A Mobile Assistant for Turkish

Gökhan Çelikkaya *Gülşen Eryiğit*
Dep. of Computer Eng. Istanbul Technical University Istanbul, Turkey
gcelikkaya@itu.edu.tr; gulsen.cebiroglu@itu.edu.tr;

ABSTRACT

In this paper we present a design and an implementation of a mobile assistant application that understands and meets users' requests in Turkish language. The application is able to understand requests for a set of phone operations such as calling a contact or sending an email and requests for a set of information services such as map, weather, and traffic. The understanding of user queries relies on existing research on natural language processing for Turkish and a hybrid approach of rule-based and statistical classification methods. Our performance tests revealed high accuracy results for the operations that our application supports.

1 Introduction

Today people try to find easier and simpler ways to interact with their mobile devices such as smart phones and tablet computers to have their work done and access to information. In this regard, voice-controlled mobile assistant applications became very popular recently. Ever since SIRI [1]became a success and several alternative applications have been released to the application markets. Nevertheless, most of those applications support only English and provide services mostly for English-speaking countries.

In the last years, successful voice recognition [1], and natural language processing [2], [3], [4], [5], [6], [7], [8] components have been released for Turkish. However, such components have not been integrated with any high tier applications.

In this study, we present our ongoing effort on developing a mobile assistant application and its associated services that understands Turkish input and accomplish users' requests with high accuracy. The application is able to understand various user intents including but not limited to calling and texting to a contact, accessing to weather and traffic information. We explain our system architecture and our methodology of understanding user requests, in which we leverage and improve existing research and tools for Turkish NLP such as morphological analyzer, morphological disambiguator, named entity recognizer, and dependency parser.

Understanding user requests is a multi-step process in our system; the user query is processed with NLP tools, and then the query is mapped to one of the supported operations through a hybrid approach of rule-based and statistical classification. In the following step, the parameters such as contact name or search term -if any- from the query are extracted so that the mobile device can handle the requested operation or an external web service can be queried with correct parameters in order to fetch the requested information. We present and discuss the performance of the classification and parameter extraction methods that we have tested using a set of real user queries.

[1] http://www.apple.com/ios/siri/

Table 2: Performance of parameter extraction in terms of accuracy

Domain	Call	SMS	E-mail	Web Search	Start Apps	Map	Weather	Exch.	Overall
Acc. (%)	72.86	81.22	88.67	37.50	74.51	81.48	72.38	36.36	69.25

The rest of this paper is organized as follows: Section 2 gives brief information about related work, Section 3 describes the design of our system, Section 4 explains the process and methodology of understanding users' requests in the Turkish language within the context of our application, and finally, Section 5 concludes and discusses future work.

2 Related Work

As it is the case for most of the language technology studies, the mobile assistant technology also started firstly for English. There are various patents [9], [10], publications [11] and applications for English; e.g. Siri (Apple) and Google Now (Google) are the most advanced and the most used voice-controlled personal mobile assistant applications in the world today. Since Turkish is a morphologically rich language, its morphology and syntax are very different and arguably more complex compared to English. Thus, most of the time it is not convenient to use the same methodologies and tools that are proposed and available for English.

To the best of our knowledge, there do not exist any voice-controlled assistant application that supports the Turkish language by leveraging advanced natural language processing techniques and services to understand users' true intent with high accuracy. Our survey on the existing Turkish virtual assistant applications, of which there are a few in total, revealed that they rely on rule-based approaches and use very simple natural language processing components or even none and focusing more on visual user experience. For such reasons, the ability of those applications to process queries in natural language and perform the requested actions is not strong or comprehensive. Turkcell Mobil Assistant[2] is one of the most well-known Turkish mobile assistant application. However, it can reply questions only in a specific pattern. For instance, while it can understand the question "hava nasıl?" (how is the weather?) and returns the weather forecast in the current location, it fails to reply with the correct result to the question "NewYork'da hava nasıl?" (how is the weather in NewYork?) where the user actually wants to query the weather in a different location. In this work, we show that utilizing the recent research and tools on Turkish NLP will facilitate the development of a mobile assistant application with a high level of performance. By this way, we expect our project to become a pioneer in its field of practice.

