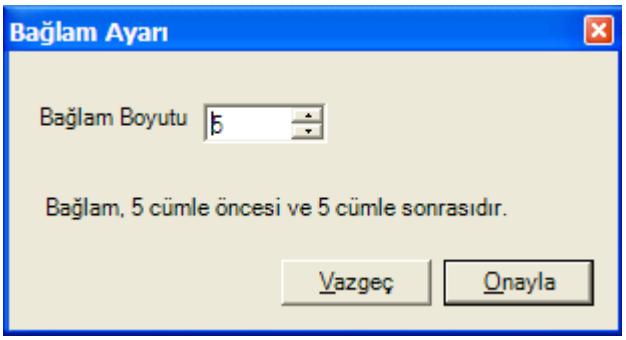
Kullanıcı sonuçlar içerisinde incelediği ve belge içerisinde ulaşmak istediği cümleye fare sol tuşu ile çift tıklayarak ilgili yere konumlanabilmektedir. Resim 3.14'de görüleceği üzere açık olan orijinal belge içerisine direk konumlanmakta ve burası seçilmiş olarak gösterilmektedir. Kullanıcı bu aşamada ilgili yeri okuyabileceği gibi, üzerinde her türlü değişikliği yapabilir, kopyalayabilir, silebilir. MS Word programının tüm özelliklerini kullanabilir.

Burada cümle için anlatılan tüm özellikler paragraf içinde geçerlidir. Kullanıcı paragrafları görmek isterse paragraf sekmesini tıklayıp, listelenen sonuçlar içerisinden istediğine fare ile çift tıklayarak belge içerisine ulaşacaktır.



Resim 3.14. Belgeye konumlanış durumu

Bağlam sekmesine basıldığında ise Bağlam ayarlarındaki değere göre arama işlemi yapılmaktadır. Bu değer varsayılan olarak 5 cümle ayarlanmıştır. Kullanıcı isterse "Ayarlar" menüsünden bu değeri değiştirebilmektedir (Resim 3.15). Arayış mantığı bir sonraki bölümde ayrıntılı olarak anlatılacaktır.



Resim 3.15. Bağlam Ayarı

Kullanıcı çalışmasını bitirdiği belgeyi dosya menüsünden kapat seçeneği ile kapatabilir. İsterse farklı belge açabilir. BELAR programından tamamen çıkmak isterse dosya menüsü çıkış komutu ile çıkabilir.

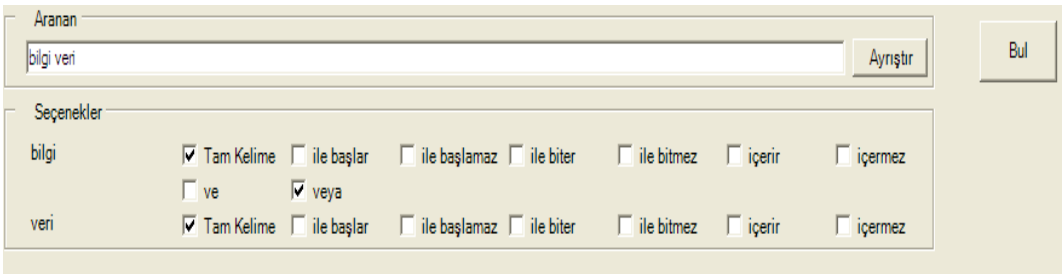
3.3. Arayış Mantığı

Bu bölümde BELAR'ın arayış mantığı program kodlarına atıflar yapılarak anlatılacaktır.

BELAR'ın arayış işlemlerini yapabileceği belge bir MS Word belgesi olmalıdır. Bu belgeyi "dosya" menüsünde "aç" komutu ile yapabilmek için gerekli kod EK-1'de verilmiştir. Programımızda *frmMain.cs* sınıfı içerisinde yer almaktadır.

Belge açıldıktan sonra "Aranan" kutusuna girilen kelime veya kelimelerin ayrıştırılması gerekmektedir. Bu ayrıştırma işlemi *Disar.cs* sınıfı içerisinde EK-2'de görülen kodlarla yapılmaktadır. Kodlardan da görüleceği üzere kelimeler boşluk karakterlerine göre ayrıştırılmakta, ve kullanıcının kelimeler arasına birden fazla boşluk vermesi durumunda ise uyarılmaktadır. Ayrıştırma işlemi "Ayrıştır" butonuna basıldığı zaman yapılmaktadır.

Kullanıcıların Resim 3.16 örneği görülen tercihlerinin belirlenebilmesi için EK-3'de görülen *tercih.cs* ve *baglac.cs* sınıfları yazılmıştır. Dinamik bir yapıya sahip olan bu tercihler aranan kutusuna girilen kelime sayısı kadar alt alta yerleştirilebilmektedir. Tercihleri yerleştirme işlemi *disar.cs* içerisinde *SecenekleriYerlestir()* metodu ile gerçekleştirilmektedir (EK-4).



Resim 3.16. Tercihlerin belirlenmesi

Tercihlerin oluşturulması aşamasında "*public string Filtre()*" metodu ile arayış işlemlerinde kullanılan Düzenli İfade *filtresi* burada belirlenmektedir.

```
if (m_TamKelime.Checked) m_Filtre = string.Format(@"(\b{0}\b)", _Kelime.Text);
```

```
else if (m_Icerir.Checked | m_Icermez.Checked) m_Filtre = string.Format(@"({0})", m_Kelime.Text);

else if (m_Baslar.Checked | m_Baslamaz.Checked) m_Filtre = string.Format(@"(\b{0})", m_Kelime.Text);

else if (m_Biter.Checked | m_Bitmez.Checked) m_Filtre = string.Format(@"({0}\s)", m_Kelime.Text);//|{0}.\s

else m_Filtre = string.Format(@"({0})", m_Kelime.Text);

return m_Filtre;
```

Bu filtreler "Bul" butonuna basıldığında "disar" sınıfı içinde yer alan *DuzenliIfadeyiBul* metoduna verilmektedir. Belge içinde cümle, paragraf ve bağlam da arayış işlemleri bu filtrelere göre yapılmaktadır.

Arayış işlemlerinde kullanılan 3 önemli metodumuz vardır. Bunlar "*Disar*" içerisinde yer alır.

- private bool DurumuTespitEt(string icerik)
- public Sonuc[] AramaSonucu(SorguTipi sorguTipi)
- private bool DuzenliIfadeyiBul(string icerik, Tercih tercih)

Kullanıcının tercihlerini oluşturmasından sonra "Bul" butonuna bastığında aşağıdaki işlemler yapılmaktadır.

Varsayılan olarak cümlede arama yapması ve listbox a listelenmesi için listBox_Cumle.DataSource = disar1.AramaSonucu(SorguTipi.Cumle) satırı ile "disar" daki Arama Sonucu metodu çelıştırılır (EK-1).

AramaSonucu() metodu içerisinde öncelikle belgenin paragrafları döngü içerisinde bir diziye yerleştirilir. Sorgu tipine göre içerikler dizisi cümlelerden de oluşabilir. Burada Paragrafı cümlelere ayrıştırma işlemide yapılmaktadır (EK-5).

