International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

Use of NLP Techniques for an Enhanced Mobile Personal Assistant: The Case of Turkish [1]

Gokhan Celikkaya[1], Gulsen Eryigit*[1]

Accepted : 05/06/2017 Published: 30/09/2017

Abstract: This article introduces a Turkish mobile assistant application which produces state-of-the art results for the Turkish language by using natural language processing (NLP) techniques. The voice-enabled mobile assistant application allows users to enter queries for nine pre-defined tasks; namely, making calls, sending sms messages and emails, getting directions, querying exchange rates, weather forecast and traffic information, searching on the internet and launching applications on the phone. Users' queries are processed in a multi-stage approach (viz., NLP, query classification and parameter extraction). Either the requested task is performed or the requested information is displayed as the response of the application. The article presents the architecture of the introduced system, its comparison with some prominent mobile assistants as well as the newly created data resources (viz., two query datasets annotated for classification and parameter extraction, two specific datasets for domain adaptation of named entity recognition and syntactic parsing NLP modules) to be used in further research. The evaluations on the impact of NLP preprocessing layers to the query classification performances reveal that the added value by NLP may range from 0.2 to 10.7 percentage points depending on the preferred machine learning algorithm for the query classification stage. The impact of NLP for the parameter extraction stage is also crucial since the outputs of NLP modules are used systematically by the extraction rules. The overall performance of the introduced approach is measured as 70.8% accuracy score which is very promising under the fact that the system is trained with very limited-size of annotated data. The technology introduced in this article is basically designed for the case of a mobile assistant but it can also be used for every voice-enabled control system to improve the user experience, such as smart homes or smart televisions.

Keywords: *natural language processing, NLP, question answer system, mobile assistant, machine learning.*

1. Introduction

Since the interaction with mobile devices increased enormously in recent years, user friendly interfaces facilitating human-machine interaction became very important. Natural language is the most natural way of communicating for humans and this explains the popularity of the mobile assistant applications. Unfortunately, the challenges caused by the typological differences of some natural languages also appear in mobile applications and the generated systems which are mostly on English could not be adapted easily to other languages such as Turkish. SIRI [2] and GoogleNow [3] are the most popular applications on this topic and there are many more. Even though SIRI has begun to support Turkish after IOS 8.3 update, its quality and coverage for different supported services are not as good as for the English version. Moreover, the other popular applications such as GoogleNow [3], Cortana [4], Robin [5], Assistant [6] do not support the Turkish language at all.

There exist few other mobile assistant applications developed for Turkish such as AsistanB [7] and CEYD-A [8], but these are mostly working on a keyword-match or phrase-match basis and do not make use of any NLP solutions. Therefore, their success rates are very low. For example, for a query such as "Bugün hava Ankara'da nasıl olacak?" (*How is the weather today in Ankara*), such applications return the weather forecast of the current location (e.g. Istanbul) instead of the requested location.

Turkish is an agglutinative language with a very rich morphological structure. This property of the language increases the need to use underlying sophisticated natural language processing tools in order to correctly process a user request in a mobile assistant application. In recent years, the accessibility of basic NLP tools for Turkish (such as text normalizers, morphological and syntactic analyzers [9], [10], [11], [12], [13], [14], [15], [16]) makes their use possible in higher level applications. The technology that we introduce in this article is basically designed for the case of a mobile assistant but it can also be used for every voice-enabled control system to improve the user experience, such as smart homes or smart televisions.

This article presents the first academic study focusing on the language related problems of a Turkish Mobile Assistant. The voice-enabled mobile assistant application allows users to enter queries for nine pre-defined tasks; namely, making calls, sending sms messages and emails, getting (driving or walking) directions, querying exchange rates, weather forecast and traffic information, searching on the internet and launching applications on the phone. Users' queries are processed in a multi-stage approach and either the requested task is performed or the requested information is displayed as the response of the application. The multi-stage approach basically consists of NLP of the input queries, their classification into 9 distinct target classes by the use of a hybrid

[1] *Dep. of Computer Eng. Istanbul Technical University Istanbul, Turkey*
Corresponding Author: Email: gulsen.cebiroglu@itu.edu.tr

classification approach, their parameter extraction by rule-based task specific modules and finally the generation of the assistant's response.

This article, which is an extended version of a conference paper [1], presents the architecture of the introduced system, its comparison with some prominent mobile assistants as well as the newly created data resources in order to be used in further research. The developed data resources are two query datasets annotated separately for the classification and parameter extraction stages, two specific datasets for domain adaptation of named entity recognition and syntactic parsing NLP modules. The evaluations on the impact of NLP preprocessing layers on the query classification performances reveal that the added value by NLP may range from 0.2 (for support vector machines) to 10.7 (for Bayesian networks) percentage points depending on the preferred machine learning algorithm for the query classification stage. The impact of NLP for the parameter extraction stage is also crucial since the outputs of NLP modules are used systematically by the extraction rules. The overall performance of the introduced approach is measured as 70.8% accuracy score which is very promising under the fact that the system is trained with very limited-size of annotated data.

The article is structured as follows: Section 2 provides the related works, Section 3 gives a brief information about Turkish language's challenging characteristics related to our domain, Section 4 gives information about the newly introduced language resources, Section 5 presents the system architecture; the mobile side interfaces (Section 5.1) and the server-side application (Section 5.2): NLP layer and domain adaptation (5.2.1), query classification (5.2.2) and parameter extraction (5.2.3). Section 6 gives the conclusion and future work.