[2] http://www.turkcell.com.tr/servisler/turkcell-mobil-asistan

3 System Design

In this section we describe the design of our system which comprises a mobile Android client application and a server-side service. The mobile application (Figure 1) is responsible to convey user requests to server and display the service results or perform the requested operation on behalf of the user such as calling a contact person or starting an application. The server processes a user query to understand his/her true intent, then if required delegates the request to the third party web services, and finally, compiles the results and send it back to the client. It is beyond the scope of this paper to discuss the runtime performance of the server-side implementation; though, it is worth mentioning that the query understanding (natural language processing and query classification) takes about half a second on average on a commodity hardware.

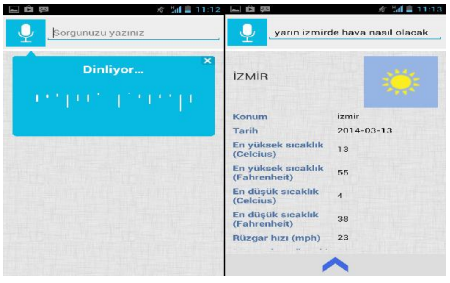

Figure 1: Screen shots of the mobile application: recognizing speech input (left), displaying results for a weather query (right).

We use Google's built-in speech recognizer to convert users' speech queries into text, which we have tested and found efficient for Turkish speech recognition (we have considered accent variations of Turkish as well). The speech recognizer has a limit of 160 characters per speech request. Besides, it provides a set of confidence levels along with the text results. We consider the text with the highest confidence level as the correct mapping of what user has said, if the confidence level of this text is greater than 0.7 (max. being 1.0), otherwise, the application presents a list of text suggestions to the user ordered in the descending order of their confidence levels. The selected query text is further sent to the server along with user context which includes the user id, the current location and time. The communication between client and server is based on REST principles to make the server-side more scalable as no user session being stored on the server between requests. Both the request and response messages are encoded with JSON format on our messaging protocol.

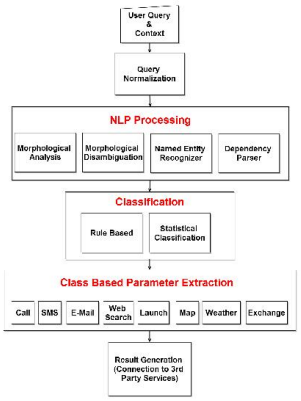

Figure 2:The flow of operations for understanding user requests.

4 Understanding User Requests

In this section, we explain the process and methodology of understanding users' requests in Turkish. Currently our application supports nine types of requests:

- calling a contact,
- sending sms to a contact,
- sending email to a contact,
- searching terms on Web (we rely on Google search),
- starting installed apps on the phone,
- getting route directions or searching a point of intereston the map (Google Maps), exchanging between currencies (Google Currency Service),
- accessing the weather forecast (World Weather Online)
- accessing to traffic information (Yandex Traffic Service)

Figure 2 presents the flow of operations for understanding user requests. Upon receiving a user's query and context, the query is normalized (curated) through the steps of abbreviation expansion, capitalization of proper nouns, and verb tense correction [8]. In the second phase, the normalized query is processed via a Turkish NLP pipeline available as a SaaS [12] consisting of a morphological analyzer [7], a morphological disambiguator, a named entity recognizer [6], and a dependency parser [5]. The outputs of natural language processing components are used both in the classification and parameter extraction phases. In the classification phase, the processed query is mapped to one of the supported operations (i.e., domains). Our classification methodology is a hybrid approach of rule-based and statistical classification. After a query is mapped to a domain, the parameters[3] are extracted from the query, so that the phone can perform the requested operation or related web services can be called with correct parameters.

