İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA SORGULARI ÜZERİNDE GÖREV TABANLI KÜMELEME

YÜKSEK LİSANS TEZİ

Almıla Selcen AKGÜN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

HAZİRAN 2018

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA SORGULARI ÜZERİNDE GÖREV TABANLI KÜMELEME

YÜKSEK LİSANS TEZİ

Almıla Selcen AKGÜN
(504121503)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Yusuf YASLAN

HAZİRAN 2018

İTÜ, Fen Bilimleri Enstitüsü'nün 504121503 numaralı Yüksek Lisans Öğrencisi Almıla Selcen AKGÜN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "ARAMA SORGULARI ÜZERİNDE GÖREV TABANLI KÜMELEME" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Dr. Öğr. Üyesi Yusuf YASLAN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Şule GÜNDÜZ ÖĞÜDÜCÜ** İstanbul Teknik Üniversitesi	.....
	Dr. Öğr. Üyesi Burcu YILDIZ Türk Alman Üniversitesi	.....

Teslim Tarihi : **2 Mayıs 2018**
Savunma Tarihi : **4 Haziran 2018**

Aileme,

ÖNSÖZ

Tez çalışmam sürecinde tecrübeleri ve bilgisi ile bana yol gösteren danışmanım Sayın Dr. Öğr. Üyesi Yusuf Yaslan'a değerli katkıları için teşekkürlerimi sunarım. Eğitim hayatım boyunca manevi desteklerini esirgemeyerek her zaman yanımda olan ve öğrenimim için her türlü imkanı sunan aileme, motivasyon kaynağım, daimi destekçim eşime sonsuz teşekkürü borç bilirim. Yüksek lisans eğitimim sürecinde verdiği maddi ve manevi destekten dolayı TÜBİTAK'a teşekkür ederim.

Haziran 2018

Almıla Selcen AKGÜN
Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

URL : Uniform Resource Locator
NLP : Natural Language Processing
DDİ : Doğal Dil İşleme
SVM : Support Vector Machine
CBOW : Continous Bag of Words
AOL : America Online
TF : Term Frequency
IDF : Inverse Document Frequency
S : Sıklık
A : Ayrışma

SEMBOLLER

Vp	**:** Sorgu öznitelik vektörü
α	**:** İki sorgu öznitelik vektörü arasındaki açı
$\cos(\alpha)$	**:** İki sorgu arasındaki kosinüs benzerliği
S	**:** Küme içi sıklık değeri
A	**:** Kümeler arası ayrışma değeri
K	**:** Küme sayısı
N	**:** Küme eleman sayısı
$S(c_i)$	**:** i. kümenin sıklık değerini
D	**:** Eğitim veri seti
n	**:** Eğitim veri setinin eleman sayısı
sim	**:** Benzerlik matrisi
maxIter	**:** Maksimum iterasyon sayısı
M_D	**:** Küme medoidleri
C	**:** Görev kümeleri
$Medoids$	**:** Küme merkezleri
maxDistance	**:** Küme elemanları arasındaki maksimum uzaklık
visitedPoints	**:** Veri setinde ziyaret edildi olarak işaretlenen noktalar

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ARAMA SORGULARI ÜZERİNDE GÖREV TABANLI KÜMELEME

ÖZET

Sorgu metinleri üzerinden görev çıkarımı, arama motorlarında ve birçok arama tabanlı uygulamalarda kullanılan önemli ve ilgi çekici konulardan birisidir. Kullanıcılar günlük aktivitelerinde ve kişisel planlarını yaparken sıklıkla internet arama motorları üzerinden sorgulama yaparlar. Metin madenciliği yöntemleriyle işlenen sorgu metinleri, popüler konu başlıklarının belirlenmesinde ve kullanıcı özelliklerinin keşfedilmesinde kullanılmaktadır.

Kullanıcılar tarafından girilen benzer sorgular çeşitli özniteliklerine göre bir araya getirilerek anlamlı görevler çıkarılabilir. Arama yapılan alana yönelik doğru sonuçların döndürülmesinde, arama metni tamamlamada, kullanıcı amacı doğrultusunda öneriler sunulmasında görev çıkarımının önemli rolü vardır.

Mevcut yaklaşımlarda kullanıcının oturum bilgisi, tıklanan doküman içerikleri ve sorgu öğeleri kullanılarak görev çıkarımı yapılmaktadır. Kullanıcının arama motorunu açtığı andan kapattığı ana kadar geçen süre kullanıcı oturumu olarak nitelendirilir. Bu süre zarfında yapılan sorgulamalar belirli bir göreve yönelik olabileceği gibi, birden fazla görev de paralel sorgulanıyor olabilir. Sorgu metinlerinin anlamsal özelliklerini temel alan çalışmalarda sorgu sonucu tıklanan doküman içerikleri, makale başlıkları önem kazanmaktadır. Önceki çalışmalarda kullanılan sorgu öğeleri ise, arama metninde geçen özel isimler, yer adları ve sözcük öbekleridir. Öğeler, sorgu içerisinde tek başına kullanıldığında bir anlam ifade ederken, bir araya geldiğinde farklı anlamlara gelen kelimelerin doğru görev kümelerine atanmasında başarımı arttırmaktadır. Son çalışmalarda, öğe kategori ilişkilerini gösteren Wikipedia kategori hiyerarşileri ve Probase taksonomi bilgileri kümeleme aşamasında kullanılarak görev kümeleri oluşturulmaktadır.

Bu çalışmada, öğe kategorileri kümeleme aşaması yerine öznitelik çıkarımı için kullanılmıştır. Böylece, arama sorguları arasındaki anlamsal benzerliğin kategoriler üzerinden daha hassas ölçümlenmesi amaçlanmıştır. Sorgular arasındaki benzerlik hesaplamalarında sorgu öznitelik vektörleri kullanılmaktadır. Diğer çalışmalardan farklı olarak, öğe ve kategori gibi sözel özniteliklerin vektörel hale getirilmesi için n-gram ve word2vec yöntemlerinden yararlanılmaktadır.

Yapılan çalışmada URL, oturum, kullanıcı, öğe ve kategori öznitelikleri farklı kombinasyonlarla birleştirilerek merkezi ve yoğunluk tabanlı kümeleme yöntemleriyle sorgular kümelenmiştir. Böylelikle, farklı öznitelik setlerinin kümeleme başarımındaki etkisi ölçümlenmiştir. Kümeleme aşamasında sayısal ve sözel öznitelikler bir arada kullanıldığı için, merkezi kümeleme yöntemi olarak K-Means algoritmasının özelleştirilmiş hali olan K-Medoids algoritması kullanılmıştır. Merkezi kümeleme yöntemlerinin gürültülü verilerden etkilendiği deney sonuçlarında görülmektedir. Bu çalışmada, gürültülü verilerin etkisini minimum seviyeye indirgemek için yoğunluk

tabanlı kümeleme yöntemi olan DB-Scan algoritması da öznitelikler üzerinde çalıştırılmış ve sonuçlardaki başarım arttırılmıştır.

Arama sorgularını temsil etmek üzere oluşturulan farklı öznitelik vektörleri elde edilen kümeleme sonuçlarının başarımı ile değerlendirilmiştir. Bir görev kümesi içerisindeki elemanların benzerliği ve görev küme merkezleri arasındaki uzaklık ölçümlenerek kümeleme sonuçları karşılaştırılmıştır. Küme içi değerlendirmede, aynı kümeye atanan elemanların birbirlerine ne kadar yakın olduğu bilgisi (Sıklık değeri) kullanılır. Kümeler arası değerlendirmede ise, her bir sorgu bir kümeye atandıktan sonra elde edilen küme merkezleri arasındaki mesafenin ne kadar uzak olduğu bilgisinden (Ayrışma değeri) faydalanılır.

Bu çalışmanın sonucunda, sorgulardan elde edilen öğe ve kategori bilgileri anlamsal öznitelikler olarak bir arada ele alınıp word2vec yöntemiyle vektörel hale getirilmiş ve yoğunluk tabanlı kümeleme yöntemi kullanılarak görev tabanlı kümeleme başarımı arttırılmıştır.

TASK-BASED CLUSTERING ON SEARCH QUERIES

SUMMARY

Task extraction on query texts is one of the important and interesting topic that used on search engines and many search-based applications. Users often search for their daily activities and personal plans on web search engines. Query texts processed by text mining methods are used in determining popular topic titles and exploring user behaviors. In this thesis, a study was carried out in the analysis of the queries entered in the search engines which collected the big data and in the meaningful task extraction.

Similar queries entered by users can be aggregated according to their various features to make meaningful tasks. There is an important role in task extraction of providing suggestions in the direction of the user's intention, in the search text completion, in returning the correct results for the domain being searched. Task clusters are identified as phrases, expressions, or more complex representations.

Session information, clicked document contents and query entities are used for feature extraction in existing approaches. The time elapsed until the user turns off the search engine is called the user session. During this period, inquiries may be directed to a specific task, or multiple tasks may be interrogated in parallel. A user session is not be appointed as a task because the user may be querying discrete tasks in parallel. In this case, it is difficult to understand user goal from queries entered into search engines in a user session.

Some other approaches use semantic features extracted from search queries to find the disjoint tasks besides to log features. Semantic features represent the meaning of a piece of search query such as entity or category. In the studies based on the semantic features of the queries, the document contents and article titles as a result of querying become important. The special names, place names and phrases in the query text are labeled as query entity. Entities increase the success in assigning words, have different meanings when they are used alone in the query or when they come together, to the correct task clusters. Task extraction is made by the Wikipedia category hierarchies or Probase taxonomy that is used at clustering level in recent studies. Queries are aggregated as per their entity category to assign similar queries into same task.

The focus of this study is to increase task clustering performance by using the semantic and lexical features of queries in the most optimal way. In this study, search queries with noisy data are first passed through the data preprocessing step. First, only punctuation marks are extracted from the dataset. Stop words (e.g. "and, the, then, also, after" in English) that do not produce any information in the semantic analysis in natural language processing (NLP), even if they have meaning, and are widely used at relevant language, removed from query log. Empty queries as the result of these operations are deleted from dataset. Lexical and semantic features are extracted from the processed data to calculate the similarity between queries. Through Dexter, the

entities of each query statement in the dataset are subtracted. There can be more than one entity in a query statement, or no entity at all. For this reason, it has been observed that in the experiments, the entity vector increases the performance of clustering queries in which the special words are used, but does not affect performance in general search words. Dexter is also used to obtain category information which the entity information belongs to. An entity can have more than one category, and each category has more than one entity. DBPedia ontology which uses Wikipedia articles is used on Dexter to map category information. For instance, when the user enter consecutive queries "cherry or grape" and "benefit of banana" to web search engine, "cherry" and "grape" are tagged as query entities for first query and "banana" is tagged as entity for second query. The category information of "cherry", "grape" and "banana" entities are "fruit". Different entities are obtained for these two queries. However, their category feature is same. If entity information is used alone to find semantic similarity, these query would be dissimilar to each other. If category information is added to feature set, their similarity would increase. In this study, entity categories are used for feature extraction instead of clustering phase. Therefore, it is aimed to measure semantic similarity between search queries on categories precisely.

Unlike some other studies, word2vec method is used for vectorization of semantic features such as entity and category. In addition, n-gram method which is one of NLP techniques has been applied to entity and category features. In another approach we used to make comparisons, the similarities between the queries were calculated by applying the word2vec method to each word in the queries. Here the query texts are separated by words according to the space (" ") character. The vector coverage of each lag was obtained using the word2vec method, which uses the Google News corpus. Words not found in corpus are evaluated as zero vector so as not to affect the result. If there is more than one word vector in a query, the query vector is calculated from the scalar sum of the word vectors. Query feature vectors are used in similarity calculations between queries. Cosine similarity is used in the computation of similarity of numerical and semantic features. As a result, a similarity matrix is obtained to be used in the clustering process from the cosine similarity between generated query vectors.

The queries are clustered by centroid-based and density-based clustering methods with the different combination of feature sets. URL, session, user, entity and category information are extracted from search queries and used as feature sets. Thus, the effect of different feature sets on clustering performance is measured in this study. Centroid and density based clustering algorithms determine task clusters. Since the numerical and semantic attributes are used together in the clustering phase, the K-Medoids algorithm, the customized version of the K-Means algorithm, is used as the centroid clustering method. Experimental results show that centroid clustering methods are affected by noisy data. In this study, DB-Scan algorithm, a density-based clustering method, was also run on the feature vectors to compare clustering result performance between centroid and density based clustering methods. In the DB-Scan algorithm, which is a density-based clustering method, queries are clustered using the minimum number of elements in one cluster and the maximum value information of the distance between two elements. In this algorithm, to find the neighbors of each point in the dataset, the distances to all other points are calculated repeatedly. The similarity matrix is used to get rid of the recursive distance calculation cost.

When the best results of the comparison are considered, it is seen that the DB-Scan algorithm achieves more successful results than K-Medoids algorithm. Density-based clustering methods are the reason for preference in text-based studies where the number of clusters can not be precisely determined since cluster numbers are determined by the dataset.

For evaluation of this research, the similarities in cluster members and the distance of clusters' centers are measured. Within-cluster evaluation uses the information (compactness value) of how close the elements assigned to the same cluster are to each other. Between-cluster evaluation, the knowledge of how far the distance between cluster centers is obtained after each query is assigned to a cluster (separation value) is used.