2. RELATED WORK

Mobil assistant applications enable users to use their mobile phones and tablets via natural language and have become popular in recent years. These applications provide relevant responses according to users' query by using NLP solutions. In the literature, there exist many applications (SIRI [2], GoogleNow [3], Cortana [4], Robin [5], Assistant [6]), publications ([17], [18], [19], [20], [21]) and patents ([22], [23], [24], [25]) related to the English language. Apple's SIRI and Google Now applications are the most popular ones in this domain. On the other hand, the studies for Turkish are very limited in number and these are mostly inadequate in terms of the number of supported services, usage of NLP technology and success rate compared to the ones built for English. Turkcell Mobil Asistan [26] has the most widely used local application for Turkish. Although it supports services such as weather, exchange, and call, it is mainly developed to query company's own services. Beyond that, AsistanB [7] and CEYD-A [8] are also commonly used local applications for Turkish. In general, these applications were developed by rule based methods and lack of NLP technology. CEYD-A uses a basic learning mechanism that allows its users to define new rules. İdris [27] application was developed with many deficiencies yielding to its withdrawn from mobile markets. Since these applications were developed with rule based approaches, queries must be asked matching pre-defined templates. For example, weather forecast information needs to be asked like "Ankara hava durumu" (*Ankara weather forecast*). These are unable to respond correctly to a natural question such as "Hava yarın Ankara'da nasıl olacak?" (*How is the weather like in Ankara tomorrow*?)".

Beyond the above mentioned local applications, with the IOS 8.3 update released in April 2015, SIRI application has begun to support the Turkish language as well. SIRI uses some natural language processing components such as a pos-tagger and a named entity recognizer and therefore has better performance compared to the other applications developed for Turkish.

3. TURKISH & ITS CHALLENGES

Turkish is an agglutinative language and has a very rich morphological structure. Since most of the syntactic information appear at word level, Turkish comes out to be a free constituent order language where the word order is very flexible compared to fixed word order languages. These properties of the language differentiate it from many languages for the automated language processing applications. There exist many studies in the literature which focus on specific and challenging features of the Turkish language for this domain. In this section, we briefly detail its outstanding properties which deserve attention for the mobile application domain and explain the benefited NLP layers.

Since most of the NLP tools are originally developed for English, their performances on Turkish are generally not as good as for English even if the tool in focus claims multilingual support. An example to this may be seen in speech recognition applications. Although today's speech recognizers are quite successful at understanding the spoken inputs, it is quite common that their outputs do not conform well to the written language formal rules; e.g. the use of capital letters at the beginning of proper names (newyork vs. New York), the writing of postpositions, prepositions and enclitics, and their differentiation from agglutinated inflectional suffixes in case of morphologically rich languages. The erroneous outputs of these speech recognizers may not be correctly processed by following NLP modules (such as a morphological analyzer or syntactic parser) which are generally developed to process well-written text. As a consequence, the speech recognizers' outputs need to be normalized by a text normalization module. Table 1 provides an example query ('Yarın Ankara'da hava nasıl' – *What will be the weather like in Ankara tomorrow*). The original output from a speech recognizer is given in the first line of the table where the text contains neither any capital letter nor punctuation mark. The locative case marker (-da) is also erroneously written separately from the location name (Ankara). The normalized query is given in the second line of the table.

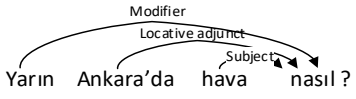
Table 1 also provides the output for the input query from different NLP layers of a classical NLP pipeline: a text normalizer, a tokenizer, morphological analysis and disambiguation, named entity recognition and syntactic parsing. A brief explanation of these NLP layers is provided below. The reader may refer to [16] for further references.

- **Tokenization** is the process of splitting the input into tokens.
- **Text Normalization** is the process of transforming text into a canonical form.
- **Morphological Analysis** is the process of analyzing a word: finding its lemma, parts-of-speech tag and inflectional features as well as its derivational structure.
- **Morphological Disambiguation** is the process of selecting the most probable morphological analysis sequence for an input word sequence.
- **Named Entity Recognition** is the process of identifying the named entities (e.g. person, location or organization names)

within the input word sequence.

- **Dependency Parsing** is the process of constructing a syntactic dependency tree to a given input sentence.

Table 1. An example query and its analysis by NLP modules

Query (output from a speech recognizer [28])	yarın ankara da hava nasıl
Normalized Query	Yarın Ankara'da hava nasıl ? *Tomorrow - in Ankara – weather – how ?*
Morphological Analysis & Disambiguation	Yarın yarın+Adverb Ankara'da Ankara+Noun+Prop+A3sg+Pnon+Loc hava hava+Noun+A3sg+Pnon+Nom nasıl nasıl+Adverb ? ?+Punc
Named Entity Recognition	[TIME] [LOCATION] hava nasıl ?
Syntactic Dependency Parsing	

For a free constituent order language such as Turkish, writing fixed patterns to match query inputs would reveal unsuccessful results since the queries would most of the time not conform to these. For example, all the following sentences and many more, with same constituents but in different order, should match to the same pattern with the English query "*What will be the weather like in Ankara tomorrow?".* Dependency parsing approach, which became a very popular parsing technique in the last 15 years, is found to be suitable especially for the syntactic analysis of free constituent order languages.

Yarın hava Ankara'da nasıl?
Ankarada hava yarın nasıl?
Hava yarın ankarada nasıl?
Ankarada hava yarın nasıl?

As in all other areas of natural language processing, the lack of data resources and the sparse data problem caused due to the morphological nature of Turkish are the extra challenges encountered in our current domain.

4. DATASETS

This section presents our newly annotated datasets in order to both train and test our different system layers and adapt the NLP tools to our new domain. Table 2 provides the number of queries in each annotated dataset.

Table 2: Data Sets

Data Set Name	# of Queries	Annotation Type
STT_Train	1000	Normalized, Category labels
NER_Train	750	Named Entities
Parser_Train	620	Dependency relations
Data_Validate	478	Category labels and responses
Data_Test	100	Category labels and responses

STT_Train consists of 1000 queries related to nine pre-defined categories; making calls, sending sms messages and emails, getting directions, querying exchange rates, weather forecast and traffic information, searching on the internet and launching applications on the phone. The queries are taken via a speech to text API (described in Section 5) and manually normalized. *STT_Train* consists of these queries and their manually marked category labels in order to be used in the query classification layer. 578 queries (478 in *Data_validate*, 100 in *Data_Test*) from this set are manually prepared with expected server responses in order to be used in the parameter extraction layer. Figure 1 gives an example to the response structure which will be detailed in Section 5. The figure provides the manually tagged responses for two sample queries related to *call* and *sms* operations. The first query ("halamı ara" – *Call my aunt*) is tagged with a *call operationType* and *the receivername* whereas the second query ("engine mesaj at yoldayım geliyorum" – *Send an sms to Engin, I'm on my way, I'm coming*) is tagged with an *sms operationType* and *both the receiver name* and the *body text* of the sms.