DurumuTespitEt() metodu ile içeriklerde DüzenliİfadeyiBul() metodu ile önceden oluşturulan filtreler kullanılarak arayış işlemi yapılır. Arayış işlemi sorgu tipine göre cümlede, paragrafta yada bağlamda olabilir. Burada bulunan sonuçlar listeye eklenir (EK-5).

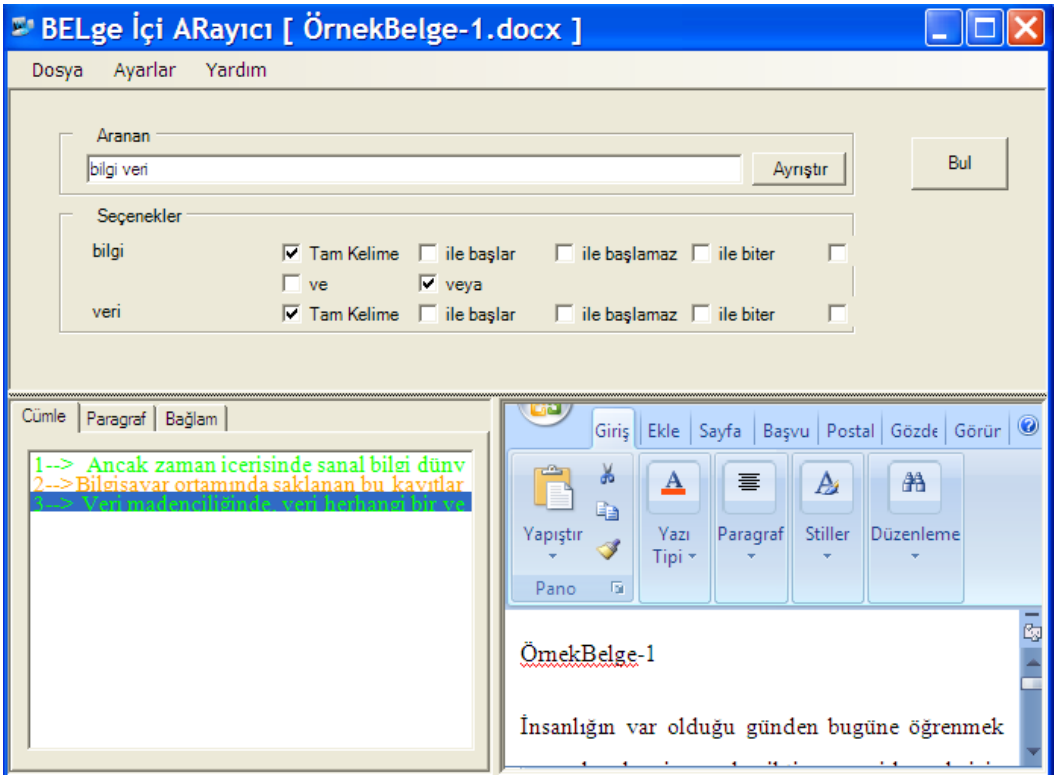
İçerik içerisinde arayış işleminden sonra bulunan sonuçlar bir listbox içerisinde kesilerek gösterilir. Liste içerisinde mouse ile üzerine gelindiğinde ise tamamı görüntülenir. Liste içinde istenilen sonuç üzerinde mouse ile çift tıklandığında *frmaMain içerisinde bulunan metdolar ile belge içinde tekrar arayış işlemi yapılır. Bu işlem doc.Application.Selection.Find.Execute* () ile yapılır. Belge içinde ilgili yerede bu şekilde konumlanılmış olur (EK-6).

Özet olarak ifade edecek olursak;

- Öncelikle belgenin tüm paragrafları elde edilir.
- Paragraf cümlelere ayrıştırılır.
- Bağlam ayarındaki sayıya göre cümleler birleştirilir.
- Ayrış işlemi içerikte düzenli ifadeler (regular expressions) kullanılarak yapılır.
- Bulunan sonuçlar listelenir.
- Liste içerisinde seçilen sonuca göre belgenin ilgili yerine konumlanılır.

4. DENEYSEL SONUÇLAR

Bu bölümde BELAR ile arayış örnekleri yapılmıştır. Arayış işlemlerinde örnek belge olarak tezimizden faydalanılmıştır. Farklı kelime ve kriterlerle aramalar sonucu bulunan sonuçlar tablo halinde sunulmuştur.



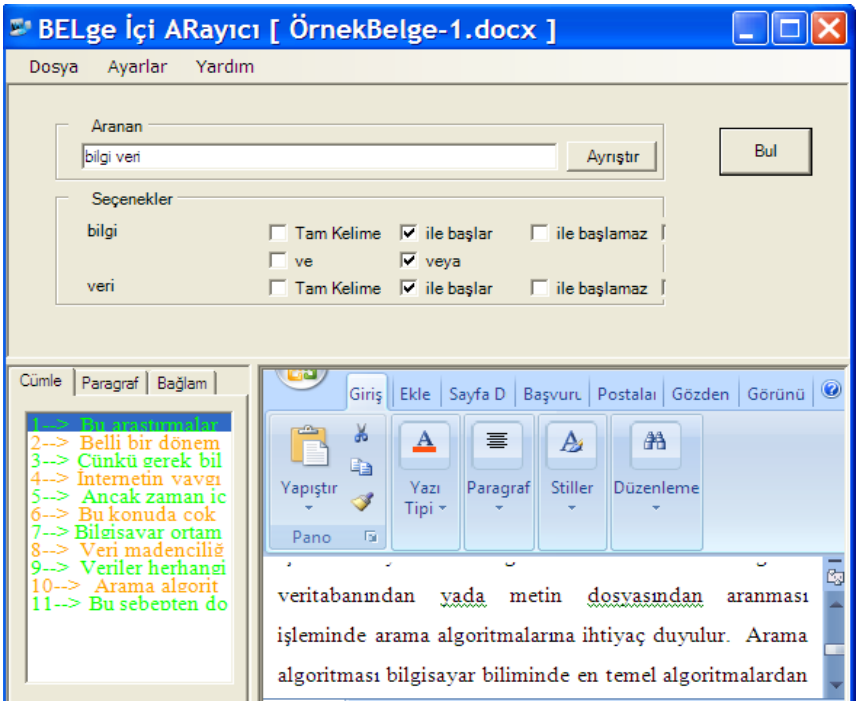
Resim 4.1. ÖrnekBelge-1 Sonuçlar Ekranı-1

Çizelge 4.1. Deneysel Örnekler-1

Örnek Belge-1 (Ek.7.)	Aranan 1. Kelime	Aranan 2. Kelime	Tercih1	Tercih2	Bağlaç	Gerçek Değerler	Programın Verdiği Sonuçlar	Doğruluk Oranı %
Cümle	bilgi	veri	Tam kelime	Tam kelime	veya	3	3	100
Paragraf	bilgi	veri	Tam kelime	Tam kelime	veya	2	2	100
Bağlam Boyut:5	bilgi	veri	Tam kelime	Tam kelime	veya	3	3	100