Our experiments demonstrate the importance of using different feature sets together to improve task extraction. The effects log features and semantic feature generation approaches using word2vec method on queries are investigated. It has been observed that when the entity and category semantic features are used in task extraction, more extensive tasks are generated. When all attributes are taken into account, the use of word2vec is influencing positively the success of the task inference, with the entity and category similarity being determined.

Experimental evaluation of the proposed method by using entity and category vectors generated by word2vec method indicates that the proposed method outperforms existing approaches. Entity and category vectors are treated as query feature and task-based clustering performance is increased by using density-based clustering method.

1. GİRİŞ

İnternet arama motorları üzerinde yapılan sorgulamaların her geçen gün artması ve işlendiğinde önemli bilgiler içeriyor olması araştırmacıların ilgi odağı olmasını sağlamıştır. Arama motorları üzerinden yapılan her bir sorgu; kullanıcı alışkanlıklarının keşfedilmesi, popüler konu başlıklarının belirlenmesi gibi önemli bilgilerin kaynağı haline gelmiştir. Kullanıcıların arama motorlarına belirli bir zaman diliminde girdiği sorgular birden fazla göreve ait olabilir. Öte yandan, kullanıcının esas amacını belirlemek için peş peşe girdiği sorguları bir arada değerlendirmek de gerekebilir. Örneğin, "Traffic rules in California", "rent a car in LA" sorguları aynı göreve atanırsa öneri sistemleri bu bilgileri kullanarak kullanıcıya "booking Hotel in Los Angeles" önerisini sorgu sonucu olarak dönebilir. Kullanıcılar tarafından belirli bir amacı gerçekleştirmeye yönelik girilen sorgular görevleri oluşturur. Literatürde, sorgular üzerinden görev çıkarımı için gerçekleştirilen ilk çalışmalarda, sorgulama sonucu tıklanan doküman bilgisi ve kısıtlı bir zaman diliminde bir kullanıcı tarafından girilen sorguları vektör veya çizge göstergesiyle kümeleyen yaklaşımlar kullanılmaktadır [3, 4]. Kullanıcı bazlı yaklaşımlarda, aynı amaca yönelik benzer sorguların farklı kullanıcılar tarafından da girilebileceği göz ardı edilmiş olur. Sorgular üzerinden sözlüksel benzerlikleri temel alan yaklaşımlarda, kelimenin cümle içindeki kullanım şekline bağlı olarak farklı anlamlarda kullanılabilmesi, anlamları dikkate almayan bu yapıların farklı sorguları tek bir görev olarak çıkarmasına neden olur. Bu da gerçekte yanlış bir çıkarım olacaktır. Örneğin, arama motoruna "Apple" kelimesi girildiğinde döndürülecek sonuçların meyve olan "Apple" mı, yoksa şirket olan "Apple" mı kastedilerek girildiğini anlamak gerekir. Bu ayrım yapılamadığında "Apple" kelimesi geçen her sorgu aynı görev kümesine atanacaktır. Yakın zamanda, Verma ve arkadaşı tarafından sorguların öğelerini [5] temel alan bir yaklaşım benimsenmiş ve başka bir çalışmalarının kümeleme aşamasında kategori hiyerarşileri [6] de kullanılarak daha başarılı sorgu kümeleri elde eden görev çıkarımı gerçekleştirilmiştir. Öğeler, sorgu içinde geçen sözcük öbekleri, özel isimler, belirli yer adları olarak tanımlanır (Örnek: Henry Ford, Paris, flowers, vb.). Sorgularda öğeler dışında kalan; fiiller, sıfatlar ve

sık kullanılan kelimeler ise terim olarak nitelendirilir (Örnek: rent, new, vb.). Görev çıkarımında sorguları genişletmek için terim sözlüğü oluşturan çalışmalar [5] özel kelimelerin kullanıldığı sorgularda başarıma katkı sağlamamaktadır.

Bu çalışmada diğer çalışmalara ek olarak, görev çıkarımındaki başarımı arttırmak için öncelikli olarak öğelerin, kategorilerin ve terimlerin Word2vec [7] gösterimi kullanılmıştır. Öğe kategorileri kümeleme aşamasında değil, sorgular arasındaki benzerlik hesaplamalarında kullanılmıştır. Karşılaştırmak üzere diğer bir yaklaşımımızda ise, sorgulardaki her bir kelimenin derin öğrenme ile elde edilen vektörleri kullanılarak farklı görev kümeleri oluşturulmuştur. Görev çıkarım başarımı küme içi ve kümeler arası uzaklıklar karşılaştırılarak ölçülmüştür. En iyi başarımı elde etmek için, sorgular arasındaki anlamsal ve sözlüksel benzerlikler farklı öznitelik kombinasyonlarıyla ele alınmıştır. Sorgulardaki yazım yanlışları, URL (Örnek: purevolume.com, myspace.cn, vb.) bilgisinin arama sözcüğü olarak girilmesi sorgulara direkt uygulanan Word2vec yönteminin başarımını düşürmektedir. Bundan kaçınmak için sorgular direkt değil, öğe ve kategori ilişkileri üzerinden ele alınmıştır. Öğe ve kategori özniteliklerine uygulanan Word2vec yöntemi, görev bazlı kümelemedeki başarımı gözle görülür ölçüde arttırmıştır.

1.1 Tezin Amacı

Arama motorları üzerinden toplanan sorguların üstel olarak artmasıyla elde edilen büyük verilerin önemli bilgiler içerdiği keşfedildikçe, arama sorguları üzerinden görev çıkarımının önemi ortaya çıkmıştır. Kullanıcıların ihtiyaçlarını tanımlayabilecek doğru görevleri çıkarmak için sorgulardan elde edilecek özniteliklerin hem log tabanlı benzerlikleri hem de anlamsal benzerlikleri içermesi gerekmektedir. Log tabanlı özellikler olarak, kullanıcı, URL verilerinden faydalanılmıştır. Sorgu tarihleri ve saatlerinden yola çıkarak oturum bilgileri elde edilmiştir. Anlamsal özellikler olarak, sorgulardan elde edilen öğeler ve onların bağlı olduğu kategori bilgileri kullanılmıştır. Metin madenciliğinde kullanılan derin öğrenme teknikleri ile kelimelerin vektörel gösterimlerinin elde edilmesinde Word2vec yöntemini kullanarak sorgu metinleri arasındaki benzerlik hesaplamalarındaki başarımın arttırılması amaçlanmıştır. Görevler benzer sorguların kümelenmesi ile elde edilmektedir. Görev kümelerinin oluşturulmasında eğitmensiz kümeleme yöntemlerinden K-Medoids ve

DB-Scan algoritmaları kullanılmış ve kümeleme başarımı küme içi ve kümeler arası mesafelerin hesaplanması ile ölçülmüştür. Görev çıkarımı başarımında, öğe ve kategori özniteliklerinin Word2vec yöntemiyle vektörleştirilerek sorgu benzerliklerinin hesaplanması yöntemi kullanıldığında en iyi sonuçların elde edildiği gözlenmiştir.

1.2 Literatür Araştırması

Literatürde sorgu metinleri üzerinde analiz gerçekleştiren birçok çalışma bulunmaktadır [3–6, 8, 9]. Görev çıkarımı ise son yıllarda gittikçe önem kazanmıştır. Görev çıkarımı üzerinde yapılan çalışmalar kullanıcı oturum bilgisini kullanmaktadır [3–6]. Kullanıcının arama motoru sayfasını açtığı andan sayfayı kapattığı ana kadar geçen süre "oturum" olarak tanımlanır. Tek bir kullanıcının aynı oturum içerisinde girmiş olduğu sorgular; zaman, içerik ve anlamsal yaklaşımlarla birbirinden ayrı görev etiketleriyle kümelenir [3]. Bu yaklaşımlarda, görev çıkarımı kullanıcı bazında özelleştirilmiş olur. Dolayısıyla, farklı kullanıcıların farklı oturumlarda aynı göreve ait benzer sorguları ayrı görev kümelerine atanır.

Lucchese ve arkadaşları oturum bilgisi belirleme işlemi için iki temel yöntem kullanmıştır. Bunlardan biri zaman boşluklarına göre sorguları ayırmaktır. 2006 AOL sorgu loglarını içeren veri kümesi üzerinde yaptıkları çalışmada bir oturumu diğerinden ayırmak için gereken en optimum süre 26 dakika olarak bulunmuştur. Aynı veri seti üzerindeki çalışmamızda sorguların oturum etiketlerini belirlemek için bu bilgiden yararlanılmıştır. Diğer yöntemlerinde ise, sorguların içerik ve anlamsal öznitelikleri üzerinden sorgu kümeleri üreterek oturum bilgileri elde edilmektedir. İki sorgu cümleciği içerisinde aynı ya da benzer terimler kullanılıyorsa bu sorgular ilişkilidir denebilir. Lucchese ve arkadaşları sorgular üzerinde 3-gram yöntemini uygulayarak Jaccard ve Levenshtein uzaklık formülleri ile sorgular arasındaki içerik bazlı mesafeyi hesaplamışlardır. Anlamsal bazlı öznitelik çıkarımında ise geniş metin içeriklerine sahip Wiktionary ve Wikipedia veri kaynakları kullanılmıştır. Sorgu terimlerinin Wiktionary ve Wikipedia makaleleri üzerinden Term Frequency - Inverse Document Frequency (TF-IDF) skorları kullanılarak anlamsal öznitelikleri vektörel hale getirilmiştir. Term Frequency (TF) bir terimin doküman içerisindeki frekansını yani, ilgili terimin doküman içinde bulunan toplam terimler sayısına oranını

gösterir. Inverse Document Frequency (IDF) ise ters doküman frekansını yani dokümanların kaçında ilgili terimin bulunduğunu gösterir. Toplam doküman adedinin ilgili terimi içeren doküman adedine oranının logaritmasıdır. TF-IDF değeri ise, TF ile IDF değerlerinin çarpımından elde edilir. Kümeleme aşamasında merkezi tabanlı yöntem olan QC-Means, yoğunluk tabanlı yöntem olan QC-Scan ve grafik bazlı yöntem olan QC-wcc algoritmaları kullanılmıştır. Bu yöntemlerin yanı sıra çalışma performansını arttırmak için tüm benzerlik grafiğini hesaplamadan bir çözüm üreten QC-htc algoritmasını geliştirmişlerdir. Bilgi çıkarım başarımı ölçümünde doğruluk, kesinlik ve F-ölçütü metrikleri kullanılmaktadır. Doğruluk, elde edilen sonucun gerçek değere yakınlığını belirtir. Kesinlik, bir analizde elde edilen sonuçların bire bir aynı yolla elde edilen diğer sonuçlara yakınlığını gösterir. F-ölçütü değeri ise, doğruluk ve kesinlik değerlerinin ağırlıklandırılmış ortalamasıdır. Sonuç olarak, F-ölçütü değeri üzerinden yaptıkları karşılaştırmada en iyi sonuçları QC-wcc algoritması üzerinden elde etmelerine rağmen ondan dört kat hızlı sonuç döndüren QC-htc algoritmasının F-ölçütü değeri sadece 1% oranında düşük çıkmıştır [3].

Görev çıkarımını zorlaştıran diğer bir nokta da aynı zaman diliminde kullanıcının birden fazla göreve yönelik aramayı paralel gerçekleştirebilmesidir. Bu belirsizlik sadece zaman parametresini ele alarak görev çıkarımının yapılamayacağının da bir göstergesidir. Verma ve arkadaşı genel bir yaklaşım sunarak, sorguların öğe etiketlerini belirleyip sorgu terimlerini öğe tipi seviyesinde birleştirerek kullanıcı bağımsız görev çıkarımı yapmaktadır [5].

Verma ve arkadaşı arama logları üzerinden görev çıkarımı yapmak üzere öncelikle sorgu terimlerinin tahmini üzerine çalışmışlardır. İkinci adım olarak da sorgu genişletme işlemiyle daha başarılı görev çıkarımlarında bulunmayı amaçlamışlardır. Sorgularda geçen öğeleri tespit ettikten sonra, Dexter aracılığı ile öğelerin tip bilgileri elde edilmiştir. Dexter, metinlerden öğe ve öğe tipi bilgilerini çıkaran bir araçtır (Detaylı bilgi için bölüm 2.3'e bakınız). Sorgu metni içerisinde, Dexter'ın tespit ettiği öğeler dışında kalan kelimeler terim olarak belirlenmiştir. Sorguların Zipf yasasındaki dağılıma uygun olduğu bilgisinden yola çıkarak sorgu metinlerinde geçen popüler öğelerin nadir öğelere oranla daha fazla sorguda tekrarlandığı tespit edilmiştir. Çalışmalarında popüler öğelerin bulunduğu sorgu kümesi eleman sayısının, nadir öğelerin yer aldığı sorgu kümesi eleman sayısından çok daha fazla olduğu ve bunun

dengesiz bir dağılıma yol açtığı belirtilmiştir. Bu durumu ortadan kaldırmak için sorgu metinlerinden elde edilen öğeler dışında kalan terimleri, ilgili sorgularda geçen öğe tipleri başlığı altında gruplayarak bir terimler sözlüğü oluşturulmuştur. Bir sorgunun öğe tipini belirledikten sonra, bu öğe tipine bağlı ve sorguda yer almayan terimleri terimler sözlüğünden bulup sorguya ekleyerek sorgu genişletme işlemi yapılmıştır. Böylece, birbirinden ayrı fakat, anlamsal olarak ilişkili sorguların daha benzer çıkmasını sağlayarak aynı görev kümesinde yer almaları sağlanmıştır. 145 oturum ve 456 terim içeren Session track data 2011, 2012 veri seti üzerinden terim sözlüğü oluşturulmuştur. Sorgu listelerinden oluşan görevlerin ne kadar özel veya genel olacağı öğe kaynaklarına bağlıdır. Öğe kaynağı olarak Verma ve arkadaşı çalışmalarında DBPedia ve Freebase kaynaklarının kullanılabileceğini belirtmişlerdir. Çalışmalarında öğe ve öğe tiplerini bulmak için Wikipedia'dan çıkarılan DBPedia'yı kullanmışlardır. Sorgu genişletme aşamasında aynı öğe tipi altında gruplanmış terimleri sıralamak için TF-IDF değerlerinden yararlanılmıştır. En iyi sonucu, maksimum 25 terim ile sorguları genişlettiklerinde elde etmişlerdir. Sonuç olarak, görev çıkarımında öğe bilgisinden yararlanılmasının başarımı arttırdığı, sorgu genişletme adımının ise spesifik kelimeler bulunmayan sorgularda başarıma olumlu etki ettiği tespit edilmiştir [5]. Ancak, sorguları genişletmek için oluşturulan terim sözlüğünün spesifik kelimeleri barındıran aramalarda başarımı etkilemediği belirtilmiştir.