NER_Train and *Parser_Train* are the datasets prepared in order to be used in domain adaptation of the named entity recognition and the parsing modules. *NER_Train* consists of 750 queries annotated with 7 named entity types namely ENAMEX, TIMEX and NUMEX categories [29]. *Parser_Train* consists of 620 morphologically and syntactically annotated queries. Manual annotation of this dataset is realized via ITU Treebank Annotation

```
req0001_1.request = halamı ara
req0001_2.expected =
{
  "responseCode": "api/status/ok",
  "responseType": "native_op",
  "nativeResponse": {
    "operationType": "native/call",
    "call": {
      "receiverName": "halam"
    }
  }
}
req0002_1.request = engine mesaj at yoldayım geliyorum
req0002_2.expected =
{
  "responseCode": "api/status/ok",
  "responseType": "native_op",
  "nativeResponse": {
    "operationType": "native/sms",
    "sms": {
      "receiverName": "engin",
      "bodyText": "yoldayım geliyorum"
    }
  }
}
```

Figure 1: Two sample queries manually tagged with expected server responses

Tool [30], [31]. In this annotation tool, the tokens of the input queries are first automatically analyzed by a morphological analyzer [16] and then human annotators are asked to select the most probable morphological tag sequence for the given input. As the syntactic layer, the tool allows to annotate the dependency structure [11] for each given input.

5. SYSTEM ARCHITECTURE

In this section we describe the design of our system which is composed of two main modules; a mobile client and a server-side application. The mobile application is responsible for 1) capturing the spoken request, 2) converting it into written text, 3) transferring the user requests to the server side and 4) displaying and/or performing the requested operation according to the result received from the server. The server side module is responsible for processing the user's query and understanding its true intention. It also collects information from third party web services if needed, and finally compiles the results and sends it back to the client. The mobile application communicates with the server-side service

using RESTful principles and via a defined JSON protocol. Figure 2 illustrates the main architecture of our system.

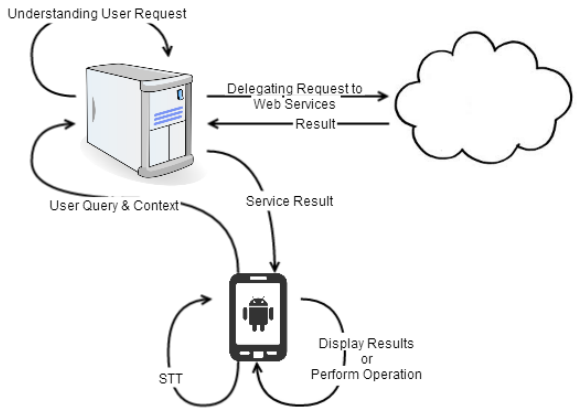


Figure 2: System architecture

The fields returned by the protocol message are as follows:

- Response Status (error or ok)
- Response Type (service or native_op)
- Operation Domain (call, sms, weather etc…)
- Content (parameters required by the determined query class)

The *response type* field specifies whether the operation requires results from third party web services (such as traffic information or map services) (*service*) or may be realized locally (*native_op*). As specified before, we have 9 supported request types specified by the field *operation domain*. The number of returned parameters (*content* field) varies depending on different request types. For example, querying exchange information will return the amount to be exchanged, the original and target currency types, and the exchange rate whereas launching an application will only return one parameter which will be an application's name on the phone. Figure 1 introduces two sample protocol messages.

As stated before, the application is designed to support nine types of requests. Table 3 provides the name of these supported operations together with the utilized third-party web services if any.

Table 3: User Request Types

Response Type	Request Type	Third Party Service
native_op	Starting calls	-
native_op	Sending sms	-
native_op	Sending e-mail	-
Service	Getting directions	Google Maps
Service	Querying currency exchange info	Google Currency Service
Service	Querying weather forecast	World Weather Online
Service	Querying traffic information	Yandex Traffic Service
Service	Searching on the web	Google search
native_op	Launching phone applications	-

5.1. MOBILE APPLICATION

Mobile application is responsible of taking the user request, transferring it to the server side and displaying or realizing the returned response. Google's speech recognition API for Android [28] is used on the mobile application side in order to capture the user spoken queries and convert them into written messages. The speech recognizer returns more than one text suggestion for a given entry and each suggestion has its own confidence level. A suggestion with a higher confidence level is more likely to be true. If any of the produced suggestions has more than a pre-defined threshold value (0.7), it is assumed as the correct suggestion. Otherwise, the application presents a list of text suggestions to the user. Figure 3 illustrates the speech recognition outputs for a sample request which will be sent to server: "gökhanı ara" (*call Gökhan*). Since none of the speech recognition API's results are above the threshold, the top 5 results are displayed to user and asked for manual selection.

After the speech recognition stage, the selected query text is sent to the server along with the user's context, which includes user id (device id), current location and time (provided on top of the figure between Content block). The response from the server is then either executed on the phone or displayed to the user. Figure 4 illustrates a sample result for a map query ("kadıköye nasıl giderim" (*how can I go to Kadıköy*)) on the mobile application.

