Ministry of Higher Education

Kabul University

Faculty of Computer science

Department of Information Technology

Question Generation System For Persian Language using NLP Techniques

Supervisor: Assistant Professor Mrs. Tayeba Abidi

Author: Sakina Mosavi

Year……2019

Acknowledgement

In the name of God, the most kind and most merciful I would like to thank my parents who kept backing me up in all the times, both financially and morally.

I would also like to thank my supervisor Mrs. Tayeba Abidi for her guidance and encouraging me to work hard and smart. Her critical comments on my work have certainly made me think of new ideas and techniques in the field of my research and not only this her behavior taught me a lot to learn.

I would also like to thank the experts who were involved in this research area. Who inspired me by their research and ideas and put the resources open for other educators like me.

I am grateful to Allah who provides resources of every kind to me, so that I make their correct use for the benefit of mankind. May He keep providing us all these resources, and the guidance to help the humanity.

Thanks!

By regard:

Sakina Mosavi

Abstract

This thesis describes a rule-based approach for automatic question generation for Persian language using Natural Language Processing(NLP) Techniques. This thesis examines factual, natural language sentences to generate factual, natural language questions. This thesis relies on rules that look for patterns in the syntactic structures, named entities included in the sentence. And the extraction of these pieces of information from a sentence is not focused in this thesis, I use the exist destructing sentences elements methods which are already available for Persian natural language. Rather, this thesis examines how these pieces of information can be used for the purpose of factual, natural language question generation. The design and implementation of the proposed approach are also explained in detail and presents an evaluation of the output of the system. This system passes three steps which are normalization, content selection, and question formation, to form a question for the given text. This system generates three types of questions which are True/False, blank Space, and short descriptive questions.

Keywords— *Natural Language Processing for Persian Language, Rule-Based, Named Entity Recognition*

Contents

Chapter1: Introduction

1.1 Introduction:

Learning is one of the inseparable circumstances of human beings. Human beings are always in the process of learning this learning may be from every day's life adventures or by educational organizations or many other resources. If this learning process face with defect or incompleteness; just by asking questions we can complete our knowledge. For example in a learning class when a student doesn't catch a subject that was explained by lecturer they can complete their understanding by questions or by asking questions they can improve their knowledge about a subject. It means asking questions does an important role in daily learning environments.

Recently researches about automatic question generation systems have been done in many languages like the English language, Arabic language, and others. This research area has a process of evolution for these languages Which resulted in acceptable outcomes. But for the Persian language this research area is at the beginning level of the evolution, for this reason, there are many limitations and problems that we would face and I will list the limitations and problems that I faced in this thesis by detail later.

Most of the automatic question generation systems are rule-based recently some systems tackle with Machine Learning Algorithms also. Rule-based means the transformation of a sentence to question is through the use of rules or templates for the given input to the system. By Machine Learning algorithm means first we train the system by Machine Learning algorithms and then for the given input system will decide and transform the input sentence to question. This proposed system stands on a rule-based approach. Primarily this system focus on generating question from a sentence And on the given input some rules will be applied and the question will be ready. But generating questions from a paragraph is also applicable.

This automatic question generation system consists of three steps. First normalization of the sentence. Second content selection and third question generation. This system provides three types of questions. First, True and False questions. Second, blank space questions and third, short descriptive questions. These questions include the types of questions like when, who, where, what.

1.2 Research Problem

Generating questions for different purposes can be time-consuming; to decrease this problem this system is planned to be implemented. For example: when a teacher wants to evaluate the student's knowledge about a subject or want to asses his/ her performance on teaching; he/she can use this system. This system can help learners to find out the knowledge gaps that they have in a subject. And also this system can be useful for Question answering systems.

1.3 Research Question

This research is going to answer the following questions:

1. Are available NLP techniques for Persian language enough to develop question generation system?
2. How many kinds of questions are possible to be generated for question generation systems?
3. What challenges are there toward question generation system in Persian language?