Hua ve arkadaşları kısa metinlerden oluşan sorguların anlamlarının belirlenmesinin ya da karmaşık görev çıkarımları yapılmasının zorluğunun aşılabilmesi için sorguda geçen her bir kelimenin o kelimeyi içeren bir Wikipedia makalesi ile eşleştirilmesinin gerektiğinden bahsetmektedir. Verilen örnek "tiger woods" sorgusunda problem ve çözümü daha iyi anlaşılmaktadır. Tiger Woods, meşhur ve başarılı bir golfçüdür. Arama motoruna girilen "tiger woods" sorgusu kelime kelime ele alındığında, "tiger" kelimesi genel kullanımı ile kaplan olarak değerlendirilip hayvan kategorisinde ele alınır. "woods" kelimesi ise genel kullanımı ile tahta olarak değerlendirilip materyal kategorisinde ele alınır. Ancak, bu sorguda "tiger" kelimesi bir lakap, "woods" ise oyuncunun soyadıdır. Dolayısıyla, "tiger woods" Wikipedia makalesine adreslenerek, bir öğe olarak ele alınmalı ve birlikte değerlendirilmelidir. Bir oturumda birden fazla göreve ait arama yapılabileceği için Hua ve arkadaşları çalışmalarında sıralı kesme (Sequential Cut) ve çizge kesme (Graph Cut) algoritmalarını kullanmıştır. Mayıs 2012

yılında ticari bir internet sitesi üzerinden bir günde toplanan 45813 oturum içerisinden en az 10 sorgu içeren rastgele 600 oturum seçerek oluşturdukları veri seti üzerinde çalışmışlardır. Oturum sorguları arasında benzerlik hesaplamalarında kavramsal (conceptual), sözlüksel (n-gram Jaccard), şablon (Levenshtein düzenleme mesafesi) özniteliklerini kullanmışlardır. Görev kümelerinin belirlenmesinde eğitmenli makine öğrenmesi algoritmalarından Lojistik Regresyon, Naive Bayes, Doğrusal Destek Vektör Makinesi (SVM) algoritmalarını kullanmışlardır [4].

Yakın zamandaki çalışmalarda ise, görev kümelerinin budaması aşamasında Wikipedia kategori hiyerarşisi kullanılarak daha başarılı görev çıkarımı gerçekleştirilmiştir. Görev çıkarımı işleminde, öncelikle sorgu öğeleri ve öğe tipleri etiketlenmiş, sonrasında da her bir öğe tipi altında bu öğe tipiyle ilişkili olan sorgular gruplanmıştır. Eğer bir sorgu uygun bir görev kümesine atanamazsa kategori hiyerarşisinde kendine en benzer sorguların bulunduğu üst düğüm ya da alt düğümün olduğu görev kümesine bağlanır. Sorguların kümelenmesinde K-Means algoritmasının bir çeşidi olan ancak, küme sayısının önceden bilinmesinin gerekmediği, veri setinden küme sayısının belirlenebildiği DPMeans algoritması kullanılmıştır [6].

Bu çalışmada, Verma ve arkadaşına benzer bir yaklaşımla Wikipedia makalelerini kullanan DBPedia ontolojisi üzerinden her bir sorgunun öğeleri ve kategorileri bulunmaktadır. Bundan farklı olarak, öğe ve kategori bilgileri Word2vec yöntemiyle vektörel hale getirilip anlamsal öznitelik çıkarımı aşamasında kullanılarak görev çıkarımı başarımına katkı sağlanmaktadır.

2. GÖREV ÇIKARIMI

2.1 Görev Çıkarımı Nedir

Görev (task), amaca yönelik girilmiş sorgular dizisi olarak tanımlanabilir. Arama motoruna girilen sorguların hangi görevi yerine getirmek üzere girildiği bilgisini üretmek için sorgu metinleri işlenir. Örneğin; "tours for singles", "solo east travel", "travelling items for babies" sorguları aynı görev kümesine ait sorgulardır. Bu sorgular "travel" görevinin altında birleştirilebilir. Görev kümeleri; sözcük öbekleri, ifadeler ya da daha karmaşık gösterimlerle tanımlanabilir [5]. Bu çalışmada, arama geçmişindeki sorgular kümelenerek görev listeleri oluşturulmaktadır. İngilizce sorgular için örnek görev ve ilgili görev kümesinde yer alan sorgular Çizelge 2.1'de gösterilmiştir.

Çizelge 2.1 : Görev ve sorguları.

CAPITAL	HOLLYWOOD ACTRESS
"Paris and Rome"	"Who is Jennifer Lawrence?"
"Travel to Berlin"	"Who are the most famous actresses in Hollywood?"
"Flights to Washington "	"Best actress Oscar for Audrey Hepburn"
"Best places in Vienna"	"List of the best Nicole Kidman movies"

2.2 Sorgu Metinlerini Göstermek İçin Kullanılan Öznitelikler

Arama sorgularının görev kümelerine atanabilmesi için öncelikle veri temizleme adımından geçirilmesi gerekmektedir. Görev kümelerini oluşturmada etkisi olmayan etkisiz sözcükler veri setinden çıkarıldıktan sonra, anlamlı görev kümeleri elde edebilmek için sorgu cümleciklerinden veriyi doğru niteleyen öznitelik vektörleri üretilir. Sorgu geçmişi verilerinden yararlanılarak arama sorguları üzerinden sözlüksel öznitelik vektörleri üretilir. Yapılan çalışmalarda kullanılan sözlüksel öznitelik vektörlerine URL vektörü, kullanıcı vektörü, oturum vektörü örnek verilebilir.

- URL özniteliği: Veri setindeki bir sorgu için her bir URL'in kaç kez tıklandığı bilgisini gösterir. Kimi çalışmalarda URL'lerin tıklanma sayısı yerine tıklanma sırası da kullanılmaktadır.

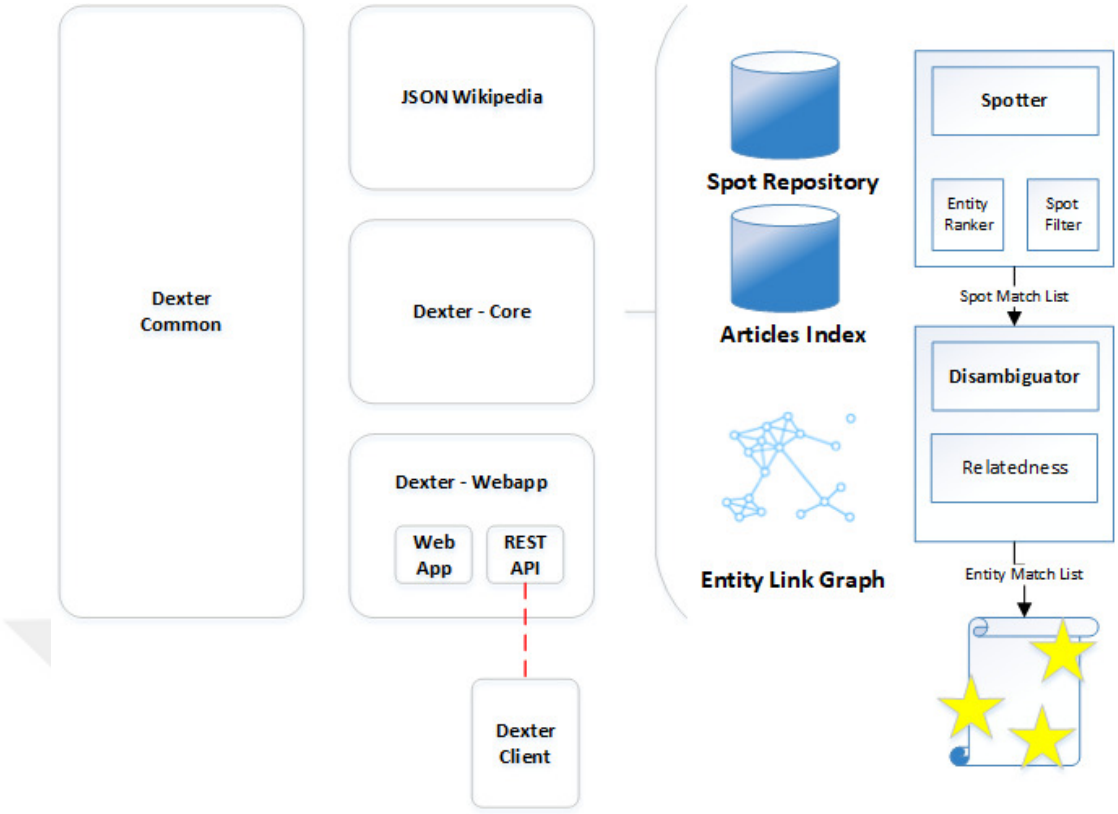
- Kullanıcı özniteliği: Veri setindeki bir sorgu için her bir kullanıcının ilgili sorguyu kaç kez girdiği bilgisini gösterir.
- Oturum özniteliği: Veri setindeki bir sorgu için her bir kullanıcı oturumunda ilgili sorgunun kaç kez girildiğinin bilgisini gösterir.

Sözlüksel özniteliklerin yanı sıra, sorgularda kullanılan kelimelerin anlamlarını temel alarak mantıksal açıdan da tutarlı görev kümeleri oluşturabilmek için anlamsal öznitelik vektörlerini de kullanmak gerekir. Anlamsal öznitelik vektörlerinde öğe ve kategori özniteliklerinden yararlanılabilir. Sözlüksel ve anlamsal öznitelikler, sözcük öbekleri arasındaki benzerliğin hesaplanmasında kullanılmaktadır.

- Öğe özniteliği: Veri setindeki bir sorgu cümlesinde bulunan kelimeler tek başına bir anlam ifade ettiği gibi, bir araya gelip bir sözcük öbeği oluşturabilir. Sorguda geçen anlamlı sözcük öbekleri, özel isimler ve yer adları öğe (entity) olarak isimlendirilir. Veri setindeki bir sorguda geçen öğeler tespit edilerek anlamsal açıdan daha benzer sorguların aynı kümede yer almasını sağlamak üzere öğe özniteliği bilgisi kullanılır.
- Kategori özniteliği: Veri setindeki her bir sorgunun öğeleri tespit edildikten sonra ilgili öğelerin hangi kategoriye ait olduğu bilgisinin benzer sorguların gruplandırılmasında faydalı ve önemli bir rolü vardır. Benzer çalışmalarda, öğelerden kategori (category) bilgisinin elde edilebilmesi için Wikipedia kategori hiyerarşisinden yararlanılmaktadır [6].

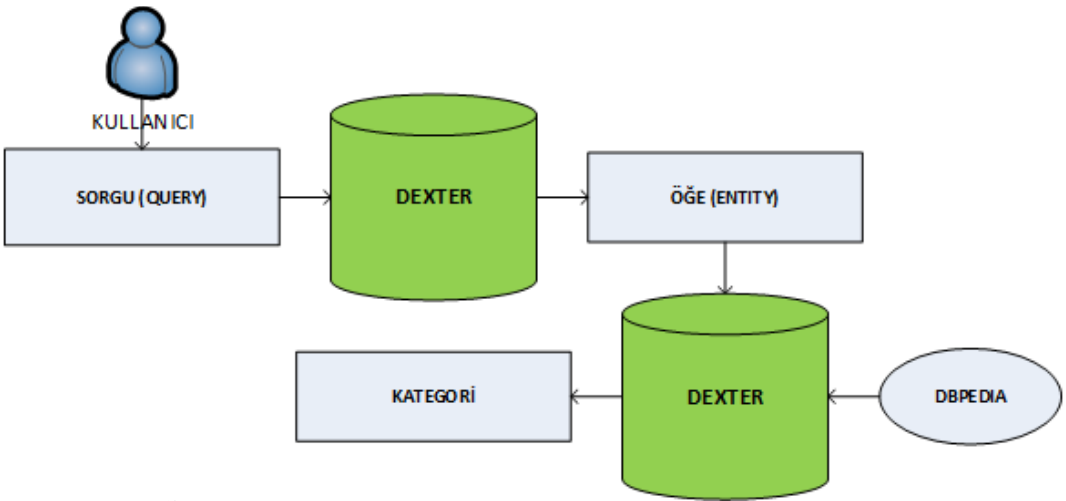
2.3 Dexter ile Öğe ve Kategori Çıkarımı

Dexter, öğe bağlantılarını bulan açık kaynak kodlu bir arayüz programıdır. Wikipedia makaleleri gibi belirli bir bilgi kaynağında bulunan öğelere atıfta bulunarak bir metnin tüm küçük metin parçalarını tanımlamayı amaçlamaktadır. Bir girdi metni verildiğinde, bir varlığa atıfta bulunabilecek metin parçaları keşfedilmeye çalışılır. Bir sözcükten birden fazla varlığa atıfta bulunulabileceğinden, adaylar arasından doğru varlığın seçildiği bir belirsizlik adımı gerçekleştirilir. Son olarak, keşfedilen varlıklar benzerlik seviyesine göre sıralanır [10]. Şekil 2.1'de Dexter'ın genel çalışma akış şeması paylaşılmıştır [2].