URL Address: IP:Port/SanalBilge/rest/info
Content:

```
{
   "username": "android",
   "query": "gökhanı ara",
   "latitude": 41.00527,
   "longitude": 28.97696,
   "timestamp": 1427735981599
}
```

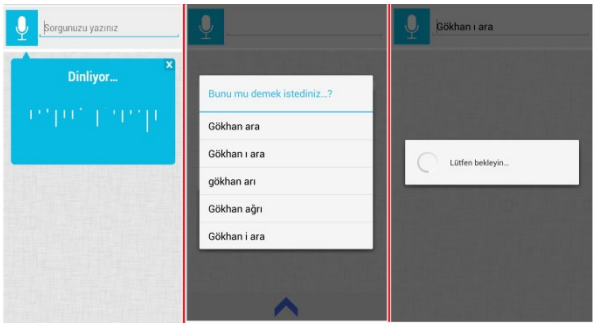


Figure 3: Mobile Application Speech Recognition Interface

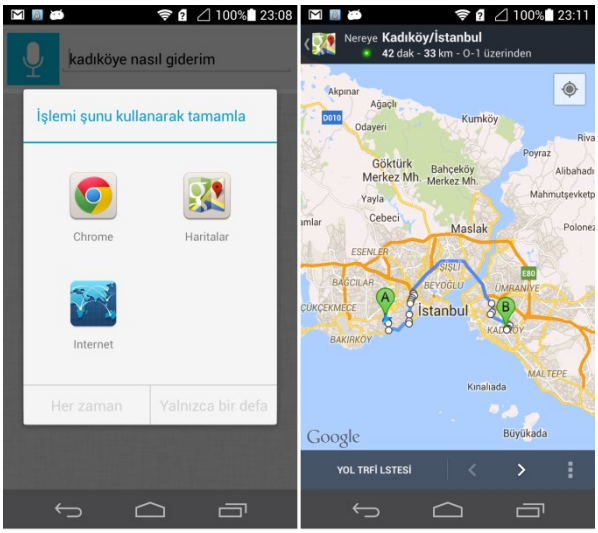


Figure 4: Screen shots of map result

5.2. SERVER-SIDE APPLICATION

The server-side application is responsible for processing the incoming request and returning generated response back to the client. Since the input data is the output of a speech recognizer and may contain mistakes (e.g., capitalization, wrong transliteration of proper nouns such as the example given in Figure 3), the incoming user requests are firstly normalized in order to avoid grammatical errors and then processed with different NLP layers. After these pre-processing steps, requests are mapped to one of the supported operations through a hybrid approach of rule-based and statistical classification. Finally, the required parameters are extracted by an extraction algorithm that is developed separately for each focused domain. Figure 5 presents the basic flow of server-side operations. Since the used 3rd party service for the traffic domain automatically obtains the current location information from the phone and does not take any additional parameter, the class based parameter extraction box in Figure 5 only lists the remaining 8 categories.

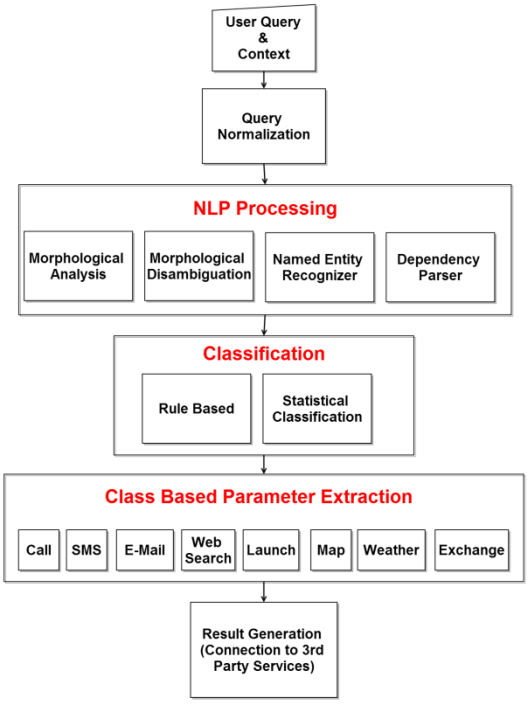


Figure 5: The main flow of the server-side application

5.2.1. Normalization & Language Processing

In this stage, incoming query is first normalized in order to increase success ratio of NLP tools and then processed by the Turkish NLP pipeline described by Eryiğit [16]. Since the mentioned pipeline is tailored for well-formed texts and the Web domain, it is noticed that some of its components do not succeed well in our new domain. An example to this phenomenon may be given from the syntactic parsing stage. Since the training sentences of the treebanks [32], [31] (used during the training stage of the utilized dependency parser) mostly have a punctuation (e.g., a period, a question mark) at the end, parsing model has difficulty in parsing our outputs coming from the speech recognizer (Figure 3). That is why, in the pre-processing stage, while some of the processors (tokenizer, morphological analyzer [13], morphological disambiguator) are directly used via the cited web service, some (namely, the normalizer, the named entity recognizer and the dependency parser) have to be adapted to our new domain.

Torunoğlu and Eryiğit [15] present a text normalizer for Web data which consists of seven different modules: letter case transformation, replacement rules and lexicon lookup, proper noun detection, diacritization (recover Turkish characters like "c" to "ç"), vowel restoration, accent normalization and a spelling corrector. Our investigations on our development set revealed that only two of these layers (capitalization of proper nouns and the accent normalization) are valuable for the problem in hand and the other normalization layers might create a harming effect on this domain. Therefore, a specific normalizer on top of the Google Speech API outputs was developed. In addition to these two components, normalizer was extended to handle separately written suffixes. To this end, we compiled a Turkish suffix list which is used to merge the suffixes with the preceding proper noun (if any) (after the insertion of an apostrophe sign and capitalization of the proper noun). For example, "ali ye" is a typical erroneous output of our speech recognizer and it is normalized as "Ali'ye" (*to Ali (a Turkish male name)*) with this normalizer.