1.4 Objective

Automatic Question Generation (AQG) is an important area of research in the fields of Artificial Intelligence (AI) and Natural Language Processing (NLP) and

is a key part of help systems and learning environments. For example, The ability to generate factual, natural language questions from a corpus of factual, natural language sentences would allow Automated Question Answering (AQA) systems like IBM Watson to perform self- training. And in learning environments as in universities it helps lecturers instead of wasting time to make questions for all the classes that they teach, they can invest their time in another purpose and use this system or just edit the questions as they want. Researches show that editing questions take less time than making new questions. And as we know; we have Kankor examination for entering universities in our country. It needs lots of stuff to make questions for this purpose, not only stuffs, it consumes the budget of the government. Or when a student does self-study, he/she will not notify how much he/she has learned. By answering questions he/she can examine him/her self and fulfill the knowledge gap. This system first Normalize the given sentence using a set of Normalization rules. Once the sentence has been normalized, this system would decide to apply is it possible to make a question from the sentence or not, if the answer is positive it goes to apply rules to transform the simplified sentences into questions. Both sets of rules are checked for patterns in the syntactic structure, named entities. To get these pieces of information, the system relies on the parser and Named Entity Recognizer (NER) from a website API which delivers this service.

1.5 What is Natural Language Processing?

Natural language processing (NLP) is a sub-field of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages. NLP talks about how to program computers to process and analyze large amounts of natural language data. NLP is an area of research and application that explores how computers can be used to understand and manipulate natural language text to do useful things. It promises for making computer interfaces that are easier to use for people since people will be able to talk to the computer in their own language, rather than to learn

a specialized language of computer commands. Its researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. History of Natural Language Processing(NLP) generally started in the 1950s by an article from Alan Turing about the Possibility of creating machines that can think. After that, a game was developed to play chess with human beings. In 1959 the first MIT AI Laboratory was set up. Then the Georgetown experiment in 1954 involved a fully automatic translation of more than sixty Russian sentences into English.

The techniques of natural language processing are here which are used in question generation systems.

Tokenization:

Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. These tokenizers work by separating the words using punctuation and spaces. It does not discard the punctuation to allow the user to decide what to do with the punctuation at the time of pre-processing. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph. Therefore we have two kinds of tokenizer as following:

Sentence Tokenizer:

Sentence tokenizer is the process of splitting the paragraph into sentences. For the English language, this process can be done by the nltk library. And for the Persian language, the Hazm library is available to tokenize the paragraph into sentences. For example:

“Hello everyone. Welcome to Afghanistan.” this paragraph can be split into two sentences. [“Hello everyone”,“welcome to Afghanistan”].

Word Tokenizer:

word tokenizer is the process of splitting the sentence into words. For the English language, this process can be done by nltk library. And for the Persian language, the Hazm library is available to tokenize the sentence into words. For example: “Hello everyone. Welcome to Afghanistan” will be split into [“Hello”, “everyone”, “welcome”, “to”, “Afghanistan”, “.”].

Parts of speech tagging:

parts of speech tagging is a process of labeling each word in a sentence with its appropriate part of speech is a list of tuples. A tuple form is (word, tag). The word is from the input sentence and the tag is from parts of speech tags which include nouns, verbs, adverbs, adjectives, pronouns, conjunction, and their sub-categories. For English language nltk has Penn Treebank tagger which is used to tag Parts of speech tagging of sentences. For example:

“its a pleasant day today” will be tagged as [(“it”, “PRP”),
(“is”,”VBZ”),
(“a”,”DT”),
(“pleasant”,”JJ”),
(“day”, “NN”),
(“today”, “NN”)]

the "PRP" represents a personal pronoun, the "VBZ" represents a singular verb, the "DT" represent determiner, the "JJ" represent adjective and "NN" represent noun.

Chunking:

chunking is also called shallow parsing which is a task that follows part-of-speech tagging and adds more structure to the sentence. The result is a grouping of the words in "chunks". In real it is the process of extracting phrases from unstructured text. Instead of just simple tokens that may not represent the actual meaning of the text, its advisable to use phrases such as "South Africa" as a single word instead of

"South" and "Africa" separate words. There are standards set of chunk tags like Noun Phrase(NP), Verb Phrase(VP), etc. For the English language, there are a lot of libraries that give phrases out-of-box such as Spacy or TextBlob. And NLTK just provides a mechanism using regular expressions to generate chunks. For example:

The “the little yellow dog barked at the cat” statement has two Noun Phrases(NP); “the little yellow dog” and “the cat”.

(NP the/DT little/JJ yellow/JJ dog/NN)

barked/VBD

at/IN

(NP the/DT cat/NN)

Dependency Parsing:

The sentence is an organized whole, the constituent elements of which are words. Between the words and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. The structural connection establishes dependency relations between the words. For example:

“India won the world cup by beating Lanka.”