Şekil 2.1 : Dexter genel çalışma akışı [2].

Öğe kategori ilişkilerinin bulunabilmesi için DBPedia kategori hiyerarşisi Dexter ile ilişkilendirilerek ayrıca bu çalışmada yazılım geliştirmesi yapılmıştır. Kategori bilgileri çıktısının üretilmesi için Dexter öğe bilgisini girdi parametresi olarak beklemektedir. Dexter aracılığıyla öğe ve kategori çıkarım akışı Şekil 2.2'de gösterilmektedir.



Şekil 2.2 : Dexter üzerinden öğe ve kategori çıkarım akışı.

2.4 Anlamsal Öznitelikleri Vektörel Hale Dönüştürme Yöntemleri

Öğe ve kategori anlamsal öznitelikleri karakter seti halinde elde edilmektedir. Sözel özniteliklerin benzerlik hesaplamalarında kullanılmak üzere sayısal hale dönüştürülmesinde, tekrar sayısına dayanan N-gram yöntemi ile derin öğrenme tekniğini kullanan Word2vec yöntemi kullanılmaktadır.

2.4.1 N-gram yöntemi

Bir sorgu cümlesi üzerinde arama yapmak, benzer sorgu cümleciklerini karşılaştırmak veya bir sorgudaki tekrar sayısını saptayabilmek için kullanılan yöntemdir [11]. N-gram yöntemi, verilen bir harf serisindeki n değerine bağlı tekrar oranını bulmayı temel alan bir yöntemdir. Gram değeri de bu tekrarın harf serisi içerisindeki ağırlığını ifade eder. Bu yöntem harf bazlı ya da kelime bazlı kullanılabilir. Sorgu cümlecikleri kısa ve yazım hatalarına müsait olduğu için kelime bazlı n-gram yerine harf bazlı n-gram yöntemi bu çalışmada tercih edilmiştir.

Sorgular arasındaki benzerliğin hesaplanabilmesi için sorguların öncelikle n-gram yöntemiyle vektörel hale getirilmesi gerekmektedir. Sorgu içerisindeki tekrarlar harf harf kaydırılarak tüm sorgu içerisinde saydırılır. Örneğin; trigram (3-gram) yönteminde $n = 3$ olarak kabul edilir.

Çizelge 2.2'de iki benzer arama sorgusu olan "run Forest run" ile "running" ifadelerinin harf bazlı trigram yöntemiyle ayrıştırılmış tekrar sayıları gösterilmektedir. Sorgu içerisinde geçen boşluk karakteri (" ") de kelimeleri birbirinden ayırdığı için bir harf gibi değerlendirilmektedir. Sorgular, ilk karakterinden başlanarak üçer karakterlik dizilim parçacıklarına ayrılır. Her bir üçlü dizilimin aynı sorgu içinde kaç kez tekrarlandığı bulunur. Bu işlemler sonucunda elde edilen sorgu vektörleri arasındaki benzerlik hesaplamalarında kosinüs benzerliğinden yararlanılmaktadır.

Sorgu 1 : "run Forest run" - [2,1,1,1,1,1,1,1,1,1,1]

Sorgu 2 : "running" - [1,1,1,1,1]

Çizelge 2.2 : Trigram yöntemiyle harf bazlı tekrar hesaplama.

"run Forest run"		"running"	
run	2	run	1
un	1	unn	1
n F	1	nni	1
Fo	1	nin	1
For	1	ing	1
ore	1		
res	1		
est	1		
st	1		
t r	1		
ru	1		

2.4.2 Word2vec yöntemi

Word2vec kelimelerin anlamlarına ve kullanım yerlerine göre vektörize edilebilmesini sağlayan bir yöntemdir. Aynı anlama gelen, aynı kategorideki veya birlikte kullanılan kelimelere yakın vektörler üretir. Kelimelere atanan vektörler arasındaki kosinüs benzerliklerine bakıldığında Word2vec modeli benzer anlamlı kelimeler için yüksek, farklı anlamlı kelimeler için düşük benzerlik değerleri üretmektedir. Böylece, Word2vec kelimeler arasındaki uzaklığın vektörel olarak hesaplanabilmesini sağlamaktadır. Word2vec yöntemi ile kelimeler arasındaki benzerliklerin kolaylıkla hesaplanabilmesinin yanı sıra bir kelimeye en yakın kelime listeleri de belirlenebilmektedir. Benzer yerlerde kullanılan kelimelerin, örneğin benzer bir çok makalede beraber kullanılmış, benzer anlamlı olacağı kabulüne dayanan bir yöntemdir.

Word2vec modeli kurulurken girdi olarak verilen dokümanlardaki kelimeleri pencereler üzerinden tarayarak ilerler. Pencere boyutu ortadaki kelimenin sağından ve solundan kaç kelimeyle ilişkilendirilebileceğini belirtmektedir.

Word2vec yönteminde yapay sinir ağları öğrenmesi tekniklerinden CBOW ve Skip-Gram yöntemleri kullanılmaktadır [12]. Bu modeller birbirinden girdi alma ve çıktı üretme açısından farklılık gösterir. CBOW yönteminde pencerenin merkezinde olmayan kelimeler girdi olarak alınıp, merkezinde olan kelimeler çıktı olarak tahmin edilmeye çalışılırken; Skip-Gram modelinde ise merkezdeki kelime girdi olarak alınıp,

merkezde olmayan kelimeler çıktı olarak tahmin edilmeye çalışılır. Bu işlem girdi dokümanındaki tüm cümleler için cümle bitene kadar uygulanır.

Örneğin; "An Algerian military transport plane crashed Wednesday just after taking off from an air base south of the capital, Algiers, killing 257 people, according to a statement from the Ministry of Defense." cümlesinde pencere boyutu 3 seçildiği varsayımıyla her bir kelime kendinden önce gelen üç kelime ve kendinden sonra gelen üç kelime ile ilişkilendirilir.

CBOW yönteminde pencere merkezindeki kelime "transport" olarak alınırsa, öncesindeki "An Algerian military" ve sonrasındaki "plane crashed Wednesday" kelimelerinden yararlanılarak pencere merkezindeki "transport" kelimesi yapay sinir ağları modeli ile tahmin edilmeye çalışılır. Sonra, pencere bir sağa kaydırılarak cümle bitene kadar bu işlem devam ettirilir.

Skip-Gram yönteminde ise pencere merkezindeki "transport" kelimesinden yararlanılarak onun çevresindeki en yakın "An Algerian military" ve "plane crashed Wednesday" kelimeleri tahmin edilmeye çalışılır.

CBOW modelleri küçük veri setlerinde tercih edilen ve daha az hesaplama gücü gerektiren bir yöntemken, Skip-Gram büyük veri setlerinde daha iyi çalışan, daha fazla hesaplama gücü gerektiren ve birden çok anlamlı kelimeleri daha iyi öğrenebilen bir yöntemdir.

Word2vec yönteminin sonucunda bütün kelimelerin olasılık değerlerini içeren ve tekilleştirilmiş kelime sayısı büyüklüğünde bir vektör elde edilir.

Arama sorguları içerisinden "healthy nutrition and sports" sorgusu üzerinden Word2vec yönteminin çalışmamızdaki kullanımını örnekleyebiliriz. Sorgu veri önişleme adımından sonra "healthy nutrition sports" olarak Word2vec yöntemi uygulanmak üzere boşluk (" ") karakterine göre kelimelerine ayrılır. "healthy", "nutrition" ve "sports" kelimeleri Google News [13] modeli üzerinden Word2vec yöntemiyle 300 boyutlu üç ayrı vektör olarak üretilir. Benzerlik hesaplamaları sorgular arasında yapıldığı için, Word2vec yöntemiyle üretilen bu üç vektör toplanarak ilgili sorgunun vektörü elde edilir.

2.5 Öznitelikler Arası Benzerlik Hesaplama

Sözlüksel öznitelikler sayısal değerlerle gösterilirken, anlamsal (semantic) öznitelikler sözel ifadelerden oluşur. Dolayısıyla, anlamsal özniteliklerin N-gram ya da Word2vec yöntemleriyle öncelikle sayısal ifadelerinin bulunması gerekir. Arama sorgularından üretilen öznitelik vektörleri arasındaki benzerlik hesaplamalarında kosinüs benzerliği kullanılır [14].

Sorgu geçmişinden seçilen herhangi bir p ve q sorguları çeşitli özniteliklerine göre sayısal hale getirilip Vp ve Vq vektörleri ile gösterilir. Oluşturulan iki vektör arasındaki α açısının kosinüs değerinden sorgular arasındaki benzerlik Denklem 2.1'deki formül ile hesaplanır. Vp_i ve Vq_i, ilgili vektörün her bir i. öznitelik (sütun) değerini; n ise, vektörlerin boyutunu gösterir. Kosinüs benzerliği -1 ile 1 arasında değerler alır. Birebir aynı yönü gösteren iki vektör arasında kosinüs benzerliği 1 değerini alacaktır. Birbirinden tamamen farklı vektörler arasındaki dik açının kosinüs değeri ise 0 olacaktır. Zıt yönleri gösteren iki vektör arasındaki benzerlik ise -1 değerini alır ki, bu iki sorgu arasında zıtlık ilişkisi bulunduğu düşünülebilir.

$$\cos(\alpha) = \frac{VpVq}{||Vp||||Vq||} = \frac{\sum_{i=1}^{n} Vp_i Vq_i}{\sqrt{\sum_{i=1}^{n}(Vp_i)^2}\sqrt{\sum_{i=1}^{n}(Vq_i)^2}} \tag{2.1}$$

3. GÖREV BAZLI KÜMELEME YÖNTEMLERİ

Mevcut çalışmalarda görev çıkarımı için merkez, yoğunluk veya grafik tabanlı kümeleme yöntemleri kullanılmıştır. Bu çalışmada merkez tabanlı kümeleme yöntemi olarak K-Medoids algoritması, yoğunluk tabanlı kümeleme yöntemi olarak DB-Scan algoritması kullanılmıştır.

3.1 K-Medoids Algoritması

Öznitelik setimizde hem sözel hem de sayısal bilgiler olduğu için Öklit uzaklığı ve aritmetik ortalama ile küme merkezlerini bulan standart K-Means algoritmasının özelleştirilmiş hali olan K-Medoids [15] methodu kullanılmıştır. K-Means algoritması başlangıç aşamasında rastgele, sonrasında ise veri setindeki noktaları kullanarak sanal bir merkez belirler. Çıkarılan öznitelikler içerisinde öğe ve kategori öznitelikleri karakter setleri halinde saklanmaktadır. Dolayısıyla, sanal bir merkez belirlemek yerine veri setinde bulunan en merkezi orta noktayı küme merkezi olarak atamak veri setine daha uygun bir yöntemdir. K-Medoids algoritmasının küme merkezi seçiminde, noktaların birbirine olan uzaklıkları hesaplanarak her bir noktanın diğer noktalara olan uzaklıkları toplanır. Minimum toplam mesafeye sahip nokta diğer noktalara en yakın nokta olduğu için küme merkezi olarak belirlenir.

Kümeleme algoritmaları tarafından kullanılan sorgu metinleri arasındaki uzaklık, her bir sorgu çiftinin benzerliği ile ters orantılıdır. Denklem 3.1 sorgular arasındaki uzaklık hesaplamasını gösterir.

$$uzaklik = 1 - benzerlik \tag{3.1}$$

K-Medoids aynı zamanda aykırı verilere karşı K-Means algoritmasına oranla daha toleranslıdır. Aykırı veriler, kümenin diğer elemanlarından çok daha uzak olan noktalar olarak tanımlanabilir. Aykırı veriler küme merkezi hesaplarına dahil edildiğinde, Öklit uzaklığını kullanarak merkez noktasını bulan K-Means algoritması kümenin gerçek

merkezinden aykırı veriye doğru saparak yeni bir merkez belirleyecektir. Bu da gerçekte o kümede olması gereken elemanların yeni küme merkezinden uzaklaşmasına ve dolayısıyla farklı kümelere atanmasına sebep olacaktır. K-Medoids algoritmasında küme elemanları içerisinden seçilen orta nokta aykırı verilerden daha az etkilenecektir. Çünkü, küme içindeki her bir nokta çifti arasındaki mesafenin minimize edilmesini temel alarak orta noktayı belirlemektedir. Sözel özniteliklerden sanal bir merkez üretilemeyeceği için ve arama sorguları içerisindeki yazım hatalarından ortaya çıkan aykırı ve gürültülü verilerden daha az etkilenebilecek merkezi tabanlı kümeleme yöntemi olarak K-Medoids algoritması seçilmiştir.