Studies related to Turkish named entity recognition on user generated content displays a lower success rate compared to the ones tested with formal texts [33] [29]. Mobile assistant data is similar to UGC (User Generated Content) but it additionally possesses the errors propagated from the speech recognition layer. In this article, during the integration of the named entity recognition layer into our system, we tested different scenarios. We firstly used a NER system trained for ENAMEX, TIMEX and NUMEX types. The original system which is tailored for well-written text obtained a very low performance score in our dataset with 15.81% as expected. When we use the NER model tailored for Turkish UGC content from the social media sides, we obtained 43.83% which is still very low. In order to adapt this system to our domain, we annotated a new dataset (NER_Train) as introduced in Section 4 and used it for domain adaptation of named entity recognition module. 3rd row in Table 4 provides the F-score obtained at the end of 10-fold cross validation on this dataset. The performance (76.31%) is much higher than the previous models despite the very small size of the developed data set. The increase of the annotated data size is expected to improve the performances in this domain and reach the ones for English.

Table 4: Performance of the NER

	Training	Test	F-Score
NER for formal text	News Articles[43]	NER_Test	15.81%
NER for UGC	News Articles[43]	NER_Test	43.83%
NER for UGC (10 fold CV)	9/10 NER_Train	1/10 Ner_Train	76.31%

For the syntactic parsing of the user requests, we use Maltparser [34] as described in Eryiğit et al.[11]. Unfortunately, the reported performances could not be obtained in our new domain with the provided Turkish model by the author. In order to alleviate the performance loss due to the domain differences, we collected 620 user queries (*Parser_Train* Section 4) by asking people to enter queries in all our supported domains (call, sms, map, weather .etc). Since the queries are most of the time simple requests in our domain such as asking for weather, sentences were accordingly short (average word count per sentence is 4.6 vs. 10 in a formal Turkish Treebank [11]). Although there is still room for improvement with the annotation of additional training data, *Parser_Train* reveals a labeled attachment score of 71.53% which is comparable with the state of the art in Turkish dependency parsing results. Below we introduce our tests for the domain adaptation of the parsing model.

We first tested our data with the model trained with IMST(ITUMetuSabancıTreebank [32]) data. The results of this test are given in Table 5. The performances are provided as labeled attachment score, unlabelled attachment score and the label accuracy.

Table 5: Performance of parsing model trained with IMST and tested on *Parser_Train*

Labeled attachment score	27.55%
Unlabeled attachment score	52.94%
Label accuracy score	34.14%

As expected, the results on our new domain were very low when compared to the reported performances (Labelled attachment score (LAS): 75.3%, Unlabeled attachment score (UAS): 83.7%). In order to alleviate the problem related to the sentence ending punctuation marks, we repeated the same test with inserting a period sign at the end of every sentence. This operation increased the success rates (Table 6) of almost 30 percentage points, but the results were still low when compared to the reported performances.

Table 6: Performance of parsing model with punctuation marks trained with IMST and tested on *Parser_Train*

Labeled attachment score	57.05%
Unlabeled attachment score	76.79%
Label accuracy score	65.20%

As a final investigation, we re-trained the parsing model with the collected queries in order to adapt the parser to our new domain. Table 7 provides the results of 10-fold cross validation experiments on *Parser_Train*. The highest performances were obtained with this new setting (86,58% UAS, 71,53 LAS). In order to prepare the parsing model to be used in the remaining stages, all the 620 queries are included to the training data and the parser is trained with this.

Table 7: Performance of parsing model 10-fold cross validation *Parser_Train*

Labeled attachment score	71.53%
Unlabeled attachment score	86.58%
Label accuracy score	74.80%

5.2.2. Query Classification

This stage aims to map the pre-processed query to one of the nine supported request types. The used method is a hybrid approach which makes use of both rule-based and machine-learning methods. As a first step, the system looks for the rules if there is a match with the query. If there is no match, the system classifies the queries by using the statistical model that is generated by machine learning algorithms.

5.2.2.1. Rule Based Classification

In this step, incoming queries that are processed via NLP tools are processed with pre-defined rules and classified into a request type if any match occurs. Rules can be added in a generic way by using regular expressions [35]. Three different rule types are defined for using NLP tools' outputs.

1. **PLAIN_TEXT:** This type does not use the results of NLP tools and matches on plain texts. E.g. "Hava nasıl (How is the weather)"

2. **NER_TEXT:** Named entities in the query is replaced with their labels. Rule is processed after this step. E.g. "<Location> hava nasil" (How is the weather in <Location>"

3. **POS_PLAIN_TEXT:** Some words in the query remain the same and some replaced with part of speech tags. For example, the rule to match "annemi ara (call my mother)" is written like "<Noun> ara".

The rule engine allows to define rules by using regular expressions. Each rule has 4 parts:

1. A sequence pattern that the query will be matched to,
2. How the rule will be handled,
3. Which domain that rule classifies into,
4. Rule's priority value.

An example rule for Call domain is given below. The example rule matches queries that start with "Lütfen (*please*)" (not mandatory) and then a person (not mandatory) and "ara (*call*)". For example "Lütfen John'u ara (Please call John)" or "John'u ara (call John)" queries will be mapped with this rule and system will classify the query to the CALL domain.

^ (Lütfen)? (<Person>)? ara, NER_TEXT, CALL, 1
(^ (Can you)? (please)? call (<Person>)?, CALL, 1)

5.2.2.2. Statistical Classification

Defining and processing rules are easy and fast. Each rule contains a specific case and rules have very high precision rate but a low recall rate. On the other hand, managing too many rules is very difficult and defining rules to handle all queries is neither possible nor practical. Therefore, only 38 rules are defined to cover basic query templates and the system is supported with statistical classification. For the statistical classification experiments, *STT_Train* is used with 10-fold cross-validation. Different machine learning algorithms (Logistic Regression [36], Support Vector Machines (SVM) [37],[38], Naive Bayes [39], Decision Tree [40] with different confidence factors and K-Nearest Neighbors [41]) are experimented by the use of Weka [42] tool.

In order to alleviate the data sparseness problem caused by the agglutinative nature of Turkish, the lemmas are used in the feature representation and the named entities are replaced with their entity types. Thereby a query like "erdinçi arar mısın lütfen" (*can you please call Erdinç*) is converted to "Erdinç ara mı lütfen" and then "Erdinç" is tagged as a person name and query converted to "<PERSON> ara mı lütfen". After this pre-processing stage on training data, Bag of Words [43] approach is used to convert sentences to number vectors where each word is represented by a numeric value. In this approach each column in the vector stands for a unique word and feature values represent the occurrence count of the related word within the sentence.