Çizelge 4.2. Deneysel Örnekler-2

Örnek Belge-1 (Ek.7.)	Aranan 1. Kelime	Aranan 2. Kelime	Tercih1	Tercih2	Bağlaç	Gerçek Değerler	Programın Verdiği Sonuçlar	Doğruluk Oranı %
Cümle	bilgi	veri	Tam kelime	Tam kelime	ve	0	0	100
Paragraf	bilgi	veri	Tam kelime	Tam kelime	ve	0	0	100
Bağlam Boyut:5	bilgi	veri	Tam kelime	Tam kelime	ve	2	2	100



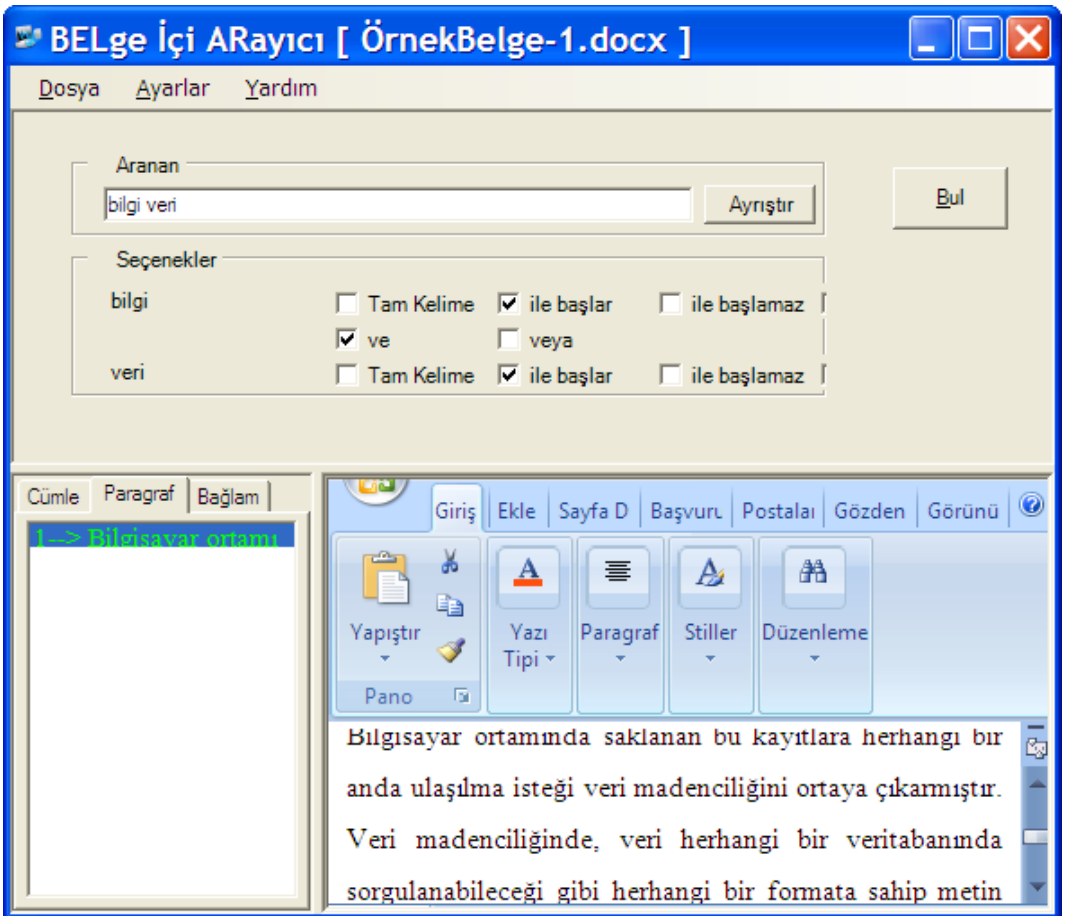
Resim 4.2. ÖrnekBelge-1 sonuçlar ekranı-2

Çizelge 4.3. Deneysel Örnekler-3

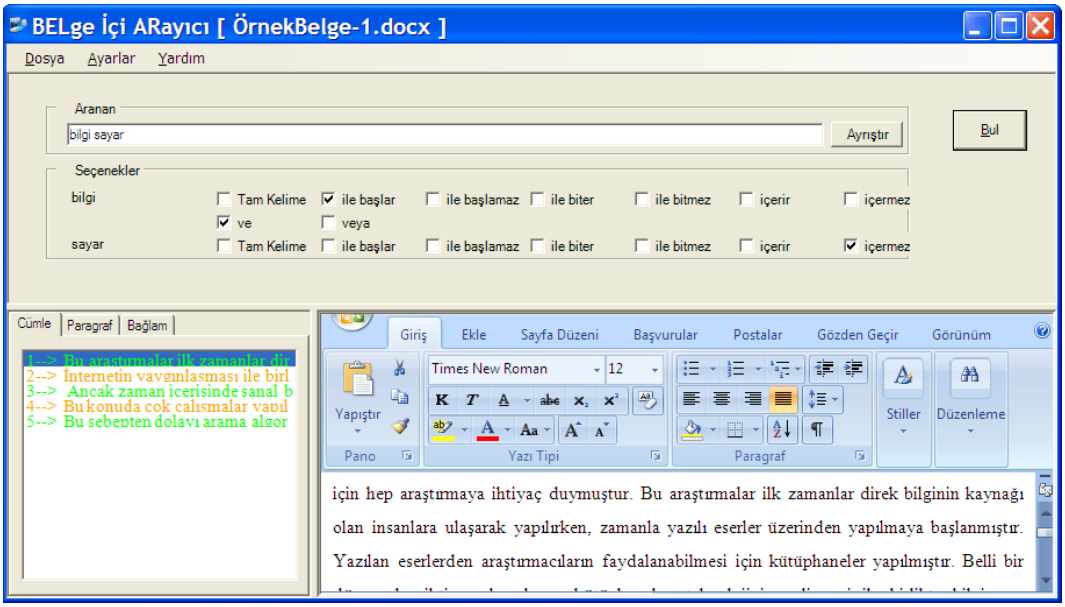
Örnek Belge-1 (Ek.7.)	Aranan 1. Kelime	Aranan 2. Kelime	Tercih1	Tercih2	Bağlaç	Gerçek Değerler	Programın Verdiği Sonuçlar	Doğruluk Oranı %
Cümle	bilgi	veri	İle başlar	İle başlar	veya	11	11	100
Paragraf	bilgi	veri	İle başlar	İle başlar	veya	2	2	100
Bağlam Boyut:5	bilgi	veri	İle başlar	İle başlar	veya	11	11	100

Çizelge 4.4. Deneysel Örnekler-4

Örnek Belge-1 (Ek.7.)	Aranan 1. Kelime	Aranan 2. Kelime	Tercih1	Tercih2	Bağlaç	Gerçek Değerler	Programın Verdiği Sonuçlar	Doğruluk Oranı %
Cümle	bilgi	veri	İle başlar	İle başlar	ve	1	1	100
Paragraf	bilgi	veri	İle başlar	İle başlar	ve	1	1	100
Bağlam Boyut:5	bilgi	veri	İle başlar	İle başlar	ve	1	1	100