K-Medoids algoritması küme sayısını parametre olarak başlangıçta alır. Veri setinde bulunan n adet noktadan rastgele seçilen K adet nokta başlangıç küme merkezleri (medoid) olarak atanır. Veri setinde kalan diğer tüm noktalar Denklem 3.1'de gösterilen uzaklık hesaplama yöntemiyle kendisine en yakın medoid ile birleştirilerek K adet küme oluşturulur. Aynı küme içindeki her bir noktanın diğer noktalara olan uzaklıkları hesaplanarak uzaklıklar toplamı alınır. Minimum uzaklıklar toplamına sahip küme elemanı yeni medoid olarak atanır. Medoid değerleri değişmeyene ya da iterasyon sayısı tamamlanana kadar kümeleme adımları tekrarlanır. K-Medoids algoritmasının çalışma adımlarına ait sözde kod Çizelge 3.1'de detayları ile paylaşılmıştır.

Bu yöntemde, K parametresi olarak önceden verilen küme sayısı bilgisi, metinsel sınıflandırmalarda karmaşıklığa yol açmaktadır. Metinlerin kümelenmesinde oluşacak küme sayılarının önceden tahmin edilmesi zordur. K parametresinin belirlenmesinde aynı veri seti üzerinde çalıştırılan yoğunluk tabanlı yöntem sonucu oluşan küme sayıları referans alınmaktadır.

Merkez bazlı yöntemler ayrıca sorgu geçmişindeki gürültülü verilerden de etkilenmektedir. Merkez tabanlı yöntemler veri setindeki aykırı verileri de bir kümeye dahil ederek küme merkezinin gerçek merkezden aykırı veriye doğru sapmasına sebep olmaktadır. Bununla birlikte, aynı kümeye dahil olan küme elemanları arasındaki uzaklık da artmaktadır. Dolayısıyla, küme içi değerlendirmede merkez tabanlı K-Medoids algoritmasının başarımı yoğunluk tabanlı DB-Scan algoritmasına göre düşüktür.

Çizelge 3.1 : K-Medoids Algoritmasının sözde kodu.

Girdi : Küme sayısı K, Eğitim veri seti $D = \{d_1, d_2, ..., d_n\}$,
Eğitim veri setinin eleman sayısı n, Benzerlik matrisi sim,
Maksimum iterasyon sayısı maxIter

Başlangıç : Rastgele K adet elemanı medoid olarak ata.
$M_D = \{m_1, m_2, ..., m_K\}$; M_D küme medoidleri, $m_K \in D$;
$iterasyon = 0$;

Tekrar :
1. **döngü :** $i = 1..n$
   döngü : $j = 1..K$
   distance$[i][j] = 1 - sim[i][m_j]$; $d_i \notin M_D$
   döngü sonu
   minimum distance$[i][j]$ için d_i elemanını c_j kümesine ata.
   $C = \{c_1, c_2, ..., c_K\}$ kümeleri oluşur.
   döngü sonu
2. **döngü :** $i = 1..K$
   döngü : $j = 1..c_i$'nin eleman sayısı
   $dist[j] = 0$;
   döngü : $z = 1..c_i$'nin eleman sayısı
   $dist[j] = dist[j] + (1 - sim[c_{ij}][c_{iz}])$
   döngü sonu
   minimum $dist[j]$'ye sahip c_{ij} elemanını bul.
   döngü sonu
   eğer $c_{ij} \neq m_i$ ise
   $m_i = c_{ij}$; (c_i kümesinin yeni medoidi olarak ata.)
   değilse
   $converge[i] = true$;
   eğer sonu
   döngü sonu
3. $iterasyon = iterasyon + 1$;
4. $Medoids = M_D$;
5. **eğer** ($maxIter > 0$ ve $iterasyon >= maxIter$) ise
   döngüden çık.
   eğer sonu
   eğer $converge[1..K] = true$ ise
   döngüden çık.
   eğer sonu

Çıktı : C kümeleri, $Medoids$ küme merkezleri

3.2 DB-Scan Algoritması

Yoğunluk bazlı kümeleme yöntemi olan DB-Scan [16] algoritması küme sayısının önceden belirlenemediği veri setleri için ideal bir kümeleme yöntemidir. Bir kümede minimum kaç eleman olması gerektiği ve iki eleman arasındaki uzaklığın maksimum değeri bilgileri kullanılarak sorgular kümelenir. Bu algoritmada, veri setindeki her bir noktanın komşularını bulmak için diğer tüm noktalara olan uzaklıklarına döngüsel olarak tekrar tekrar bakılmaktadır. Döngüsel uzaklık hesaplama maliyetinden kurtulmak için benzerlik matrisi kullanılmıştır.

DB-Scan algoritması kullanılarak üretilen küme sınırları içerisinde yoğunluk fazlayken, aykırı veya gürültülü verilerin bulunduğu yerlerde yoğunluk azdır. Dolayısıyla, DB-Scan algoritması kümelerin birbirinden farklı şekillerde ayrıldığı veri setlerinde en ideal kümelemeyi yapmaktadır. K-Medoids algoritmasında yani merkezi tabanlı kümeleme yöntemlerinde olduğu gibi küme şekillerinin dairesel olması gerekmez. Küme sayısının önceden bilinmesine gerek yoktur, veri setine bağlı olarak küme sayıları kümeleme adımında oluşturulur. Böylece, sabit bir küme sayısı sınırlaması gerektirmez. Aykırı verilerin tespit edilmesinde başarılı olduğu için gürültülü veri setlerinde kümeleme başarımı yüksektir. Büyük hacimli verilerde hesaplama maliyeti yüksek olmasına karşın başarılı sonuçlar üretmektedir.

DB-Scan algoritması küme içindeki elemanlar arası mesafenin maksimum değerini (maxDistance) ve bir kümede minimum kaç eleman (minNumElements) olması gerektiği bilgilerini parametre olarak alır. Veri setinden rastgele bir başlangıç noktası seçilir (x_i). x_i noktasının veri setindeki tüm diğer noktalarla olan uzaklıkları hesaplanır. maxDistance değerinden küçük veya eşit tüm komşuları bulunur. Eğer en az minNumElements adet komşusu varsa x_i noktası çekirdek nokta olarak işaretlenir. Eğer komşu sayısı minNumElements değerinden az fakat, herhangi bir çekirdek noktanın komşusu ise sınır nokta olarak işaretlenir. Çekirdek ya da sınır nokta olmayan veriler aykırı yani gürültülü verilerdir. Veri setindeki tüm çekirdek noktalar bulunur ve her biri bir küme oluşturacak şekilde etiketlenir. Özyinelemeli olarak çekirdek nokta ile yoğunluk bağlantılı tüm noktalar aynı küme etiketiyle etiketlenir. Veri setindeki tüm noktalar ziyaret edilene kadar işlem sürdürülür. DB-Scan algoritmasının çalışma adımlarına ait sözde kod Çizelge 3.2'de detayları ile paylaşılmıştır.

Çizelge 3.2 : DB-Scan Algoritmasının sözde kodu [1].

Girdi : Eğitim veri seti $D = \{d_1, d_2, ..., d_n\}$, Benzerlik matrisi sim,
Küme elemanları arasındaki maksimum uzaklık maxDistance,
Minimum küme eleman sayısı minNumElements

Başlangıç : Veri setinde ziyaret edildi olarak işaretlenen noktalar
$visitedPoints = NULL$;

1. $C = 0$;
2. **döngü :** işaretlenmemiş her bir P için $(P \in D)$
   P noktasını $visitedPoints$'e ekle;
   $circlePoints$ = komsuNoktalarıBul$(P, maxDistance)$;
   eğer $circlePoints$'in eleman sayısı $>= minNumElements$
   $C =$ yeni küme;
   kümeGenislet$(P, circlePoints, C, maxDistance, minNumElements)$;
   eğer Sonu
   döngü sonu

kümeGenislet$(P, circlePoints, C, maxDistance, minNumElements)$:
1. P noktasını C kümesine ekle;
2. **döngü :** her bir P' için $(P' \in circlePoints)$
   eğer $P'visitedPoints$ de değilse
   $P'visitedPoints$ e ekle;
   $circlePoints' =$ komsuNoktalarıBul$(P', maxDistance)$;
   eğer $circlePoints'$ nün eleman sayısı $>= minNumElements$
   $circlePoints = circlePoints \cup circlePoints'$;
   eğer sonu
   eğer P' hiç bir kümenin elemanı değilse
   P' noktasını C kümesine ekle;
   eğer sonu
   eğer sonu
   döngü sonu

komsuNoktalarıBul$(P, maxDistance)$:
döndür : P noktasını merkez alarak
$maxDistance$ çapındaki bir dairenin içinde kalan tüm noktaları

Çıktı : C kümeleri

Aykırı noktaların yoğunluk bağlantısı bulunamayacağı için herhangi bir kümeye atanamazlar. Bu yaklaşımla oluşturulacak kümelerin sorgu geçmişindeki gürültülerden etkilenmemesi hedeflenmiştir.

4. DEĞERLENDİRME YÖNTEMLERİ

Merkezi ve yoğunluk bazlı kümelerin başarımını ölçmek için küme içi ve kümeler arası değerlendirme yöntemleri kullanılır [17]. Bu yöntemlerde sıklık ve ayrışma değerlerine göre kümeleme başarımı ölçülür. Öznitelik setlerini çeşitli kombinasyonlarda kullanarak elde edilen sorgu kümelerinin görev çıkarımındaki başarımına etkisi bu değerler üzerinden karşılaştırılmaktadır.

4.1 Küme İçi Değerlendirme Yöntemi (Sıklık)

Sıklık (S), bir küme içindeki veri noktalarının ne kadar yakın olduğunu gösterir. Küme içerisindeki her bir noktanın diğer noktalara olan uzaklıkları toplamının ortalaması alınarak küme içi sıklık değeri hesaplanır.

Bir c kümesinin sıklık değeri $S(c)$ hesaplaması Denklem 4.1'de tanımlanmıştır. Küme içindeki noktalar arasındaki uzaklık hesaplamalarında önceden oluşturulmuş kosinüs benzerlik matrisindeki değerlerden yararlanılır. Bir kümedeki x_i ve x_j noktaları arasındaki benzerlik *sim* benzerlik matrisindeki $sim[x_i][x_j]$ hücresinde tutulmaktadır. Bu iki nokta arasındaki uzaklık Denklem 3.1'de gösterilen formül ile bulunur. N, c kümesinin eleman sayısını temsil eder.

$$S(c) = \frac{2}{N \times (N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} (1 - sim[x_i][x_j]) \tag{4.1}$$

Denklem 4.2'de gösterildiği gibi, kümeleme sonucu oluşan tüm kümeler $c_1, c_2, ..., c_K$ için elde edilen sıklık değerlerinin aritmetik ortalaması alınarak kümeleme sıklığı (S) hesaplanır. K, sistem tarafından üretilen küme sayısını; $S(c_i)$, i. kümenin sıklık değerini ifade eder.

$$S = \frac{1}{K} \sum_{i=1}^{K} S(c_i) \tag{4.2}$$

4.2 Kümeler Arası Değerlendirme Yöntemi (Ayrışma)

Ayrışma (A), oluşan kümelerin birbirinden ne kadar ayrık olduğunu ölçer. Her bir küme merkezinin diğer küme merkezlerine olan uzaklıkları toplamının aritmetik ortalaması alınarak kümeler arası ayrışma değeri hesaplanır.