The impact of using stemming and named entity recognition NLP layers on the success of statistical classification is measured and provided in Table 10. The results reveal that the added value by NLP pre-processing ranges between 0.2 (for support vector machines) and 10.7 (for Bayesian networks) percentage points depending on the preferred machine learning algorithm for the query classification stage. The best results are obtained with the SVM algorithm and pre-processing steps. The remaining

experiments are performed by selecting this best performing algorithm.

Table 8 provides the success rates of the SVM algorithm on each separate request type. It is observed that the highest results are obtained for the "Weather" and "Traffic" domains since most of the queries in these domains contain similar patterns or words, e.g. "Londra'da hava nasıl" (*how is the weather in London*), "Bugün trafik nasıl" (*how is the traffic today*). The algorithm has the lowest score with 92.1% on the web Search domain where the queries vary more.

Statistical classification alone reaches a 95.8% F-measure with the SVM algorithm. When the approach is merged into a hybrid model together with the rule-based classification introduced above, the success improves to 98.3%.

Table 8: Class based success results of the SVM algorithm

Domain	**Accuracy**	**F-Measure**
Call	99.0%	95.7%
Exchange	97.0%	97.0%
Email	94.0%	96.9%
Map	93.90	95.1%
Weather	100.0%	98.6%
Message	96.8%	96.8%
Traffic	100.0%	100,0%
Application	93.4%	92.4%
Search	92.1%	93.8%
Average	95.8%	95.8%

5.2.3. Parameter Extraction

The last stage of the server-side application is the parameter extraction where related parameters (listed in **Hata! Başvuru kaynağı bulunamadı.**) should be extracted for each supported tasks. Each domain has its own parameter extraction algorithm which uses morphological analysis, named entity recognition and dependency parser outputs. *Data_Validate* is used for the tuning of this stage. As an example, **Hata! Başvuru kaynağı bulunamadı.** illustrates the parameter extraction flow for the *weather* domain.

The overall system performance on *Data_Test* is compared with Turkcell Mobil Asistan, Asistan B, CEYD-A and SIRI applications.

Table 11 provides the success results of applications on each request types. Fields with "N/A" value in the table means that the related application does not support the relevant request type.

As may be observed from

Table 11, the architecture introduced in this article and the SIRI application, which use natural language processing techniques, give better performances compared to other applications. 70.8% success rate of our introduced architecture is treated to be very promising despite the fact that the system is trained with a very modest size of training data compared to commercial applications; e.g. SIRI 69.62%. The performances obtained with Turkcell Mobil Asistan, AsistanB and CEYD-A applications, which use rule based approaches, fall very behind when compared to the two top performing ones. The main reason for this may be related to the fact that these applications do not make use of NLP solutions adequately and try to directly map the queries to pre-defined fixed patterns. Within these applications, CEYD-A has a richer rule structure and it allows its users to define their own rules. Moreover a new language is developed in order to allow its users to define rules easily. As a result, CEYD-A application provides better results compared to the other rule based applications.

When SIRI and the introduced approach are compared more closely on each specific request type, we may see that the introduced approach in this article performs better on weather, exchange, email, map and call related request types whereas SIRI performs better on web search, message and launching related tasks.

Table 9: The parameters to be extracted for each domain.

Class	Parameters
Call	receiver name, phone number
Sms	body text, receiver name, phone number
Email	body text, receiver name, email address
Web Search	query text, web site
Start Apps	application name
Map	departure, destination, PoI
Weather	location, time
Exchange	from/to currency pair

Table 10: Statistical Classification Results Before and After NLP Pre-Processing Stages

	Before Stemming and NER Steps		**After Stemming and NER Steps**	
Algorithm	**Acc.**	**F-Measure**	**Acc.**	**F-Measure**
Decision Tree 0.25	85.9%	86.1%	89.9%	90.0%
Decision Tree 0.5	85.7%	86.0%	90.3%	90.4%
Decision Tree Unprunned	85.4%	85.8%	90.3%	90.4%
Logistic Regression	94.1%	94.1%	95.2%	95.2%
NaiveBayes Multinominal	92.6%	92.2%	93.8%	93.5%
NaiveBayes	92.2%	92.2%	93.8%	93.8%
Bayes Net	82.5%	83.0%	92.3%	92.3%
KNN	87.9%	87.8%	90.4%	90.4%
SVM	95.3%	95.5%	95.8%	95.8%

The advantage of collecting high-volume of sample data from different patterns and integration with semantic web technologies is obvious on SIRI's performance for web search and message sending tasks. SIRI may directly respond to queries like "Atatürk'ün doğum tarihi kaç" (*When was Atatürk born*) due to the integration of semantic web technologies and to queries like "gökhana sor neredesin" (*ask gökhan where are you*) due to the availability of more training data. In this later example, since the verb "sormak" (*ask*) never occurred in our collected small-sized training data set, our system could not react properly to this sample. This shows that the introduced approach has still room for improvement by the addition of more training data. While our developed application successfully operates queries like "Gökhan'a daha sonra geri döneceğimi email at" (*send mail to Gökhan that I will come back later*) and extract required parameters without asking for new information, SIRI asks user additional questions such as "who would you like to send mail" and "what would you like to say". SIRI is found inadequate for extracting parameters in calling someone from contacts and extracting road directions. As discussed earlier, SIRI's English version is much better than its Turkish localization. This validates our claim that using more sophisticated natural language processing techniques tailored for Turkish helps significantly to the success of such applications.

6. CONCLUSION AND FUTURE WORK

This article introduces a Turkish mobile assistant architecture and a prototype application. The aim of the study is to show the impact and the potential of using basic natural language processing techniques in this new domain. Comparisons with similar applications are provided within the article. The evaluations on the impact of NLP pre-processing layers on the query classification performances reveal that the added value by NLP may range from 0.2 to 10.7 percentage points depending on the preferred machine learning algorithm for the query classification stage. The impact of NLP for the parameter extraction stage is also crucial since the outputs of NLP modules are used systematically by the extraction rules. The overall accuracy of the introduced approach is measured as 70.8% which is very promising under the fact that the system is trained with very limited-size of annotated data when compared to its competitors.