4.1 Query Normalization

As speech queries are in natural language, they often include grammar and pronunciation errors. Moreover, the used speech recognizer contributes to such errors by for instance separating suffixes from their stem despite the fact that Turkish is an agglutinative language and by outputting mistakenly the first letter of proper nouns with lower case letters (e.g. "gökhan a" instead of the correct form "Gökhan'a"). Consequently, the performance of the NLP tools we use degrades due to these erroneous spellings. In order to alleviate this problem, we needed to make extra normalization effort in addition to the previous work [8]

[3] Since we do not extract any parameter for the traffic service, Figure 2 shows only 8 domains in the parameter extraction phase

normalizes its input by the following stages: letter case transformation, replacement rules and lexicon lookup, proper noun detection, deasciification, vowel restoration, accent normalization. During our experiments, we observed that the most important normalization layers for our case were 1) The capitalization of Proper Nouns and 2) Accent Normalization which fixes verb tense errors occurring under spoken accents. The normalizer understands the tense of a given verb and replaces it with its correct tense form. For example "gelcem" (informal way of saying "I will come") is turned into "geleceğim" (I will come) which is the proper form. In addition to verb inflections, incorrect spelled question words are corrected as well. For instance, "gidiyozmu?" is corrected into "gidiyor muyuz?" (are we going?). In this work, we have extended the normalizer with a support for handling separated suffixes issue which is special to our used speech recognizer. We have compiled a Turkish suffix list, which we use to merge separated suffixes with the preceding word. If the word is a proper noun, we merge the noun and its suffix with an apostrophe and then capitalize the proper noun. For instance; "ahmet e" (Ahmet is a proper noun) becomes "Ahmet'e".

4.2 Query Normalization

The Named Entity Recognizer (NER) tool that we use in this work is based on the work of Şeker and Eryiğit, [6]. It utilizes a statistical modeling method, namely Conditional Random Fields (CRFs) [13], which is a framework for building probabilistic models to segment and label sequences of input samples. The original NER tool is designed for general purpose, hence we needed to adapt it for our domain of concern. To this end, we have added 750 tagged queries and commands in to the existing data set. We have also added currency and time gazetteers, and then divided all the gazetteers into two categories: 1) Base Gazetteers: First Name, Last Name, Location, Money, Time and 2) Generator Gazetteers: Person, Location, Organization. Similarly to the original article, to retrain the NER tool, we considered the following features; whole word, word stem, POS tag or word, noun state, proper noun, inflectional groups, lower-upper case, sentence begin, and existence in gazetteers. Besides, the extended NER tool is able to tag PERSON, LOCATION, ORGANIZATION, TIME, MONEY, and PERCENTAGE entities in a given sentence. This tool is crucial for our system, since the output of the NER tool is directly used in the classification and extraction phases presented in the following sections, respectively.

4.3 Classification

In this section, we explain our query classification methodology.

1) ***Rule Based Classification***: In this work, we have developed a rule engine to directly create and modify sets of classification rules. The engine enables to define rules containing regular expressions, and POS and NER tags together. Each rule expression follows the order of a pattern string to be matched against the processed user query, the domain name, and a value to define an order of priority for the evaluation sequence. The following expression is an example of a rule of Call domain.

ˆ(Lütfen)?(<Person>) arayabilir misin)?,CALL,1
(Can you)? (please)? call (<Person>)?, CALL,1

2) ***Statistical Classification***: As it is impractical to define rules that would cover all possible kinds of queries for a domain, we use a statistical classifier for the queries that do not match any rules. In our current application setting, we use a Support Vector Machine (SVM) classifier [14], a decision

Table 1: Performance of classification methods

Algoritma	Doğruluk	Precision	Recall	Fskoru	ROC Alanı	TP oranı	FP oranı
Decision Tree 0.25	85,90%	0,883	0,859	0,861	0,956	0,859	0,025
Decision Tree 0.5	85,70%	0,878	0,857	0,860	0,957	0,857	0,025
Decision Tree Unprunned	85,40%	0,870	0,854	0,858	0,959	0,854	0,025
Logistic Regression	94,10%	0,942	0,941	0,941	0,991	0,941	0,007
NaiveBayes Multinominal	92,60%	0,929	0,926	0,922	0,986	0,926	0,011
NaiveBayes	92,20%	0,924	0,922	0,922	0,982	0,922	0,012
Bayes Net	82,50%	0,841	0,825	0,830	0,959	0,825	0,029
IB1	87,90%	0,883	0,879	0,878	0,970	0,879	0,016
SVM	95,80%	0,959	0,958	0,958	0,989	0,958	0,007

based on our performance evaluation that we present below.