Sistem tarafından üretilen $c_1, c_2, ..., c_K$ kümelerinin ayrışma değeri olan A Denklem 4.3'de gösterilen formül ile hesaplanır. Küme merkez noktaları veri setindeki noktalar arasından seçildiği için bu noktalar arasındaki uzaklık hesaplamalarında önceden oluşturulmuş kosinüs benzerlik matrisinden yararlanılır. $\overline{x}_i$, i. kümenin merkez noktasını; $\overline{x}_j$, j. kümenin merkez noktasını gösterir. i ve j küme merkezleri arasındaki benzerlik *sim* benzerlik matrisindeki $sim[\overline{x}_i][\overline{x}_j]$ hücresinde tutulmaktadır. Bu iki nokta arasındaki uzaklık Denklem 3.1'de gösterilen formül ile bulunur. K, sistem tarafından üretilen küme sayısını temsil eder.

$$A = \frac{2}{K \times (K-1)} \sum_{i=1}^{K} \sum_{j=i+1}^{K} (1 - sim[\overline{x}_i][\overline{x}_j]) \tag{4.3}$$

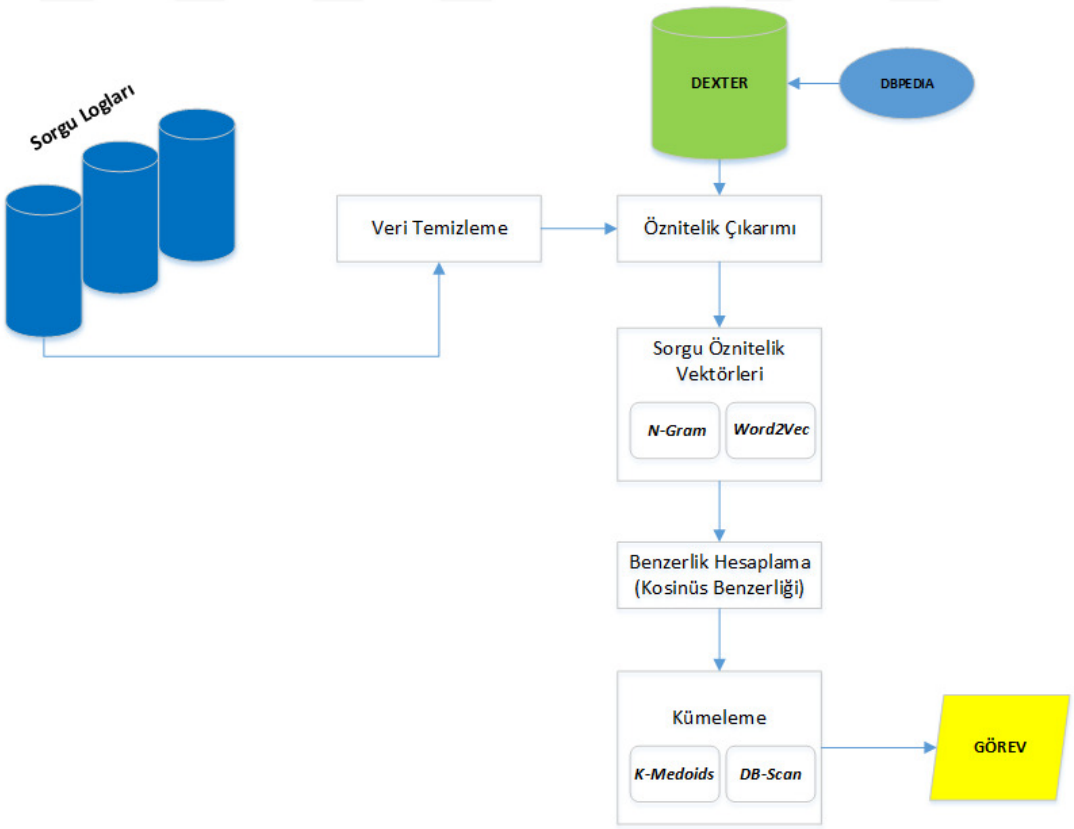
Değerlendirme hesaplamaları doğrultusunda kümeleme başarımı aşağıdaki maddeler dikkate alınarak değerlendirilir.

- Küme içi değerlendirmede, sıklık değeri ne kadar küçük olursa, aynı kümeye atanmış sorgular arasındaki mesafe de o kadar yakındır.
- Kümeler arası değerlendirmede, ayrışma değeri ne kadar büyük olursa, sorgu kümelerinin merkezleri arasındaki mesafe de o kadar uzak olur.

Bu bilgiler ışığında başarılı bir kümenin ayrışma değerinin yüksek, sıklık değerinin düşük olması beklenir.

5. GELİŞTİRİLEN YÖNTEM

Arama motoruna kullanıcılar tarafından girilen sorgu logları veri kaynağı olarak ele alınır. Gürültülü veriler içeren arama sorguları öncelikle veri önişleme adımından geçirilir. İşlenmiş veri üzerinden sözlüksel ve anlamsal öznitelikler, sözcük öbekleri arasındaki benzerliği hesaplamak için çıkarılmaktadır. Kategori özniteliği çıkarımında kullanılmak üzere Dexter aracına DBPedia kategori hiyerarşisi entegre edilir. Dexter aracılığıyla çıkartılan anlamsal öznitelikler, N-gram ve Word2vec yöntemleri kullanılarak vektörel hale dönüştürülür. Sayısal ve sözel özniteliklere sahip sözcük öbeklerinin benzerlik hesaplamasında kosinüs benzerliği kullanılır [14]. Bu işlemlerin ardından benzer sorguları kümelemek için K-Medoids [15] ve DB-Scan [16] kümeleme algoritmaları kullanılır. Elde edilen her bir sorgu kümesi bir göreve aittir. Arama sorguları üzerinden görev tabanlı kümeleme yapmak üzere geliştirilen yönteme ait akışı gösteren sistem genel blok şeması Şekil 5.1'deki gibidir.



Şekil 5.1 : Sistemin genel blok şeması.

5.1 Veri Seti Özellikleri

Bu çalışmada 2006 America Online (AOL) [18] veri seti kullanılmıştır. Veri seti, 2006 yılının 3 ayı içerisinde AOL arama motoruna yaklaşık 650 bin kullanıcı tarafından girilen 21 milyon sorgu içerir. Veri seti içeriği Şekil 5.2'de gösterilmektedir. AnonID kolonunda sorguyu giren kullanıcı bilgisi tutulmaktadır. Query kolonu sorgu içeriğini, QueryTime kolonu sorgunun girildiği tarih ve saat bilgisini tutmaktadır. Eğer bu sorgu sonucunda döndürülen linklerden biri tıklanmışsa, tıklanan linkin URL bilgisi ClickURL kolonunda ve bu linkin döndürülme sırası ise ItemRank kolonunda tutulmaktadır. Sorgu sonucu döndürülen linklerden herhangi biri tıklanmamışsa ClickURL ve ItemRank kolonları ilgili sorgu satırı için boş gelir.

AnonID	Query	QueryTime	ItemRank	ClickURL
144	car rentals	2006-03-01 18:22:49		
144	mail consort	2006-03-02 14:02:49		
144	www.charlottcountystadium.com	2006-03-03 09:39:03		
144	when is opening day for ticket sales in boston	2006-03-28 17:50:34		
144	www.herbchambers.com	2006-04-23 09:23:48	3	http://www.carsearch.com
144	www.herbchambers.com	2006-04-23 09:23:48	1	http://www.herbchambers.com
144	www.eastern bank.com	2006-05-10 10:55:50	1	http://www.easternbank.com
227	psychiatric disorders	2006-03-02 17:30:36	1	http://www.merck.com
227	psychiatric disorders	2006-03-02 17:30:36	3	http://allpsych.com

Şekil 5.2 : Veri seti içeriği.

Değerlendirme veri seti olarak Lucchese ve arkadaşlarının yaklaşık 3000 satırdan oluşan test veri seti kullanılmıştır [3]. Veri setinde 18 farklı kullanıcı tarafından girilmiş, 997 tekil sorgu vardır. Sorgulardan 852 adet öğe, 3959 adet kategori bilgisi elde edilmiştir. Toplamda, 26 dakikalık zaman boşluklarına göre ayrılan 624 oturum ve arama sonucu tıklanmış 986 adet link bulunmaktadır.

5.2 Veri İşleme Adımları

Öncelikle veri seti önişleme aşamasından geçirilerek gürültülü veriler veri setinden çıkarılır. Temizlenmiş veriler üzerinden çeşitli öznitelik vektörleri elde ediler. Sorgular arasındaki benzerlik hesaplamalarında elde edilen bu öznitelik vektörleri çeşitli kombinasyonlarda bir araya getirilerek kullanılır. Benzerlik hesaplamasında kosinüs benzerliği kullanılmaktadır.

5.2.1 Veri önişleme

Öncelikle, sadece noktalama işaretlerinden oluşan sorgular veri setinden çıkarılmıştır. Doğal dil işlemede, anlam taşısa dahi semantik çözümlemede hiç bir bilgi

üretmeyen ve ilgili dilde yaygın olarak kullanılan etkisiz sözcükler sorgu geçmişinden çıkarılmıştır (örn. İngilizce'de: and, or, the, then, also, after, vb.). Bu işlemler sonucu oluşan boş sorgular veri setinden silinmiştir.

5.2.2 Kullanılan öznitelik vektörleri

Her bir sorgunun sözlüksel özniteliklerinin vektörel gösteriminde aşağıdaki öznitelik vektörleri hesaplanmıştır.

5.2.2.1 URL vektörü

Tüm sorgular için her bir URL'in kaç kez tıklandığı bilgisini gösteren öznitelik vektörüdür. Veri setinde, sorgu girildiğinde kullanıcının tıkladığı URL bilgileri bulunmaktadır. Tüm veri seti taranarak tekilleştirilmiş URL listesi oluşturulur ve her bir sorgu için ilgili URL'in tıklanma sayısı saklanır. Eğer sorgu cümleciği girildiğinde ilgili URL'e tıklanmadıysa 0 değeri atanır. Elde edilen $n \times u$ boyutlu vektör URL vektörüdür. n, sorgu sayısını; u, URL sayısını gösterir.

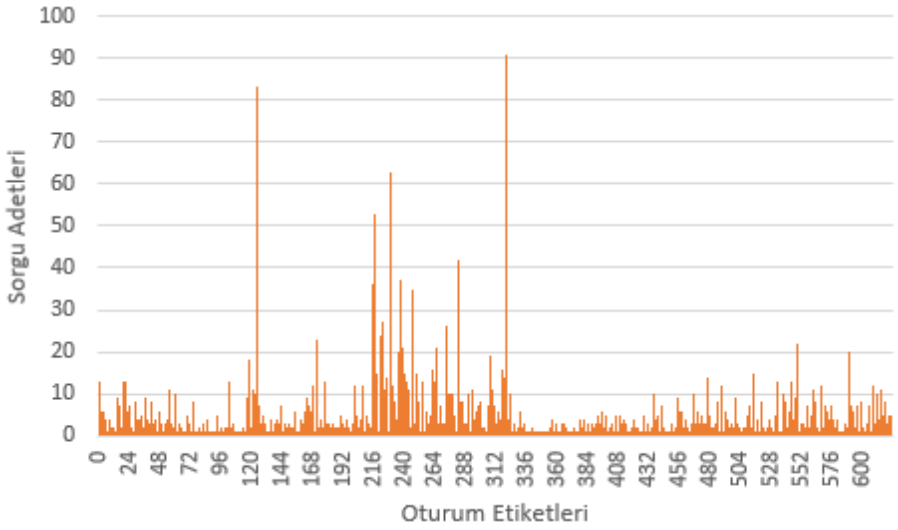
5.2.2.2 Kullanıcı vektörü

Tüm sorgular için her bir kullanıcının ilgili sorguyu kaç kez girdiği bilgisini gösteren öznitelik vektörüdür. Veri setinde, sorguları giren kullanıcı numaraları (anonID) bilgisi bulunmaktadır. Tüm veri seti taranarak tekilleştirilmiş kullanıcı listesi oluşturulur ve her bir sorgu için ilgili kullanıcının sorguyu kaç kez girdiği sayısı saklanır. Eğer sorgu cümleciği ilgili kullanıcı tarafından girilmediyse 0 değeri atanır. Elde edilen $n \times k$ boyutlu vektör kullanıcı vektörüdür. n, sorgu sayısını; k, kullanıcı sayısını gösterir.

5.2.2.3 Oturum vektörü

Aynı veri seti üzerinde oturum bazlı kümeleme çalışması yapan Lucchese ve arkadaşları en ideal oturum süresini 26 dakika olarak tespit etmiştir [3]. Çalışmamızda sorgular zamana göre sıralanıp iki sorgu arasındaki zaman farkı 26 dakikayı aşıyorsa farklı oturum kabul edilerek oturum etiketleri üretilmiştir. Her bir oturumda ilgili sorgunun kaç kez girildiğinin bilgisi öznitelik vektörünü oluşturur. Tüm veri seti taranarak tekilleştirilmiş oturum listesi oluşturulur ve her bir sorgu için ilgili oturumda sorgunun kaç kez girildiği sayısı saklanır. Eğer sorgu cümleciği ilgili oturumda hiç girilmediyse 0 değeri atanır. Elde edilen $n \times o$ boyutlu vektör oturum vektörüdür. n,

sorgu sayısını; *o*, oturum sayısını gösterir. Şekil 5.3'de değerlendirme veri setinde etiketlenen 624 oturuma düşen sorgu adetleri gösterilmektedir.



Şekil 5.3 : Oturum sorgu adetleri.

Her bir sorgunun anlamsal özniteliklerinin belirlenmesinde öğe ve kategori bilgilerinden faydalanılmıştır. Örneğin, "cherry or grape" sorgusunun öğeleri "cherry" ve "grape" olarak belirlenir. Bu öğelerin kategori bilgisi "fruit" olarak bulunur. Öğe ve kategori bilgileri sayısal gösterimde olmadığı için bunlar arasındaki kosinüs benzerliği, N-gram ya da Word2vec yöntemleriyle vektörel hale getirildikten sonra hesaplanır.

5.2.2.4 Öğe vektörü

Dexter (bakınız Bölüm 2.3) ile her bir sorgunun öğeleri etiketlenir. Bir sorgu içerisinde birden çok öğe olabileceği gibi hiç bir öğe de bulunamayabilir. Bu sebeple, deneylerimizde öğe vektörünün özel arama kelimelerinin kullanıldığı sorguların kümelenmesinde başarımı arttırırken, genel aramalarda başarımı etkilemediği gözlemlenmiştir. Bir sorgu içinde birden fazla öğe bulunabilir. Bu durumda, Word2vec yöntemi kullanıldığı senaryolarda her bir öğenin vektörü ayrı ayrı elde edildikten sonra skalar toplama işlemi yapılarak sorgunun öğe vektörü üretilir.