Resim 4.3. ÖrnekBelge1 sonuçlar ekranı-3



Resim 4.4. ÖrnekBelge1 Sonuçlar Ekranı-4

Çizelge 4.5. Deneysel Örnekler-5

Örnek Belge-1 (Ek.7.)	Aranan 1. Kelime	Aranan 2. Kelime	Tercih1	Tercih2	Bağlaç	Gerçek Değerler	Programın Verdiği Sonuçlar	Doğruluk Oranı %
Cümle	bilgi	sayar	İle başlar	içermez	ve	5	5	100
Paragraf	bilgi	sayar	İle başlar	içermez	ve	0	0	100
Bağlam Boyut:5	bilgi	sayar	İle başlar	içermez	ve			

Düzenli ifade desenleri ile arayış örnekleri;

Çizelge 4.6. Deneysel örnekler-6

Örnek Belge-1 (Ek.7.)	Aranan 1. Kelime	Aranan 2. Kelime	Tercih1	Tercih2	Bağlaç	Gerçek Değerler	Programın Verdiği Sonuçlar	Doğruluk Oranı %
Cümle	b[ie]lge	-	içerir	-	-	2	2	100
Paragraf	b[ie]lge	-	içerir	-	-	1	1	100
Bağlam Boyut:5	b[ie]lge	-	içerir	-	-	2	2	100

Çizelge 4.7. Düzenli İfadelerle Arayış

Örnek	Bulunmak istenen kelimeler	Aranan Düzenli İfade	Tercih
1	Bilge, belge	b[ie]lge	içerir
2	Sadece sayılar	\d	içerir
3	Sonu “tır” ile bitenler	tır\b	içerir
4	] ile biten sayılar	\d]\B	içerir
5	(2005) gibi gösterimler veya yıl	(\d)\B, \d{4}	içerir
6	Belge ile başlayan satırbaşları	^Belge	içerir

5. SONUÇ VE ÖNERİLER

Günümüzde her bir alanda elektronik ortamda yayımlanmış çok sayıda, kitap, makale gibi çalışmalar mevcuttur. Araştırmalar, okuyucuların ilgi duydukları konu, başlık ve kelime gibi özel nitelik taşıyan bilgilere hızlı ve verimli şekilde ulaşabilmenin büyük önem arz etmekte olduğunu göstermektedir.

Burada yapılan çalışma sonucunda diğer metin işleyicilerinin arama işlemlerindeki noksanlıkların büyük oranda giderildiği bir uygulama ortaya konmuştur. Amaçlanmış olduğu üzere, metin belgeleri içinde kolay ve ayrıntılı arama yapabilen BELAR isimli yazılım geliştirilmiştir.

Word programını bir alt program olarak kullanan BELAR ile bilgisayar ortamlarında bulunan .doc veya .docx uzantılı belgeler içinde gelişmiş arayış işlemleri yapılabilmektedir. Kullanıcının her anahtar kelime için, “tam kelime”, “içerir”, ”içermez”, “ile başlar”, “ile başlamaz”, “biter”, “bitmez” gibi tercihleri kullanmasına ve böylece ayrıntılı arayış işlemlerini kolayca yapmasına imkan tanınmıştır. Bu tercihlerin belirlenmesi kutu işaretlenmesi suretiyle yapılmaktadır. Böylece, programın anlaşılabilirliği ve kullanımının kolaylığı artırılmıştır.

Birden fazla kelime ile arama yapıldığında, tercihler arasında “ve” ile “veya” bağlaçları kullanılarak bileşik ifadeler oluşturulmuştur. Bu bağlaçlar art arda geldiğinde işlem sırası diziliş sırasına uygun olmaktadır.

Metinler üzerinde arama işlemlerinde önemli bir yeri olan düzenli ifade (regular expression) desenlerinin BELAR’da kolaylıkla kullanılabilir olması sağlanmıştır. Düzenli ifade anahtarları ile arama seçenekleri daha da gelişmiş bir yapıya kavuşmuştur. Birden fazla düzenli ifade desenleri arasında mantıksal bağlaçlarında kullanılabilir olması, benzer programlara göre bir ayrıcalıktır.

Birden fazla anahtar kelime kullanımı söz konusu olduğu için arama işlemlerinin gerçekleştirileceği metin parçaları üç farklı ölçü kullanılarak yapılmıştır. Bunlar kullanıcının tercihine bırakılan cümle, paragraf ve bağlamdır. Mevcut metin işleyicilerinde bu tarz bir yaklaşıma rastlanmamaktadır.

Belgede yapılan aramalar neticesinde elde edilen sonuçlar listelenmektedir. Cümle, paragraf veya bağlam olarak istenen şekilde görülebilen bu sonuçların belge içersindeki yerine hızlıca konumlanılabilmektedir.

File search assistant, Arcivarius, Super text search gibi bu tarz programların bir çoğunda orijinal belge içinde bulunan sonuçlar başka bir editörde gösterildiğinden kullanım zorluğu yaşanmakta, BELAR' da ise orijinal belge üzerinde çalışıldığından Word programının kendi içinde bulunan, silme, değiştirme, renklendirme v.s. birçok özelliği kullanılabilmektedir.

Programın Microsoft Word ile entegre olması, Türkçe desteği, kullanım kolaylığı, geliştirilebilir olması öne çıkan önemli ayrıcalıklarıdır. Bir önemli özelliği de BELAR'ın kullanımı için herhangi bir ekstra lisanslı yazılıma ihtiyaç olmamasıdır. Veri tabanı kullanılmadığından sadece Microsoft Word ün olması yeterlidir.

Bu çalışmada hedeflenen gelişmiş arayış işlemleri BELAR ile yapılan bütün deneylerde tam başarı elde edilmiştir. Mevcut metin işleyicilerinin yetersizliklerine dair hususlar büyük oranda düzeltilmiştir. Amaç arama işlemlerini ayrıntılandırmak olduğundan, belge içindeki verilerin özelliklerine göre büyük boyutlu belgelerde hız ve performans düşüklüğü görülebilmektedir

BELAR'ın gelişmiş arama seçenekleri ve diğer özellikleri sayesinde araştırmacılara faydalı olacağı düşünülmektedir.