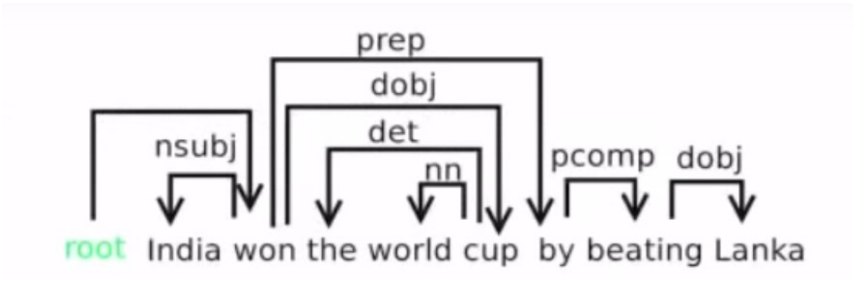

figure: output of Stanford dependency parsing

Named Entity Recognition:

Named entity recognition(NER) is one information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values,

percentages, etc. for example: The Named entity values of sentence "European authorities fined Google a record $5.1 billion on Wednesday for abusing its power in the mobile phone market and ordered the company to alter its practices" are here.

[(“European”, “NORP”,

(“Google”, “ORG”),

(“$5.1 billion”, “MONEY”),

(“Wednesday”, “DATE”)]

The expression NORD means nationalities or religious or political groups). In this example the European is a NORP, Google is an Organization, the $5.1 billion is money and the Wednesday is a date which is extracted from the above sentence.

Semantic Role Labeling:

The semantic role labeling is also called shallow semantic parsing or slot-filling is the process that assigns labels to words or phrases in a sentence that indicate their semantic role in the sentence, such as agent, goal, or result. It consists of the detection of the semantic arguments associated with the predicate or verb of a sentence and their specific roles. For example, for the sentence "Mary sold the book to john", the task would be to recognize the verb "to sell" as the predicate, "Mary" as representing the seller(agent), "the book" as representing the goods(theme), and "John" as representing the recipient. Actually, this is an important step for making sense the of the meaning of the sentence.

Chapter2: Literature review

2.1 previous works on automatic question generation systems

Recently, automatic question generation systems have got the attention of researchers. Many researchers have presented their work in this area of research and they are still working in this area to achieve higher accuracy. As I searched, using natural language processing techniques for English language prior works have been done for automatic question generation systems with different approaches. But I couldn't find any evidence that shows research in this field for the Persian language or Farsi language. I got inspired by the English language researches and implementations that I searched for. Now I want to explain briefly the papers or researches I have read.

As evidence shows in 1976 Wolfe worked on Question Generation and mentioned that automatically-generated questions can be as effective as human-generated questions. After that, researches are done by Kunichika in 2004, by Mitkov in 2006 and by Russet in 2007. In 2009 the first workshop on the Question Generation Shared Task and Evaluation Challenge was taken which cause get enormous attention from the researchers. In 2011, Heilman addressed various question generation challenges in his paper. Like this, till now 2019 many research has been done in this area by using Natural Language Processing techniques. Here I want to demonstrate by detail some of these researches that have been done within these years.

2.2 Question Generation from Paragraph at Upenn by Prashanth Mannem, Rashmi Prasad, and Aravind Joshi at the University of Pennsylvania in 2010:

This system uses predicate-argument structures of sentences with semantic roles for the question generation task from paragraphs. The semantic role labels are used to

identify relevant parts of the text before forming questions on them. The generated questions are then ranked to pick the final six best questions. They used semantic role labels(SRL parse) provided by ASSERT. For each sentence, ASSERT provides multiple predicate-argument structures for all the predicates in the sentence. The Semantic Role Labeling identifies semantically meaningful parts for predicates in a sentence with their semantic roles. These arguments can be used to ascertain whether a particular part is appropriate for generating a question or not.

For example identification of ASSERT system for "One-handed backhand players move to the net with greater ease than two-handed players" is:

[ARG1 One-handed backhand players] [PRED move] [ARG2 to the net] [ARGM-MNR with greater ease than two-handed players] [ARGM-CAU because the shot permits greater forward momentum and ⋯]

In the ASSERT system the semantic roles from ARG0…… ARG5 are mandatory arguments and the roles starting with ARGM- are optional arguments. The ASSERT system can't handle copular verbs which are important for question generation. For this reason, they use dependency parsing and combining the result with the SRL parse.