5.2.2.5 Kategori vektörü

Dexter (bakınız Bölüm 2.3) ile öğe bilgilerinin hangi kategoriye ait olduğu bilgisi Wikipedia makalelerini kullanan DBPedia kütüphanesinden yararlanılarak elde edilir. Bir öğe birden fazla kategoriye dahil olabilmektedir ve her kategorinin içinde birden fazla öğe bulunur. Bir öğenin birden fazla kategorisi olduğu durumda, Word2vec

yöntemi kullanıldığı senaryolarda her bir kategorinin vektörü ayrı ayrı elde edildikten sonra skalar toplama işlemi yapılarak sorgunun kategori vektörü üretilir.

5.2.3 Benzerlik hesaplama

Kümeleme adımında kullanılmak üzere öznitelikler arasındaki benzerlik hesaplaması kosinüs benzerliği ile yapılmıştır. URL, oturum ve kullanıcı vektörleri tek bir vektör olacak şekilde peş peşe eklenerek birleştirilip ([URL, Oturum, Kullanıcı]) veri setindeki sorgular arasında sözlüksel öznitelikler arası kosinüs benzerliği hesaplanmaktadır. $n \times u$ boyutlu URL vektörü, $n \times o$ boyutlu oturum vektörü, $n \times k$ boyutlu kullanıcı vektörü peş peşe eklendiğinde $n \times (u + o + k)$ boyutlu öznitelik vektörü oluşur. n, sorgu sayısını; u, URL sayısını; o, oturum sayısını; k, kullanıcı sayısını göstermektedir.

Sorgulardan elde edilen öğeler arasındaki benzerliğin hesaplanmasında N-gram yöntemi ile elde edilen dizgiler arasındaki kosinüs benzerliği hesaplanmıştır. Deneylerimizde bu yöntem için en iyi sonuç $N = 3$ alındığında elde edilmiştir. Bunun yanı sıra, öğelerin benzerlikleri Google News [13] modeli üzerinden sinir ağları öğrenmesi ile sözcük vektörlerini üreten Word2vec yöntemi kullanılarak da ayrıca ölçülmüştür. Elde edilen beş farklı öznitelik vektörü eşit ağırlıklandırılarak ($w = 0.2$) kosinüs benzerlik matrisi elde edilir.

Karşılaştırma yapmak üzere benimsediğimiz diğer bir yaklaşımımızda ise, Word2vec yöntemini sorgularda geçen her bir kelimeye uygulayarak sorgular arasındaki benzerlikler hesaplanmıştır. Burada sorgu metinleri boşluk (" ") karakterine göre kelimelere ayrılmıştır. Her bir kelimenin vektörel karşılığı Google News bütüncesini kullanan word2vec yöntemiyle elde edilmiştir. Bütüncede bulunmayan kelimeler sonuca etki etmemesi için etkisiz vektör (sıfır vektörü) olarak değerlendirilmiştir. Denklem 5.1'de bir sorguda birden fazla kelime vektörü bulunduğu durumlarda, q sorgusuna ait sorgu vektörünün $\vec{V}(q)$, bu sorguda bulunan k kelimelerine ait kelime vektörlerinin $\vec{V}(k)$ skalar toplamından hesaplanma formülü gösterilmektedir. Sonuç olarak, oluşturulan sorgu vektörleri arasındaki kosinüs benzerliğinden kümeleme işleminde kullanılmak üzere benzerlik matrisi elde edilir.

$$\vec{V}(q) = \sum_{k \varepsilon q} \vec{V}(k) \tag{5.1}$$

5.3 Model Eğitimi ve Sınanması

Oluşturulan öznitelik setinin özellikleri göz önüne alınarak seçilen merkez tabanlı kümeleme yöntemi K-Medoids algoritması ile yoğunluk tabanlı kümeleme yapan DB-Scan algoritması karşılaştırılmak üzere kullanılmıştır.

Bu çalışmada öncelikle farklı öznitelik vektörlerinin kümeleme performansı üzerindeki etkisi test edilmiştir. Kümeleme başarımı, ortalama uzaklık hesaplamaları üzerinden elde edilen ayrışma ve sıklık değerleri ile ölçülmektedir. Tablolarda belirtilen öznitelik vektörlerinin kombinasyonları aşağıda sunulmuştur.

- URL-Kullanıcı-Oturum: URL, kullanıcı ve oturum vektörlerinin birleştirilmesi sonucu elde edilmiştir.
- Öğe-Kategori (n-gram): Öğe ve kategori bilgilerinin trigram yöntemi ile oluşturulan vektörleri üzerinden elde edilmiştir.
- Öğe-Kategori (word2vec): Öğe ve kategori bilgilerinin Word2vec yöntemi ile oluşturulan vektörleri üzerinden elde edilmiştir.
- Tüm Öznitelikler (n-gram): Tüm özniteliklerin (URL, kullanıcı, oturum, öğe ve kategori) eşit ağırlıklandırılarak birleştirilmesi ile elde edilmiştir. Anlamsal özniteliklerin benzerliğinde trigram yöntemi kullanılmıştır.
- Tüm Öznitelikler (word2vec): Tüm özniteliklerin eşit katsayı ile çarpılarak birleştirilmesi sonucu elde edilmiştir. Anlamsal özniteliklerin vektörel hale getirilmesinde Word2vec yöntemi kullanılmıştır.
- word2vec: Sorguların direkt Word2vec yöntemi ile oluşturulan vektörleri üzerinden elde edilmiştir.

Sorguların görev çıkarımı için kümelenmesinde kullanılan, merkez tabanlı K-Medoids algoritmasının test veri seti üzerinde farklı küme sayıları için çalıştırılması ile elde edilen sonuçlar Çizelge 5.1'de sunulmuştur. K-Medoids algoritmasında

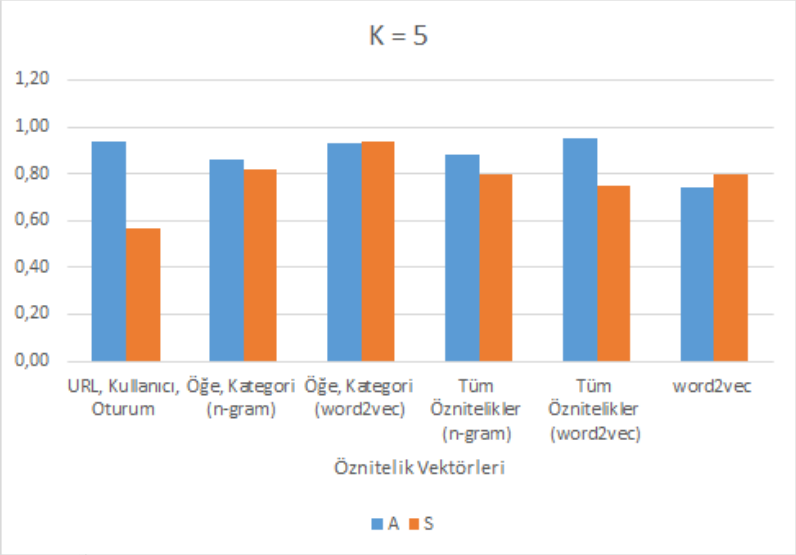
oluşacak küme sayısı önden parametre olarak verilir. Bu küme sayıları DB-Scan algoritması sonucu oluşan küme sayılarına göre belirlenmiştir. K-Medoids kümeleme algoritmasında, en iyi ayrışma ve sıklık değerleri 20 küme ($K = 20$) üzerinden yapılan hesaplamalarda elde edilmiştir. En iyi ayrışma Öğe-Kategori (word2vec) öznitelik vektörü kullanıldığında, en iyi sıklık ise, URL-Kullanıcı-Oturum öznitelik vektörü kullanıldığında gözlemlenmiştir.

Çizelge 5.1 : K-Medoids değerlendirme sonuçları.

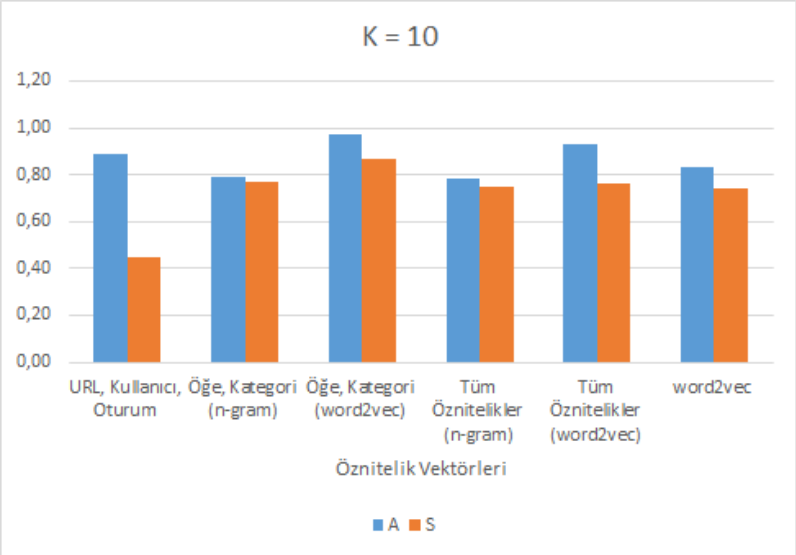
Öznitelik Vektörleri	Küme Sayısı (K)					
	5		10		20	
	A	S	A	S	A	S
URL-Kullanıcı-Oturum	0.94	0.57	0.89	0.45	0.91	0.41
Öğe-Kategori (n-gram)	0.86	0.82	0.79	0.77	0.83	0.71
Öğe-Kategori (word2vec)	0.93	0.94	0.97	0.87	0.98	0.82
Tüm Öznitelikler (n-gram)	0.88	0.80	0.78	0.75	0.86	0.63
Tüm Öznitelikler (word2vec)	0.95	0.75	0.93	0.76	0.91	0.68
word2vec	0.74	0.80	0.83	0.74	0.81	0.72

K-Medoids algoritmasında genel olarak küme içi başarımı ölçen sıklık değerlerinin oldukça yüksek olduğu K-Medoids değerlendirme sonuçları şekillerinde görülmektedir. K-Medoids algoritmasının $K = 5$ için çalıştırılmasıyla elde edilen sonuçlar Şekil 5.4'de, $K = 10$ için çalıştırılmasıyla elde edilen sonuçlar Şekil 5.5'de, $K = 20$ için çalıştırılmasıyla elde edilen sonuçlar Şekil 5.6'da gösterilmiştir. Merkez tabanlı yöntemlerle yapılan kümelemelerde veri setindeki gürültülerin küme başarımına etkisi deneylerimizde de görülmüştür. Sorgulara doğrudan Word2vec yönteminin uygulandığı kümelemelerde diğer özniteliklere oranla daha başarısız sonuçlar elde edilmiştir.

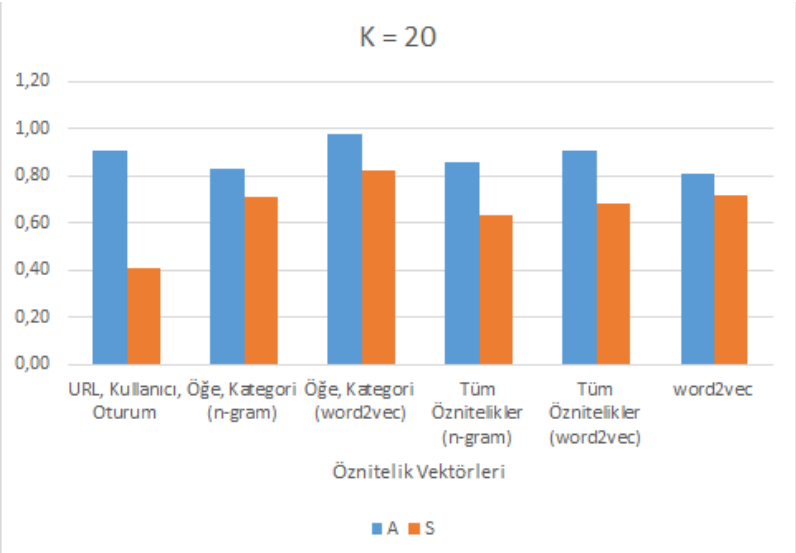
Sorguların kümelenmesinde kullanılan yoğunluk tabanlı kümeleme yöntemi olan DB-Scan algoritması test veri seti üzerinde çalıştırılmıştır. DB-Scan algoritmasında bir kümede bulunması gereken minimum nokta sayısı ve iki nokta arasındaki maksimum uzaklık değerleri parametreleri kullanılmaktadır. Deneylerimizde en iyi sonuçlar minimum 5 nokta parametresi ile elde edilmiştir. Çizelge 5.2'de farklı maksimum uzaklık değerleri için DB-Scan algoritması tarafından oluşturulan küme sayıları gösterilmektedir. Noktalar arası maksimum uzaklık arttırıldığında tüm sorguların tek bir kümede toplandığı gözlenmektedir.



Şekil 5.4 : K-Medoids değerlendirme sonuçları (a).



Şekil 5.5 : K-Medoids değerlendirme sonuçları (b).



Şekil 5.6 : K-Medoids değerlendirme sonuçları (c).