To the best of our knowledge, this article is the first academic study on this topic. The technology introduced in this article is basically designed for the case of a mobile assistant but it can be used for every voice-enabled control system to improve the user experience, such as smart homes or smart televisions.

As future work, dialog management and text to speech features should also be investigated. Basic natural language processing layers are just a first step for a successful Turkish mobile assistant application. Semantic technologies are also very crucial in this new domain and more research is needed to obtain high-performing and more natural assistant applications.

Acknowledgements

This work is supported by Turkey's Ministry of Science, Industry and Technology under SANTEZ grant no: 0073-STZ.2013-1. The conclusions and remarks given in this article are not the official views of the ministry.

Table 11: Domain Based Performances of Tested Applications (tested on May 2015)

Domain /Application	AsistanB	Turkcell	CEYD-A	SIRI	This article
Weather	0.00%	50.00%	50.00%	83.33%	**91.67**%
Search	0.00%	0.00%	27.27%	**63.64**%	36.36%
Exchange	60.00%	30.00%	40.00%	N/A	**90.00**%
E-Mail	N/A	N/A	0.00%	36.36%	**63.64**%
Map	0.00%	16.67%	66.67%	66.67%	**75.00**%
Call	0.00%	0.00%	27.27%	63.64%	**81.82**%
Launch	50.00%	N/A	58.33%	**91.67**%	83.33%
Message	0.00%	50.00%	10.00%	**70.00**%	40.00%
Average					**70.8%**

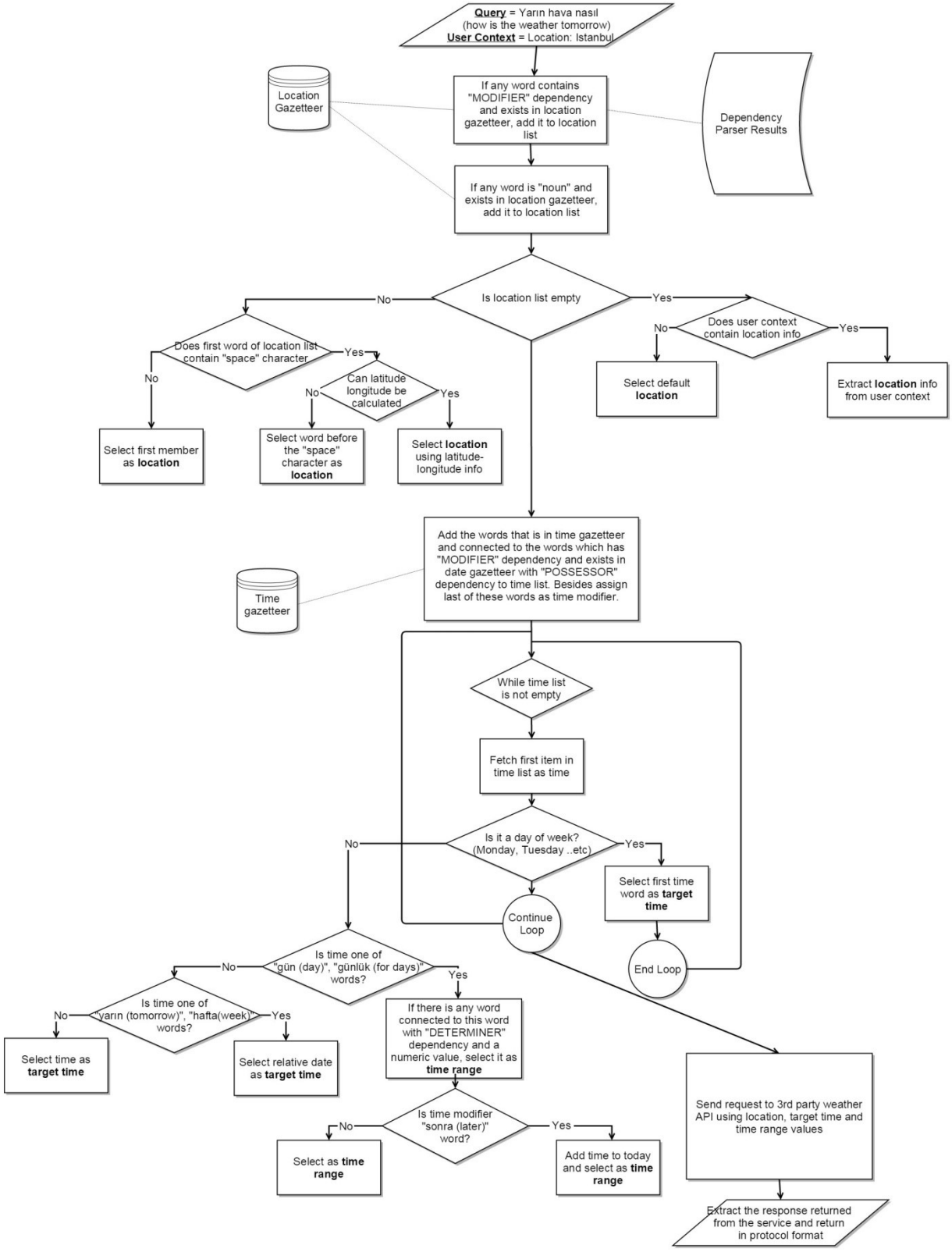


Figure 6: Weather Extraction Flow

References

[1] Celikkaya, G. and G. Eryiğit, A mobile assistant for Turkish. TÜRKİYE BİLİŞİM VAKFI BİLGİSAYAR BİLİMLERİ ve MÜHENDİSLİĞİ DERGİSİ, 2014. 7(1 (Basılı 8).

[2] Apple. SIRI. 2011; Available from: http://www.apple.com/ios/siri/.