a) Data Model: Our training and test data consists of 1000 queries or commands (100+ queries per domain) collected from real users. For each query, first, we remove the stop words. Then, each remaining word in the query is stemmed, and replaced with a NER tag if it matches any. As a consequence, the same type of entities are represented with the same feature to improve the classification performance. Finally, we tag the resulting query with its corresponding domain type. Below is an example of this process.
Original Query: yarın istanbul da yağmur yağacak mı (will it rain tomorrow in istanbul).
Processed Query: <time> <location> yağmur yağ, Weather (rain <time> <location>, Weather)
To represent the data as numerical values, we use the bag-of-words (bag-of-n-grams) model, in which a query is represented as the multi set of its n-gram (contiguous sequence of n items from the query). This model is widely considered in document classification methods, where the frequency of occurrence of each n-gram is used as a feature for training a classifier.

b) Classification Methods: In this work, we have evaluated the performance of various classification methods using our query data set. These methods include Decision Tree (with 0.25, 0.5 and unpruned as confidence factors), Logistic Regression, NaiveBayes (Multinominal), Bayesian Network, K-Nearest-Neighbor, and Support Vector Machine. Since the scale of our data is small, we have used 10-fold cross validation to compare the performance of the classification methods. We repeated the tests ten times and we report the average results. We only present the results in which we use 1-gram model, which yields better results compared to that of when we use 2-gram and 3-gram models.

Table 1 presents the performance of the classification methods in terms of accuracy, precision, recall, F-Score, ROC area, true and false positive rate metrics. The SVM and the Bayesian Network achieves the best and worst results, respectively. The SVM outperforms the other methods except for the metric of ROC area. We observe that the overall results are very high as by nature our data set comprises relatively short queries and commands. Moreover, we also found out that the performance of the classifiers for the Web Search and Start Application related queries are relatively low. We attribute this to the lack of NER support for application names and to the fact that one can search on web with very broad search terms.

4.4 Parameter Extraction

For each domain, we have devised a complex individual parameter extraction method. Table 3 presents the parameters that we can extract from queries for each domain. The explanation of all extraction methods is beyond the scope of this paper, though, it is worth mentioning that the extraction methods leverage user context, NER and POS tags, the dependency relations, and the existence of pre-defined keyword patterns. As an example, Figure 3 illustrates the parameter extraction method for the Map domain.

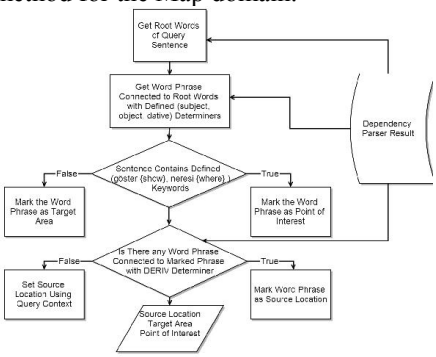

Figure 3: Parameter extraction process of the Map domain.

Table 3:The parameters that can be extracted for each domain

Domain	Parameters
Call	receiver name, phone number
Sms	body text, receiver name, phone number
Email	body text, receiver name, email address
Web Sear	query text, web site
Start Apps	application name
Map	departure, destination, point of interest
Weather	location, time
Exchange	from/to currency pair
Traffic	location

Table 2 shows the accuracy results of our parameter extraction methods that we run on our 1000 queries data set. We define accuracy as the ratio of true parameter extractions to

the total number of parameters. We have attained high accuracy results for our parameter extraction methods, with an average score of 88%, except the methods of Call and Web Search, which yields around 70% accuracy.