This system includes three stages which are: Content Selection, Question Formation, Ranking**.**

Content Selection

In the case of question generation, content selection is the text over which the question has to be asked. For a general question, the content is the complete paragraph. In the case of a medium scope question, the content is typical spans 2-3 sentences in the paragraph. In this paper, SRL parses are used for content selection. The ASSERT system is applied over all the sentences in a paragraph to obtain the predicates, their semantic arguments and the semantic roles for the arguments. The kinds of predicates which are as the output will be explained also here. Then this

information is used to identify the possible target content for a question. The target for a question is the text over which the question is to be asked.

Mandatory arguments

Questions can be posed over the mandatory arguments of a predicate. A mandatory argument for this system is an argument with one of the six semantic roles {ARG0 · · ·ARG5 }. For instance, in Example one, the question can be form over ARG1 of move. Who moves to the net with greater ease than two-handed players? (Ans: one-handed backhand players). On ARG2, the question could be Where do one-handed backhand players move to with greater ease than two-handed players? (Ans: to the net). Mandatory arguments of all predicates all sentences of the paragraph are considered as possible targets and passed over to the question formation stage. This method of identifying targets generates more than one target for a sentence and therefore a large number of targets for a paragraph. Predicates with less than two arguments are ignored since a question cannot be formed on just one argument.

Optional arguments

Optional arguments can also be a good choice to target selection for the question. Arguments with roles {ARGM-MNR, ARGM-PNC, ARGM-CAU, ARGM-TMP, ARGM-LOC, ARGM-DIS } are optional arguments and good candidates for being a target. In Example 1 which is on the above paragraph, the question on ARGM-CAU of move would be Why do one-handed backhand players move to the net with greater ease than two-handed players? Table 1 gives a description of these roles and the question type associated with each of the roles.

Copular verbs

The copular verbs are different from the other verbs because the semantic role labeler doesn't give a predicate-argument structure for them. They use the dependency parse to determine the arguments of a be verb. The right argument of the verb is almost a good target in this system for a question or the sentence is

existential (ex: there is a · · ·). For instance, in the sentence "Latent semantic analysis is a technique in natural language processing, in particular, · · ·", the right argument of the verb "a technique in

natural language processing, · · ·" can be a good target to make question on. The question will be "What is Latent semantic analysis (LSA)?" The left arguments for copular verbs are very complex to generate questions on. Only the right arguments can be passed as targets to the next stage.

Question Formation

The targets along with the ASSERT analyses of the sentences are passed over from the previous stage to form questions on them. In this stage, the verb complex for each sentence is first identified and then a series of transformations are applied for every sentence corresponding to a target to generate the final question.

Ranking

While the system accumulates a list of targets after the content selection stage, it generates questions for these targets in the question formation stage. In the final stage, this set of questions is ranked to select the best 6 questions. The list of questions is ranked in two steps. In the first step, the questions are ranked by the depth of the predicate in the dependency parse. The semantic role labeler gives us predicate-argument structures for all the predicates in the sentence, questions are also generated from unimportant predicates in complex long sentences. This ranking method makes sure that the questions generated from main clauses get a higher rank than the ones generated from subordinate clauses.

In the second step, the questions with the same rank are ordered by the number of pronouns occurring in the question. Since the system doesn't handle coreferences, questions containing pronouns are given a lower score. Finally, 6 questions are selected from the ranked list based on the task requirements.

2.3 AUTOMATIC QUESTION GENERATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES by Onur KEKLİK in July 2018 in İZMİR Turkey.

This work is a rule-based approach to question generation from a sentence. Onur KEKLİK used three kinds of templates to generates shallow and deep questions into stages. Shallow questions like what, who, where, how many, when and deep questions like why, how, how would you describe, indicates characteristics of, for what purpose. And the two stages are the deconstruction stage and construction stage. In deconstruction stage the sentence is divided into components and the parsers are used to label these components and in the construction stage the sentence is matched to the available templates and the question will be generated according to the matching templates.