KAYNAKLAR

1. Güden, H., Vakvak, B., Özkan, B. E., Altıparmak F., Dengiz, B., "Genel Amaçlı Arama Algoritmaları ile Benzetim En İyilemesi**",** ***Endüstri Mühendisliği Dergisi,*** 16 (1): 2-15 (2003).
2. Benzer, A. İ., "Yapay Zeka Uygulamalarında Kullanılan Arama Algortimalarının Kıyaslanması ", Yüksek Lisans Tezi, ***Gazi Üniversitesi Bilişim Enstitüsü***, Ankara, 1 (2007).
3. Cambazoğlu, B. B., "Models and Algorithms for Parallel Text Retrieval", Doktora Tezi, ***Bilkent Üniversitesi, Mühendislik ve Fen Bilimleri Enstitüsü,*** Ankara , 1-2 (2006)
4. Güldal, H., "E-posta Metinleri Üzerinde İçerik Taraması Yapan Bir Uygulama Geliştirilmesi", Yüksek Lisans Tezi, ***Trakya Üniversitesi Fen Bilimleri Enstitüsü,*** Edirne, 1 (2005).
5. Güven, A., "Türkçe Belgelerin Anlam Tabanlı Yöntemlerle Madenciliği", Doktora Tezi, ***Yıldız teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 1 (2007).
6. Saraçoğlu, R., "Bulanık Kümeleme Kullanılarak Benzer Belge Aranması", Doktora Tezi, ***Selçuk Üniversitesi Fen Bilimleri Enstitüsü***, 1-3 (2007).
7. Dursun, B., "Türkçe Metinlerin Benzerliğinin Hesaplanması İçin Yeni Bir Yöntem", Yüksek Lisans Tezi, ***Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 1 (2008).
8. Cringean, J. K. England, R., Manson, G. A., Willet, P., "Parallel text searching in serial files using a processor farm", ***Annual ACM Conference on Research and Development in Information Retrieval,*** Belgium, 429-453 (1989).
9. Navarro, G., "Approximate Text Searching", PhD thesis, ***Dept. of Computer Science University of Chile***, Santiago–Chile, 1-4(1998).
10. Sun Wu, S., Manber, U., "Fast Text Searching with Errors", M.Sc. Thesis, ***Department of Computer Science University of Arizona***, Tucson, 1 (1991).
11. Alpdoğan, Y., "Kendinden Düzenlenen Haritalar ile Döküman Sınıflandırma", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 1-4 (2007).
12. Türkeeş, M.K., "Bilgi Erişiminde Tamlama Temelli Dizinleme", Yüksek Lisans Tezi, ***İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 1-2 (2007).
13. Okur, H., "Dizinleme Amacı ile Kullanılabilecek Yöntemlerin Kıyaslanması ve Arama sistemi Geliştirilmesi", Yüksek Lisans Tezi, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 1-3 (2007).
14. İnternet: Gazi Üniversitesi "Hesap Teorileri" http://w3.gazi.edu.tr/~dcalik/=Duyuru=/Duyuru=DERSLER/=HesapTeorileri/%202009iyhesap.pdf (2009).

15. İnternet: Trakya Üniversitesi "Düzenli İfadeler Sunumu" http://yilmazkilicaslan.trakya.edu.tr/teaching.html (2009).
16. İnternet: Vikipedi "Özgür Ansiklopedi" http://tr.wikipedia.org/wiki/Noam_Chomsky (2009).
17. Algan, S.,"Her yönüyle C#",***Pusula Yayıncılık,*** İstanbul, 24-25, 426-427 (2008).
18. Govyaerts, J., Levithan, S., "Regular Expressions Cookbook", **O'Reilly Media**, USA, 1-5 (2009).
19. Friedl, J. E. F., "Mastering Regular Expressions", **O'Reilly Media**, USA, 5 (2009).
20. Stublebine, T., "Regular Expressions for Perl, C, PHP, Python, Java and .NET", **O'Reilly Media**,USA, 20 (2009).
21. Forta, B., "Sams teach yourself regular expressions in 10 minutes", **Sams Publishing**, USA, 19-27 (2004)
22. Habibi, M., "Java Regular Expressions", **Sams Publishing**, USA, 173-213 (2004).
23. Gennick, J., Linsley, P., "Oracle Regular Expressions", **O'Reilly Media**, USA, 173-213 (2003).
24. Good, N. A., "Regular Expression Recipes for Windows Developers: A Problem-Solution Approach (Kindle Edition)", **Apress,** NewYork, 1 -31 (2004).
25. Schildt, H., "C# 2.0 ", **Alfa Yayıncılık**, İstanbul, 9 (2006).

EKLER

EK-1. Belge Aç C# kodu

```
private void açToolStripMenuItem_Click(object sender, EventArgs e)
      {
         string filNm;
         openFileDialog1.Multiselect = false;
         openFileDialog1.Filter = "MS Word Belgeleri (*.docx,*.doc,*.dot) |
*.docx;*.doc;*.dot";
         if (openFileDialog1.ShowDialog() == DialogResult.OK)
         {
            filNm = openFileDialog1.FileName;
            this.Text = string.Format("BELge İçi ARayıcı [ {0} ]", new
System.IO.FileInfo(filNm).Name);
            listBox_Cumle.DataSource = null;
         }
         else return;

         try
         {
            objWinWordControl.CloseControl();

         }
         catch { }
         finally
         {
            objWinWordControl.document = null;
            WinWordControl.WinWordControl.wd = null;
            WinWordControl.WinWordControl.wordWnd = 0;
         }
         try
         {

            //Load the template used for testing.
            objWinWordControl.LoadDocument(filNm);
         }
         catch (Exception ex) { String err = ex.Message; }
      }
```

EK-2. Ayrıştır işlemleri

```
private void button_Ayristir_Click(object sender, EventArgs e)
      {
            if (groupBox_Secenekler != null) groupBox_Secenekler.Controls.Clear();
            if (m_Tercihler != null) m_Tercihler.Clear();
            if (m_Baglaclar != null) m_Baglaclar.Clear();

            string aranan = m_Aranan.Text.Trim();
            if (!aranan.Equals(""))
            {
               string[] kelimeler = aranan.Split(' ');

               bool hataVarMi = false;
               foreach (string kelime in kelimeler)
               {
                  if (kelime.Equals("")) hataVarMi = true;
               }

               try
               {
                  if (hataVarMi) throw new Exception("Kelimeler arasına birden fazla
boşluk bırakmayınız.");
          SecenekleriYerlestir(groupBox_Secenekler, kelimeler);
               }
               catch (Exception ex)
               {
                  MessageBox.Show(ex.Message, "UYARI!!!",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
               }
               m_KontrolButonu.Enabled = true;
            }


            }
```

EK-3. Tercih ve Baglac Sınıfları

```
public partial class Tercih : UserControl
   {

      private string m_Filtre;
      private int m_Oncelik;

      public int Oncelik
      {
         get { return m_Oncelik; }
         set { m_Oncelik = value; }
      }

 public string Filtre
 {
       get
         {
            if (m_TamKelime.Checked) m_Filtre = string.Format(@"(\b{0}\b)",
m_Kelime.Text);
            else if (m_Icerir.Checked | m_Icermez.Checked) m_Filtre =
string.Format(@"({0})", m_Kelime.Text);
            else if (m_Baslar.Checked | m_Baslamaz.Checked) m_Filtre =
string.Format(@"(\b{0})", m_Kelime.Text);
            else if (m_Biter.Checked | m_Bitmez.Checked) m_Filtre =
string.Format(@"({0}\s)", m_Kelime.Text);//|{0}.\s
            else m_Filtre = string.Format(@"({0})", m_Kelime.Text);
            return m_Filtre;
         }
         set { m_Filtre = value; }
      }
      public Tercih()
      {
         InitializeComponent();
      }

private void m_Baslar_CheckedChanged(object sender, EventArgs e)
      {
         if (m_Baslar.Checked)
         {
            m_TamKelime.Checked=m_Baslamaz.Checked = m_Biter.Checked =
m_Bitmez.Checked = m_Icerir.Checked = m_Icermez.Checked = false;
         }
      }
```