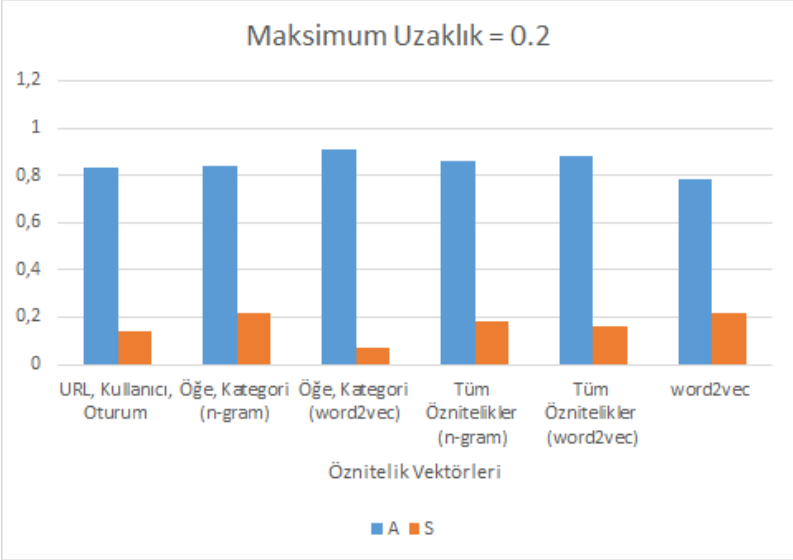
Çizelge 5.2 : DB-Scan tarafından oluşturulan küme sayıları.

Öznitelik Vektörleri	Maksimum Uzaklık Değerleri		
	0.2	0.4	0.6
URL-Kullanıcı-Oturum	45	18	15
Öğe-Kategori (n-gram)	13	11	1
Öğe-Kategori (word2vec)	9	10	8
Tüm Öznitelikler (n-gram)	6	24	21
Tüm Öznitelikler (word2vec)	13	25	11
word2vec	125	13	1

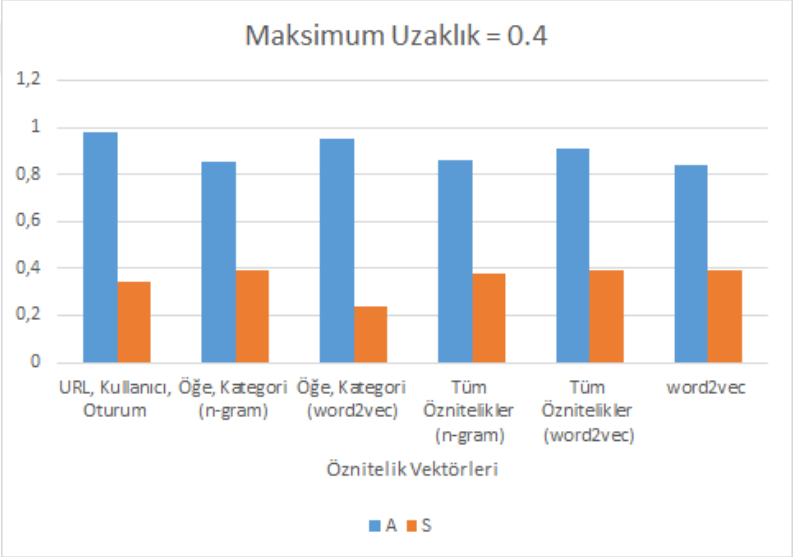
DB-Scan algoritmasının test veri seti üzerinde minimum 5 nokta ve farklı maksimum uzaklık değerleri parametreleri için çalıştırılması ile elde edilen sonuçlar Çizelge 5.3'de gösterilmektedir. Noktalar arası maksimum uzaklık arttırıldığında tüm sorguların tek bir kümede toplanması, kümeler arası mesafe hesaplamasına dayanan ayrışma değerinin ölçülememesine neden olmaktadır. Bu sonuçlar tabloda "-" ile gösterilmiştir. Genel olarak DB-Scan kümeleme algoritmasında, minimum 5 elemanlı kümeler için en iyi sıklık değerleri maksimum uzaklık 0.2 alındığında, en iyi ayrışma değerleri maksimum uzaklık 0.4 alındığında gözlemlenmiştir. Noktalar arası maksimum uzaklık değeri arttırıldığında sıklık değerinin de arttığı, dolayısıyla başarımın düştüğü DB-Scan değerlendirme sonuçları şekillerinde görülmektedir. DB-Scan algoritmasının maksimum uzaklık 0.2 için çalıştırılmasıyla elde edilen sonuçlar Şekil 5.7'de, maksimum uzaklık 0.4 için çalıştırılmasıyla elde edilen sonuçlar Şekil 5.8'de, maksimum uzaklık 0.6 için çalıştırılmasıyla elde edilen sonuçlar Şekil 5.9'da gösterilmiştir. Bu bilgiler ışığında, maksimum uzaklık 0.2 alındığında en iyi sonuçların Öğe-Kategori (word2vec) öznitelik vektörü kullanılarak elde edildiği söylenebilir.

Çizelge 5.3 : DB-Scan değerlendirme sonuçları.

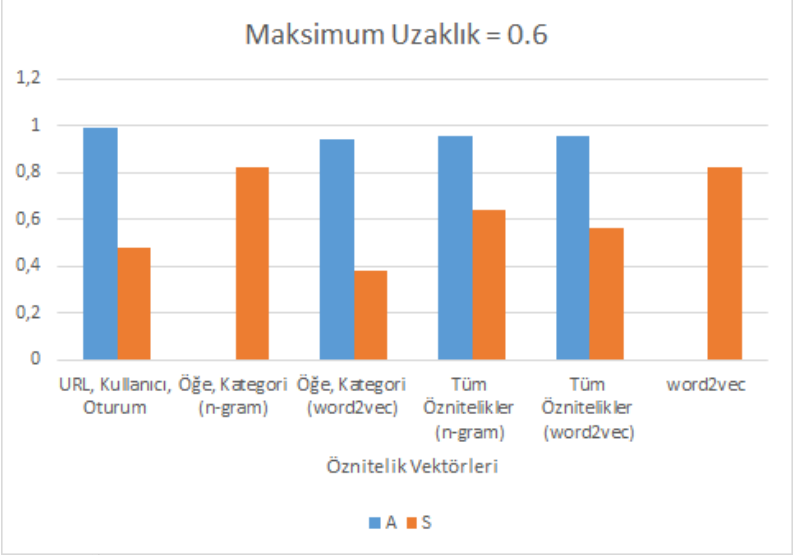
Öznitelik Vektörleri	Maksimum Uzaklık Değerleri					
	0.2		0.4		0.6	
	A	S	A	S	A	S
URL-Kullanıcı-Oturum	0.83	0.14	0.98	0.34	0.99	0.48
Öğe-Kategori (n-gram)	0.84	0.22	0.85	0.39	-	0.82
Öğe-Kategori (word2vec)	0.91	0.07	0.95	0.24	0.94	0.38
Tüm Öznitelikler (n-gram)	0.86	0.18	0.86	0.38	0.96	0.64
Tüm Öznitelikler (word2vec)	0.88	0.16	0.91	0.39	0.96	0.56
word2vec	0.78	0.22	0.84	0.39	-	0.82



Şekil 5.7 : DB-Scan değerlendirme sonuçları (a).



Şekil 5.8 : DB-Scan değerlendirme sonuçları (b).



Şekil 5.9 : DB-Scan değerlendirme sonuçları (c).

Karşılaştırma tablolarının en iyi sonuçları ele alındığında DB-Scan algoritmasının K-Medoids algoritmasına göre daha başarılı sonuçlar elde ettiği görülmektedir. Yoğunluk bazlı kümeleme yöntemlerinin küme sayılarını veri setine göre belirlemesi, küme sayısının önceden kesinleştirilemeyeceği metin tabanlı çalışmalarda tercih sebebidir. Elde edilen sonuçlar da bunu doğrulamaktadır.

6. SONUÇ VE ÖNERİLER

Arama motorları veya e-ticaret siteleri üzerinde bir kullanıcı tarafından belirli bir süre içinde gerçekleştirilen aramalar birden fazla amaca yönelik olabilir. Bu farklı amaçların her biri o kullanıcının gerçekleştirmek istediği bir görev olarak adlandırılmaktadır. Görevlerin belirlenmesi özellikle kullanıcının amacına yönelik sonuçların geri getirilmesinde önem taşımaktadır.

Bu tez kapsamında büyük verilerin toplandığı arama motorlarına girilen sorgularda görev çıkarımları üzerinde bir çalışma yapılmıştır. Görev çıkarımları için, kullanıcı bilgileri, oturum bilgileri, sorgu metinleri, sorgu metinlerinin kategorisi gibi farklı bilgiler kullanılarak arama metinleri öznitelik vektörleri olarak gösterilmiştir. Deneysel çalışmalar bu alanda çokça kullanılan AOL veri kümesi üzerinde gerçekleştirilmiştir. Bu kapsamda URL, kullanıcı, oturum öznitelikleri ile sorgular üzerinden Word2vec yöntemiyle sadece anlamsal öznitelik üretilmesi yaklaşımlarının görev çıkarımına olan etkileri incelenmiştir. Görev çıkarımında öğe ve kategori anlamsal öznitelikleri kullanıldığında daha geniş içerikli görevler oluşturulduğu deneyimlenmiştir. Tüm öznitelikler dikkate alındığında öğe ve kategori benzerliklerinin belirlenmesinde Word2vec kullanımı görev çıkarımı başarımını olumlu yönde etkilemektedir.

Bu çalışmadan elde edilen çıktılar görev çıkarımlarından faydalanan öneri sistemlerinin geliştirilmesinde, farklı yazılı iletişim kanallarında metin tamamlama işlemlerinde kullanılabilecektir.

KAYNAKLAR

[1] **Url-1**, `https://algorithmicthoughts.wordpress.com/2013/05/29/machine-learning-dbscan/`, alındığı tarih: 01.05.2018.

[2] **Url-2**, `http://dexter.isti.cnr.it/`, alındığı tarih: 22.04.2018.

[3] **Lucchese, C.**, **Orlando, S.**, **Perego, R.**, **Silvestri, F. ve Tolomei, G.** (2011). Identifying task-based sessions in search engine query logs, *Proceedings of the fourth ACM international conference on Web search and data mining*, ACM, s.277–286.

[4] **Hua, W.**, **Song, Y.**, **Wang, H. ve Zhou, X.** (2013). Identifying users' topical tasks in web search, *Proceedings of the sixth ACM international conference on Web search and data mining*, ACM, s.93–102.

[5] **Verma, M. ve Yilmaz, E.** (2014). Entity oriented task extraction from query logs, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, ACM, s.1975–1978.

[6] **M.Verma ve Yilmaz, E.** (2016). Category Oriented Task Extraction, *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, ACM, s.333–336.

[7] **Yuan, Y.**, **He, L.**, **Peng, L. ve Huang, Z.** (2014). A new study based on word2vec and cluster for document categorization, *Journal of Computational Information Systems*, *10*(21), 9301–9308.

[8] **Jansen, B.J.** (2006). Search log analysis: What it is, what's been done, how to do it, *Library & information science research*, *28*(3), 407–432.

[9] **Silverstein, C.**, **Marais, H.**, **Henzinger, M. ve Moricz, M.** (1999). Analysis of a very large web search engine query log, *ACm SIGIR Forum*, cilt 33, ACM, s.6–12.

[10] **Ceccarelli, D.**, **Lucchese, C.**, **Orlando, S.**, **Perego, R. ve Trani, S.** (2013). Dexter: an open source framework for entity linking, *Proceedings of the sixth international workshop on Exploiting semantic annotations in information retrieval*, ACM, s.17–20.

[11] **Sperr, H.**, **Niehues, J. ve Waibel, A.** (2013). Letter n-gram-based input encoding for continuous space language models, *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, s.30–39.

[12] **Ling, W.**, **Dyer, C.**, **Black, A.W. ve Trancoso, I.** (2015). Two/too simple adaptations of word2vec for syntax problems, *Proceedings of the 2015*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, s.1299–1304.

[13] **Google**, (2013), word2vec, `https://code.google.com/archive/p/word2vec/`, erişim 14 Şubat 2018.

[14] **Huang, A.** (2008). Similarity measures for text document clustering, *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, s.49–56.

[15] **Bhat, A.** (2014). K-medoids clustering using partitioning around medoids for performing face recognition, *International Journal of Soft Computing, Mathematics and Control*, *3*(3), 1–12.

[16] **Kriegel, H.P. ve Pfeifle, M.** (2005). Density-based clustering of uncertain data, *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, ACM, s.672–677.

[17] **He, J.**, **Tan, A.H.**, **Tan, C.L. ve Sung, S.Y.**, (2004). On quantitative evaluation of clustering systems, Clustering and information retrieval, Springer, s.105–133.

[18] **Pass, G.**, **Chowdhury, A. ve Torgeson, C.** (2006). A picture of search., *InfoScale*, cilt152, s. 1.

ÖZGEÇMİŞ

Ad Soyad: Almıla Selcen AKGÜN

Doğum Tarihi ve Yeri: 12.01.1990 / AKSARAY

E-Posta: akarabatak@itu.edu.tr

ÖĞRENİM DURUMU:

- **Lisans:** 2011, Sakarya Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği.
- **Y. Lisans:** 2018, İstanbul Teknik Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Programı.

MESLEKİ DENEYİMLER VE ÖDÜLLER:

- 2011 yılında Sakarya Üniversitesi'nde lisansını üniversite birinciliği ile tamamladı.

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- Akgün, A.S., Yaslan, Y., 2018. Task-based Clustering On Search Queries. *Signal Processing and Communications Applications Conference (SIU), 2018 26th. IEEE, 2018.*