2.2.1 Dependency based templates

First, the Semantic Role Labeling is applied to the sentence and then the Dependency Parsing is applied to the sentence. The semantic arguments are matched with the found dependency tags to check if the sentence template corresponds to any dependency template. The dependency-based template has S-V-acomp (adjectival complement), S-V-attr (attribute), S-V-dobj (direct object), S-V-pcomp (complement of a preposition) and S-V-dative (three entities) S-V xcomp (open clausal complement) and S-V-ccomp (clausal complement) S-V-oprd (object predicate) templates which are all one option to create question from these templates.

Here I represent an example for some of these templates.

1. S-V-oprd(object predicate) which is an object predicate that defines the subject.

S: Brain waves during REM sleep appear similar to brain waves during wakefulness.

Q: How would you describe brain waves during REM sleep?

2. S-V-pcomp(complement of a preposition) is a complement of a preposition that modifies the meaning of a prepositional phrase.

S: REM sleep is characterized by darting movement of closed eyes

Q: What is REM sleep characterized by?

2.2.2 NER based templates

First, the Semantic Role Labeling is applied to sentence and then the Named Entity Recognition, and the system detects words that are labeled as person, location, date or number. Therefore the system can ask who, where, when and how many questions according to the template. Then, the tagged word is removed from the semantic arguments. Finally, using the rest of the semantic arguments, the system checks if the constructed sentence has components like the subject, direct object, verb, etc.

Templates and Example

1. S-V-number.

S: In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area.

Q: How many staff did the trust employ in 1996?

2. S-V-location.

S: In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area.

Q: Where did the trust manage another six sites in 1996?

3. S-V-date.

S: In 1996, the trust employed over 7,000 staff and managed another six sites in Leeds and the surrounding area.

Q: When did the trust employ over 7,000 staff?

4. S-V-person.

S: The fourth Yuan emperor, Buyantu Khan (Ayurbarwada), was a competent emperor.

Q: Who was a competent emperor?

2.2.2 Semantic Role Labeling based templates

In this template, the Semantic Role Labeling is applied to the sentence and the arguments can be as following:

• ARGM-CAU is a cause clause and indicates the reason of action. We conclude that

asking the "why" question is suitable. For example:

S: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit.

Q: Why was Tesla returned to Gospic on 24 March 1879?

• ARGM-MNR is a manner marker and specifies how the action is performed. We

conclude that asking the "how" question is suitable. For example:

S: Tesla’s work for Edison began with simple electrical engineering and quickly progressed to solving more difficult problems.

Q: How did Tesla’s work for Edison progress quickly to solving more difficult problems?

• ARGM-LOC indicates where some action takes place. For example:

S: Mr. Bush met him privately, in the White House, on Thursday.

Q: Where did Mr. Bush meet him on Thursday?

• ARGM-TMP shows when the action took place.

S: Mr. Bush met him privately, in the White House, on Thursday.

Q: When did Mr. Bush meet him in the White House?

2.4 Automated Generation of Questions from Factual, Natural Language Sentences by Tyler Yates in 2016

This paper explains methods for examining factual, natural language sentences to generate natural language questions whose answers will be found in the sentences. These methods searches for patterns in the syntactic structure, named entities, and grammar relations of a sentence. Tyler named the implementation of these methods in the form of a computer program as Specimus.

Specimus first simplify the given sentence using a set of simplification rules. When the sentence has been simplified, Specimus will apply the rules to transform the simplified sentences into grammatical questions. These two sets of rules search for patterns in the syntactic structure, named entities, and grammatical relations of a sentence. Specimus relies on the parser and Named Entity Recognizer (NER) from Stanford's CoreNLP toolkit.

This program only generates concept completion (who, what, when, and where) questions. The explanation of this program is the following:

2.3.1 Sentence Simplification

Factual sentences contain many pieces of information within various syntactic constructions. By extracting these syntactic constructions into separate sentences, Specimus can create syntactically simpler sentences. This caused to create question generation rules that are more correct and simpler. All sentences that Specimus takes in as input are first passed to sentence simplification system. The resulting simplified sentences output are then passed on to the question generation system. For example, if the parenthetical consists of two DATEs, it assumes that the dates are about the person's birth and death since this is a very common convention in English. Two simplified sentences are then generated using the NP containing the PERSON word and the two DATEs. Like "George Washington (1732 – 1799) was the first