[3] Google. Google Now. 2015; Available from: https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox.

[4] Microsoft. Cortana. 2014; Available from: https://en.wikipedia.org/wiki/Cortana_(software).

[5] Labs, R. Robin - the Siri Challenger. Available from: https://play.google.com/store/apps/details?id=com.magnifis.parking.

[6] Speaktoit. Assistant 2011; Available from: https://en.wikipedia.org/wiki/Assistant_(by_Speaktoit).

[7] Asistan B - Turkçe Sesli Asist. 2015; Available from: https://play.google.com/store/apps/details?id=com.buronya.asistanb.

[8] CEYD-A Turkçe Sesli Asistan. 2015; Available from: https://play.google.com/store/apps/details?id=com.cenker.yardimci.app.

[9] Oflazer, K., Two-level description of Turkish morphology. Literary and linguistic computing, 1994. 9(2): p. 137-148.

[10] Yuret, D. and F. Türe. Learning morphological disambiguation rules for Turkish. in Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. 2006. Association for Computational Linguistics.

[11] Eryiğit, G., J. Nivre, and K. Oflazer, Dependency parsing of Turkish. Computational Linguistics, 2008. 34(3): p. 357-389.

[12] Seker, G.A. and G. Eryigit. Initial Explorations on using CRFs for Turkish Named Entity Recognition. in COLING. 2012.

[13] Sahin, M., U. Sulubacak, and G. Eryigit. Redefinition of Turkish morphology using flag diacritics. in Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013), Phuket, Thailand, October. 2013.

[14] Eryigit, G., et al. Turksent: A sentiment annotation tool for social media. in Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse. 2013.

[15] Torunoglu, D. and G. Eryigit. A cascaded approach for social media text normalization of Turkish. in Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)@ EACL. 2014.

[16] Eryigit, G. ITU Turkish NLP Web Service. in EACL. 2014.

[17] Neustein, A. and J.A. Markowitz, Mobile speech and advanced natural language solutions. 2013: Springer Science & Business Media.

[18] Bellegarda, J.R., Spoken language understanding for natural interaction: The siri experience, in Natural Interaction with Robots, Knowbots and Smartphones. 2014, Springer. p. 3-14.

[19] Ward, W. and S. Issar. Recent improvements in the CMU spoken language understanding system. in Proceedings of the workshop on Human Language Technology. 1994. Association for Computational Linguistics.

[20] He, Y. and S. Young. A data-driven spoken language understanding system. in Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on. 2003. IEEE.

[21] Dowding, J., et al. Gemini: A natural language system for spoken-language understanding. in Proceedings of the 31st annual meeting on Association for Computational Linguistics. 1993. Association for Computational Linguistics.

[22] Hawkins, J.C., et al., Integrated personal digital assistant device. 2008, Google Patents.

[23] Acero, A., et al., System for using statistical classifiers for spoken language understanding. 2012, Google Patents.

[24] Bangalore, S., N.K. Gupta, and M.G. Rahim, System and method of spoken language understanding in human computer dialogs. 2013, Google Patents.

[25] Dusan, S. and J. Flanagan, System and method for adaptive language understanding by computers. 2002, Google Patents.

[26] Turkcell. Turkcell - Mobil Asistan. 2015; Available from: http://www.turkcell.com.tr/tr/hakkimizda/video-galeri/reklam-filmleri/mobil-asistan-cell-in.

[27] Idris. 2014; Available from: https://forum.shiftdelete.net/threads/idris-2-turkce-sesli-sanal-asistan-cikti.384057/.

[28] Google. Google Speech Recognition API Available from: https://developer.android.com/reference/android/speech/SpeechRecognizer.html.

[29] Seker, G.A. and G. Eryigit. Extending a CRF-based Named Entity Recognition Model for Turkish Well Formed Text and User Generated Content. in Semantic Web Journal. 2016.

[30] Eryiğit, G. ITU treebank annotation tool. in Proceedings of the Linguistic Annotation Workshop. 2007. Association for Computational Linguistics.

[31] Pamay, T., et al. The Annotation Process of the ITU Web Treebank. in The 9th Linguistic Annotation Workshop held in conjuncion with NAACL 2015. 2015.

[32] Sulubacak, U., T. Pamay, and G. Eryiğit, IMST: A revisited Turkish dependency treebank, in The First International Conference on Turkic Computational Linguistics - TurCLing 2016 2016: Konya, Turkey. p. 1-6.

[33] Çelikkaya, G., D. Torunoğlu, and G. Eryiğit. Named entity recognition on real data: a preliminary investigation for Turkish. in Application of Information and Communication Technologies (AICT), 2013 7th International Conference on. 2013. IEEE.

[34] Nivre, J., et al., MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, 2007. 13(02): p. 95-135.

[35] Regular Expression. Available from: http://en.wikipedia.org/w/index.php?title=Regular_expression&oldid=654317915.

[36] Le Cessie, S. and J.C. Van Houwelingen, Ridge estimators in logistic regression. Applied statistics, 1992: p. 191-201.

[37] El-Manzalawy, Y. and V. Honavar, WLSVM: integrating libsvm into weka environment. Software available at http://www. cs. iastate. edu/yasser/wlsvm, 2005.

[38] Chang, C. and C. Lin, {LIBSVM}: a Library for Support Vector Machines (Version 2.3). 2001.

[39] John, G.H. and P. Langley. Estimating continuous distributions in Bayesian classifiers. in Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. 1995. Morgan Kaufmann Publishers Inc.

[40] Quinlan, J.R., C4. 5: Programming for machine learning. Morgan Kauffmann, 1993: p. 38.

[41] Aha, D.W., D. Kibler, and M.K. Albert, Instance-based learning algorithms. Machine learning, 1991. 6(1): p. 37-66.

[42] Holmes, G., A. Donkin, and I.H. Witten. Weka: A machine learning workbench. in Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on. 1994. IEEE.

[43] Zhang, Y., R. Jin, and Z.-H. Zhou, Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics, 2010. 1(1-4): p. 43-52.