EK-3. (Devam) Tercih ve Baglac Sınıfları

```
 private void m_Baslamaz_CheckedChanged(object sender, EventArgs e)
      {
         if (m_Baslamaz.Checked)
         {
            m_TamKelime.Checked = m_Biter.Checked = m_Bitmez.Checked =
m_Icerir.Checked = m_Icermez.Checked = m_Baslar.Checked = false;      }        }

private void m_Biter_CheckedChanged(object sender, EventArgs e)
      {
         if (m_Biter.Checked)
         {
            m_TamKelime.Checked = m_Bitmez.Checked = m_Icerir.Checked =
m_Icermez.Checked = m_Baslar.Checked = m_Baslamaz.Checked = false;
         }

      }

private void m_Bitmez_CheckedChanged(object sender, EventArgs e)
      {
         if (m_Bitmez.Checked)
         {
            m_TamKelime.Checked = m_Icerir.Checked = m_Icermez.Checked =
m_Baslar.Checked = m_Baslamaz.Checked = m_Biter.Checked = false;
         }

      }

 private void m_Icerir_CheckedChanged(object sender, EventArgs e)
      {
         if (m_Icerir.Checked)
         {
            m_TamKelime.Checked = m_Icermez.Checked = m_Baslar.Checked =
m_Baslamaz.Checked = m_Biter.Checked = m_Bitmez.Checked = false;
         }

      }

private void m_Icermez_CheckedChanged(object sender, EventArgs e)
      {
         if (m_Icermez.Checked)
         {
            m_TamKelime.Checked = m_Baslar.Checked = m_Baslamaz.Checked =
m_Biter.Checked = m_Bitmez.Checked = m_Icerir.Checked = false;
         }
      }
```

EK-3. (Devam) Tercih ve Baglac Sınıfları

```
 private void m_TamKelime_CheckedChanged(object sender, EventArgs e)
      {
         if (m_TamKelime.Checked)
         {
           m_Icermez.Checked = m_Baslar.Checked = m_Baslamaz.Checked =
m_Biter.Checked = m_Bitmez.Checked = m_Icerir.Checked = false;
         }
      }   }

public partial class Baglac : UserControl
   {
      public Baglac()
      {
         InitializeComponent();
      }

      public Baglac Kopyasi()
      {
         Baglac b = new Baglac();
         b.m_Ve = new CheckBox();
         b.m_Ve.Checked = this.m_Ve.Checked;
         b.m_Veya = new CheckBox();
         b.m_Veya.Checked = this.m_Veya.Checked;
         return b;
      }

 private void m_Veya_CheckedChanged(object sender, EventArgs e)
      {
         m_Ve.Checked = !m_Veya.Checked;
      }

 private void m_Ve_CheckedChanged(object sender, EventArgs e)
      {
         m_Veya.Checked = !m_Ve.Checked;
      }
```

EK-4. Tercihlerin Yerleştirilmesi

```
private void SecenekleriYerlestir(GroupBox tasiyici, string[] kelimeler)
      {

         if ((kelimeler != null) & (kelimeler.Length > 0))
         {
            m_Baglaclar = new BaglacKoleksiyonu();
            m_Tercihler = new TercihKoleksiyonu();
            for (int i = 0; i < kelimeler.Length; i++)
            {
               //Tercihler
               Tercih t = new Tercih();
               t.Kelime.Text = kelimeler[i];
               t.Location = new Point(10, i * 40 + 20);
               m_Tercihler.Ekle(t);
               tasiyici.Controls.Add(t);


               //Bağlaçlar
               if (i != kelimeler.Length - 1)
               {
                  Baglac b = new Baglac();
                  b.Location = new Point(10, i * 40 + 40);
                  m_Baglaclar.Ekle(b);
                  tasiyici.Controls.Add(b);
               }
            }
         }

               }
```

EK-5. Arayış İşlemleri

```
private bool DurumuTespitEt(string icerik)
      {
         bool durum = false;
         if (m_Tercihler.Count == 1)
         {
            if (DuzenliIfadeyiBul(icerik, m_Tercihler[0]))
            {
               durum = true;
            }

         }
         else
         {
            durum = DuzenliIfadeyiBul(icerik, m_Tercihler[0]);
            for (int i = 1; i < m_Tercihler.Count; i++)
            {
               if (i < m_Tercihler.Count - 1)
               {
                  if (m_Baglaclar[i - 1].Ve.Checked)
                  {
                     durum &= DuzenliIfadeyiBul(icerik, m_Tercihler[i]);

                  }
                  else
                  {
                     durum |= DuzenliIfadeyiBul(icerik, m_Tercihler[i]);
                  }
               }
               else
               {
                  if (m_Baglaclar[i - 1].Ve.Checked) durum &=
DuzenliIfadeyiBul(icerik, m_Tercihler[i]);
                  else durum |= DuzenliIfadeyiBul(icerik, m_Tercihler[i]);
               }
            }
         }
         return durum;
      }
```

EK-5. (Devam) Arayış İşlemleri

```
public Sonuc[] AramaSonucu(SorguTipi sorguTipi)
      {
         ArrayList liste = new ArrayList();
         Sonuc.Miktar = 0;

         Word.Paragraphs paragraflar =
m_Belge.Application.ActiveDocument.Paragraphs;

         int sayac = 0;
         char[] cumleAyraci = new char[] { '.', ':', '!', '?' };
         bool[] veBaglaclari = new bool[m_Baglaclar.Count];
         bool[] veyaBaglaclari = new bool[m_Baglaclar.Count];
int[] cumleSayilari = new int[paragraflar.Count];

         for (int pc = 1; pc <= paragraflar.Count; pc++) cumleSayilari[pc - 1] =
paragraflar.Item(pc).Range.Text.Split(cumleAyraci).Length;

            for (int i = 1; i <= paragraflar.Count; i++)
            {
               string[] icerikler = null;
               if (sorguTipi == SorguTipi.Paragraf)
               {
                  icerikler = new string[1];
                  icerikler[0] = paragraflar.Item(i).Range.Text;
               }
               else if (sorguTipi == SorguTipi.Cumle)
               {
                  icerikler = paragraflar.Item(i).Range.Text.Split(cumleAyraci);
               }
               else if (sorguTipi == SorguTipi.Baglam)
               {

                  icerikler = paragraflar.Item(i).Range.Text.Split(cumleAyraci);

                  for (int k = 0; k < m_Baglaclar.Count; k++)
                  {
                     veBaglaclari[k] = m_Baglaclar[k].Ve.Checked;
                     veyaBaglaclari[k] = m_Baglaclar[k].Veya.Checked;
                     m_Baglaclar[k].Ve.Checked = false;
                     m_Baglaclar[k].Veya.Checked = true;
                  }
               }
```