1) Mapping Parameters to the Entities in the Phone: The Turkish alphabet, which is a variant of the Latin alphabet, consists of 29 letters, six of which (C-Ç, G-Ğ, I-İ, O-Ö, S-Ş, U-Ü) have two variants for the phonetic requirements of the language. This poses a challenge in matching names in queries to the names on the contact list of a phone, as users tend to save contact names using dot-less characters while speech-to-text service converts person names to their original form (e.g., saving Özgür as Ozgur – a Turkish person name). Moreover, users may mention only the first or last name of a contact or partial name for an application when they ask to call someone or start an application. To solve the exact matching issues mentioned above, in the mobile application we use the Monge-Elkan approximate text string matching algorithm [15], which measures the similarity between two strings each of which may comprise one or more words. In case the highest similarity score is below a certain threshold, we let the user choose among a few results corresponding to the strings with the highest similarity scores.

5 Conclusion and Future Work

In this paper we have presented our ongoing work on developing a mobile assistant application that understands Turkish language. We explained our system architecture and our methodology of understanding user queries, in which we leverage and improve existing research and tools for Turkish NLP. We also presented the performance of query classification and parameter extraction methods that we use in this work.

As future work, we plan to support a dialog-based interaction between users and the application and make the application personalized considering the context and previous queries of the users. We also plan to extend our services for instance with support for factual questions, and extend the performance tests with a broader set of real user queries.

Acknowledgements

This work is supported by the research project SANTEZ (Grant: 0073-STZ.2013-1). The authors want to thank Ö.Ozan Sönmez for helpful discussion.

6 References

[1] H. Sak, M. Saraclar, and T. Gungor, "Morphology-based and sub-word language modeling for turkish speech recognition," in Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. IEEE, 2010, pp. 5402–5405.

[2] J. Nivre, J. Hall, J. Nilsson, G. Eryig˙it, and S. Marinov, "Labeled pseudo-projective dependency parsing with support vector machines," in Proceedings of the Tenth Conference on Computational Natural Language Learning. Association for Computational Linguistics, 2006, pp. 221–225.

[3] S. Buchholz and E. Marsi, "Conll-x shared task on multilingual de- pendency parsing," in Proceedings of the Tenth Conference on Compu- tational Natural Language Learning. Association for Computational Linguistics, 2006, pp. 149–164.

[4] K. Oflazer and İ. Kuruöz, "Tagging and morphological disambiguation of turkish text," in Proceedings of the fourth conference on Applied natural language processing. Association for Computational Linguistics, 1994, pp. 144–149.

[5] G. A. Şeker and G. Eryiğit, "Initial explorations on using CRFs for Turkish named entity recognition," in Proceedings of COLING 2012, Mumbai, India, 8-15 December 2012.

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

[7] J. C. Hawkins, W. B. Rees, D. O. Chyi, and R. Y. Haitani, Interface for processing of an alternate symbol in a computer device. Google Patents, dec " 13" 2005, uS Patent 6,975,304.

[8] P. Sawhney, P. King, H. W. Ham, and L. Wang, Apparatus and method for mobile personal assistant. Google Patents, aug " 11" 2011, uS Patent App. 13/207,781.

[9] A.NeusteinandJ.A.Markowitz,MobileSpeechandAdvanced Natural Language Solutions. Springer, 2013.

[10] G. Eryiğit, "ITU Turkish NLP web service," in Proceedings of the Demonstrations at EACL 2014 (EACL). Gothenburg, Sweden: Association for Computational Linguistics, April 2014.

[11] J.Nivre,J.Hall,J.Nilsson,A.Chanev,G.Eryiğit,S.Kübler,S.Marinov, and E. Marsi, "Maltparser: A language-independent system for data-driven dependency parsing," Natural Language Engineering Journal, vol. 13, no. 2, pp. 99–135, 2007.

[12] G. Eryiğit, J. Nivre, and K. Oflazer, "Dependency parsing of Turkish," Computational Linguistics, vol. 34, no. 3, pp. 357–389, 2008.

[13] D. Torunoğlu and G. Eryiğit, "A cascaded approach for social media text normalization of Turkish," in 5th Workshop on Language Analysis for Social Media (LASM) at EACL. Gothenburg, Sweden: Association for Computational Linguistics, April 2014.

[14] C. Cortes and V. Vapnik, "Support-vector networks," Mach. Learn., vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[15] A. Monge and C. Elkan, "The field matching problem: Algorithms and applications," in In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 267–270