EK-5. (Devam) Arayış İşlemleri

```
for (int t = 0; t < icerikler.Length; t++)
        {
          if (DurumuTespitEt(icerikler[t]))
          {
             if (sorguTipi != SorguTipi.Baglam)
             {
                Sonuc sonuc = new Sonuc();
                sonuc.Icerik = icerikler[t];
                liste.Add(sonuc);
             }
             else
             {
            //Bulunan cümlenin baglam miktarı kadar öncesi ve sonrasını elde et
          StringBuilder baglam = new StringBuilder();

               int s = 0;
               int zc = t;
               int zp = i;
               string kisaBilgi = icerikler[t];
               while (s != m_BaglamBoyutu)
               {
                  while (zc != 0)
                  {
                     zc--;
                     s++;
                     if (s == m_BaglamBoyutu) break;
                     baglam.Insert(0,
paragraflar.Item(zp).Range.Text.Split(cumleAyraci)[zc]);
                     //baglam.AppendLine();
                  }
                  if (s == m_BaglamBoyutu) break;
                  zp--;
                  if (zp == 0) break;
                  zc = cumleSayilari[zp-1];
               }

               s = 0;
               zc = t;
               zp = i;

               while (s != m_BaglamBoyutu)
               {
                  string[]
dizi=paragraflar.Item(zp).Range.Text.Split(cumleAyraci);
                  while (zc != cumleSayilari[zp-1])
```

EK-5. (Devam) Arayış İşlemleri

```
                    {
                        if (s == m_BaglamBoyutu) break;
                        baglam.Append(dizi[zc]);
                        zc++;
                        s++;
                    }
                    if (s == m_BaglamBoyutu) break;
                    zp++;
                    zc = 0;
                    if (zp == paragraflar.Count) break;
                }



                //Hepsi Veya lanmış bağlaçlar eski haline getiriliyor
                for (int k = 0; k < m_Baglaclar.Count; k++)
                {
m_Baglaclar[k].Ve.Checked=veBaglaclari[k];
        m_Baglaclar[k].Veya.Checked = veyaBaglaclari[k];
                }

  //DurumTespitini bağlama göre tekrar çalıştır
                if (DurumuTespitEt(baglam.ToString()))
                {
  //Sonuç nesnesi oluştur ve listeye ekle
                    Sonuc sonuc = new Sonuc();
                    sonuc.Icerik = baglam.ToString();
                    sonuc.KisaBilgi = kisaBilgi;
                    liste.Add(sonuc);
                }
              }
            }
            sayac++;
          }

          if (sorguTipi == SorguTipi.Baglam)
          {
 //Hepsi Veya lanmış bağlaçlar eski haline getiriliyor
              for (int k = 0; k < m_Baglaclar.Count; k++)
              {
                  m_Baglaclar[k].Ve.Checked = veBaglaclari[k];
                  m_Baglaclar[k].Veya.Checked = veyaBaglaclari[k];
              }          }          }
```

EK-5. (Devam) Arayış İşlemleri

```
            Sonuc[] sonuclar = null;
            if ((liste != null) & (liste.Count > 0))
            {
               sonuclar = new Sonuc[liste.Count];
               for (int i = 0; i < sonuclar.Length; i++)
               {
                  sonuclar[i] = (Sonuc)liste[i];
               }
            }
            return sonuclar;
         }


         private bool DuzenliIfadeyiBul(string icerik, Tercih tercih)
         {
            bool sonuc = false;
            try

{
               MatchCollection mc = Regex.Matches(icerik, tercih.Filtre,
RegexOptions.IgnoreCase);
               if (mc.Count != 0)
               {
                  sonuc = true;
               }
               if (tercih.Icermez.Checked | tercih.Baslamaz.Checked |
tercih.Bitmez.Checked)
               {
                  sonuc = !sonuc;
               }
            }
            catch (Exception)
            {

//MessageBox.Show(ex.Message,"HATA!!!",MessageBoxButtons.OK,MessageBoxI
con.Error);
            }

            return sonuc;


         }
```

EK-5. (Devam) Arayış İşlemleri

```
private bool DuzenliIfadeyiBul(string icerik, Tercih tercih)
      {
         bool sonuc = false;
         try
         {
            MatchCollection mc = Regex.Matches(icerik, tercih.Filtre,
RegexOptions.IgnoreCase);
            if (mc.Count != 0)
            {
               sonuc = true;
            }
            if (tercih.Icermez.Checked | tercih.Baslamaz.Checked |
tercih.Bitmez.Checked)
            {
               sonuc = !sonuc;
            }
         }
         catch (Exception)
         {

//MessageBox.Show(ex.Message,"HATA!!!",MessageBoxButtons.OK,MessageBoxI
con.Error);
         }
         return sonuc;


      }
```

EK-6. Sonuçlara Göre Belgede Arayış

“frmMain.cs”

```
private void listBox_Cumle_MouseMove(object sender,MouseEventArgs e)


      {
         int indeks=listBox_Cumle.IndexFromPoint(e.X, e.Y);
         if ((listBox_Cumle.Items.Count > 0)&(indeks>=0))
         {
            Sonuc snc = (Sonuc)listBox_Cumle.Items[indeks];
            toolTip1.SetToolTip(listBox_Cumle, snc.Icerik);
         }
      }

private void listBox_Cumle_MouseDoubleClick(object sender, MouseEventArgs e)
      {
         if ((listBox_Cumle.Items.Count > 0) & (listBox_Cumle.SelectedIndex >= 0))
         {
            Sonuc snc = (Sonuc)listBox_Cumle.SelectedItem;
            object oCumle =
snc.Icerik.Length>120?snc.Icerik.Substring(0,120):snc.Icerik;
            object missing = System.Reflection.Missing.Value;
            Word.Document doc = disar1.Belge;
            doc.Application.Selection.SetRange(0, doc.Characters.Count);
            try
            {
  if (!doc.Application.Selection.Find.Execute(ref oCumle,
     ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref
missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,
ref missing))
               {
                  throw new Exception("Özel Word nesnelerinde arama yapılamaz.");
               }
            }
            catch (Exception ex)
            {

MessageBox.Show(ex.Message,"HATA!!!",MessageBoxButtons.OK,MessageBoxIc
on.Error);
            }
         }
      }
```

EK-6. (Devam) Sonuçlara Göre Belgede Arayı

```
 private void listBox_Paragraf_MouseDoubleClick(object sender, MouseEventArgs
e)


      {
    if ((listBox_Paragraf.Items.Count > 0) & (listBox_Paragraf.SelectedIndex >= 0))
        {
 Sonuc snc = (Sonuc)listBox_Paragraf.SelectedItem;
object oCumle = snc.Icerik.Length > 120 ? snc.Icerik.Substring(0, 120) : snc.Icerik;
            object missing = System.Reflection.Missing.Value;
            Word.Document doc = disar1.Belge;
            doc.Application.Selection.SetRange(0, doc.Characters.Count);
            try
            {
       if (!doc.Application.Selection.Find.Execute(ref oCumle,
         ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref
missing,ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,
ref missing))
               {
                  throw new Exception("Özel Word nesnelerinde arama yapılamaz.");
               }
            }
            catch (Exception ex)
            {
               MessageBox.Show(ex.Message, "HATA!!!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }       }       }

 private void frmMain_Load(object sender, EventArgs e)
      {
         disar1.KontrolButonu = button1;
      }
private bool yeniParagrafSorgusuMu;
private bool yeniBaglamSorgusu;
private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
      {
         this.Cursor = Cursors.WaitCursor;
         toolTip1.ToolTipTitle = "Cümle Sorgusu Sonuçları";
 if ((tabControl1.SelectedIndex == 1)&(yeniParagrafSorgusuMu))
         {
         toolTip1.ToolTipTitle = "Paragraf Sorgusu Sonuçları";
   listBox_Paragraf.DataSource = disar1.AramaSonucu(SorguTipi.Paragraf);
            yeniParagrafSorgusuMu = false;  }
```

EK-6. (Devam) Sonuçlara Göre Belgede Arayış

```
 if ((tabControl1.SelectedIndex == 2)&(yeniBaglamSorgusu))
          {
             toolTip1.ToolTipTitle = "Bağlam Sorgusu Sonuçları";
             disar1.BaglamBoyutu = Ayar.DegerOku();
             listBox_Baglam.DataSource = disar1.AramaSonucu(SorguTipi.Baglam);
             yeniBaglamSorgusu = false;
          }
          this.Cursor = Cursors.Default;
       }
private void listBox_Baglam_MouseMove(object sender, MouseEventArgs e)
       {
          int indeks = listBox_Baglam.IndexFromPoint(e.X, e.Y);
          if ((listBox_Baglam.Items.Count > 0) & (indeks >= 0))
          {
             Sonuc snc = (Sonuc)listBox_Baglam.Items[indeks];
             toolTip1.SetToolTip(listBox_Baglam, snc.Icerik);
          }
       }

private void listBox_Baglam_MouseDoubleClick(object sender, MouseEventArgs e)
       {
          if ((listBox_Baglam.Items.Count > 0) & (listBox_Baglam.SelectedIndex >=
0))
          {
             Sonuc snc = (Sonuc)listBox_Baglam.SelectedItem;
             object oCumle = snc.KisaBilgi.Length > 50 ? snc.KisaBilgi.Substring(0,
50) : snc.KisaBilgi;
             object missing = System.Reflection.Missing.Value;
             Word.Document doc = disar1.Belge;
             doc.Application.Selection.SetRange(0, doc.Characters.Count);
             try
             {
 if (!doc.Application.Selection.Find.Execute(ref oCumle,
    ref missing, ref missing, ref missing, ref missing, ref missing, ref missing, ref
missing,ref missing, ref missing, ref missing, ref missing, ref missing, ref missing,
ref missing))
                {
     throw new Exception("Özel Word nesnelerinde arama yapılamaz.");
                }
             }
             catch (Exception ex)
             {
                MessageBox.Show(ex.Message, "HATA!!!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
             }          }       }
```

EK-7. ÖrnekBelge-1

İnsanlığın var olduğu günden bugüne öğrenmek zorunda olan insan, bu ihtiyacını gidermek için hep belge araştırmaya ihtiyaç duymuştur. Bu araştırmalar ilk zamanlar direk bilginin kaynağı olan bilgelere ulaşarak yapılırken, zamanla yazılı eserler üzerinden yapılmaya başlanmıştır. Yazılan eserlerden araştırmacıların faydalanabilmesi için kütüphaneler yapılmıştır. Belli bir dönem bu ihtiyacı karşılayan kütüphaneler, teknolojinin gelişmesi ile birlikte bilgisayar ortamlarına taşınmışlardır. Çünkü gerek bilginin üretimi, gerekse saklanması için bilgisayarlar hayatın vazgeçilmez bir parçası haline gelmiştir. İnternetin yaygınlaşması ile birlikte dünyanın farklı ülkelerinde yayınlanmakta olan bilgilere de ulaşılabilmektedir. Ancak zaman içerisinde sanal bilgi dünyasında artan kaynakların yoğunluğu, yapılan araştırmalarda istenen sonuçlara hızlı ulaşabilme problemini ortaya çıkarmıştır. Bu konuda çok çalışmalar yapılmış, bilgiye hızlı erişim için yöntemler geliştirilmiştir.

Bilgisayar ortamında saklanan bu kayıtlara herhangi bir anda ulaşılma isteği veri madenciliğini ortaya çıkarmıştır. Veri madenciliğinde, veri herhangi bir veritabanında sorgulanabileceği gibi herhangi bir formata sahip metin içerikli dosyada da sorgulanabilir. Veriler herhangi bir veritabanından yada metin dosyasından aranması işleminde arama algoritmalarına ihtiyaç duyulur. Arama algoritması bilgisayar biliminde en temel algoritmalardan biridir. Arama algoritması, bir probleme, bir çözüm uzayı içerisinden en uygun çözümü arayan algoritmadır. Arama işlemine sürekli ihtiyaç duyulduğu için aramanın hızlı olması da oldukça önemlidir. Bu sebepten dolayı arama algoritmaları elde olan bilgiyi en iyi şekilde kullanarak sonuca nasıl daha hızlı ulaşılacağı üzerinde yoğunlaşır.

Arama algoritmaları, algoritmanın yapısına uygun olarak seçilen bir başlangıç çözümünden ya da çözümlerinden başlar ve farklı yapılar kullanarak yeni çözümlere ulaşırlar. Bu çözümler amaç fonksiyonu (benzetim modeli) ile değerlendirilir. Bu akış belirli bir durdurma koşulu sağlanıncaya kadar devam eder ve bu koşula ulaşılınca en iyi / iyi çözüm bulunmuş olur [1].

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: DEMİRSÖZ, Orhan
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 20.12.1972 Ankara
Medeni hali	: Evli
Telefon	: 0 (312) 385 81 00
Faks	: 0 (312) 385 19 98
e-mail	: orhan@demirsoz.net

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Eğt.	2009
Lisans	Gazi Üniversitesi/ Bilgisayar Eğt.	1995
Lise	Gazi Teknik Lisesi	1991

İş Deneyimi

Yıl	Yer	Görev
1995-1998	Taşkent Bilgisayar Lisesi	Bilgisayar Öğretmeni
1998-1999	Kızılırmak Lisesi	Bilgisayar Öğretmeni
1999-2000	Hava K.K. lığı	Yedek Subay
2000-2004	Emre Bilgisayar A.Ş.	Teknik Des. ve Eğt. Müd.
2004-	Fatih Üniversitesi	Öğretim Görevlisi

Yabancı Dil

İngilizce

Yayınlar

Hobiler

Masa Tenisi, Bilgisayar teknolojileri, Kitap okumak