President.". The PERSON NP is "George Washington" and the DATE NPs are “1732” and “1799”. Thus, two simplified sentences can be generated:

1. George Washington was born in 1732.

2. George Washington died in 1799.

After a sentence is passed through the sentence simplification system, the resulting simplified sentences are passed to the question generation system. The question generation system uses defined rules to potentially generate questions for each sentence. Each rule is written using a structured language. Every rule takes in a sentence and returns a set of natural language questions. The returned set may be empty if the rule never activates or passes its preconditions. To retrieve all questions that a given sentence answers, the sentence is passed to each rule and all of the resulting sets are combined into a single answer set. By using a set the system need not to worry about duplicate questions that are generated by the various rules. The question generation system was designed this way to maximize the question coverage of the system: Because every rule is given a chance to generate questions for each input sentence, the system does not have to worry about questions being potentially missed. The downside to this approach is reduced performance as every rule is given the chance to examine each input sentence.

For example, one rule is as:

NP-VP Rule:

Function Definitions

wh(x) – Generates ‘who’ if the head of x refers to a person, ‘what’ otherwise

passiveIfNecessary(v) – Transforms the given VP into passive voice, if possible, if the VP contains the preposition ‘by’ before the first NP.

Preconditions

None

Generate

wh(NP-1) + passiveIfNecessary (VP-1)

as these rules have been written to generate questions from the simplified questions.

Specimus has been evaluated on two hypotheses:

1. The generated questions are grammatically correct.

2. The original sentence answers the generated questions.

For this, human judges were asked to evaluation and rate the output of Specimus on a set of sentence-question pairs.

Twelve human judges were each given twenty sentence-question pairs to evaluate the two hypotheses. This allowed each sentence-question pair to be evaluated by three different judges. The judges were instructed to use a 1-5 numeric scale when evaluating the two hypotheses. The scales for each of the hypotheses are given below. The result is as following for each hypothesis.

Result for hypotheses 1: “The generated question is grammatically correct”.

Rating	Responses
5 – Very Good	147
4 – Good	34
3 - Okay	38
2 - Bad	17
1 – Very Bad	4

Result for hypotheses 2: “The original sentence answers the generated question”.

Rating	Responses
5 – Very Good	117
4 – Good	64
3 - Okay	38
2 - Bad	17
1 – Very Bad	4

Chapter3: Methodology

3.1 What is research?

research is used to refer to the activity of a diligent and systematic inquiry or investigation in an area, with the objective of discovering or revising facts, theories, applications etc. The goal is to discover and disseminate new knowledge. There are several methods that can be used in CS. Here is explanation of some research methodologies.

3.2 Experimental Method

Experimental shows the experiments that will occur in order extract results from real world implementations. Experiments can test the veracity of theories. This method within CS is used in several different fields like artificial neural networks, automating theorem proving, natural languages, analyzing performances and behaviors, etc. It is important to restate that all the experiments and results should be reproducible. Concerning, for example, network environments with several connection resources and users, the experiments are an important methodology Also in CS fields and especially IS fields that take in consideration the Human-Computer Interaction. It is mandatory the usage of experimental approaches. If we use the experimental method in IS field we may need to use some methods or tools in conjunction with the experimental method. These methods or tools used to support and prove the legibility of the developed project. For example if a student wants to implement new social network with new concepts or develop an existing social network, who he can measure the legibility of his implementation? The answer of this question consists of two parts according to the nature of the project. The technical issue is the first part of the project that can be tested by benchmarks like (availability, reliability, scalability, stability, , etc). The usability of the project is the second part of testing that needs a feedback from the users of the system; the results for the second part can be obtained by the statistical analysis of a questionnaire

which is a tool the used in conjunction with the experimental method. Experimental method has two variables: Dependent variable and Independent variable

Independent variable

A factor or condition that is deliberately manipulated to determine whether it causes any change in another behavior or condition.

Dependent variable

A factor or condition that is measured at the end of an experiment.

3.3 Simulation Method:

Simulation method used especially in CS because it offers the possibility to investigate systems or regimes that are outside of the experimental domain or the systems that is under invention or construction. Normally complex phenomena that cannot be implemented in laboratories evolution of the universe. Some domains that adopt computer simulation methodologies are sciences such as astronomy, physics or economics; other areas more specialized such as the study of non-linear systems, virtual reality or artificial life also exploit these methodologies. A lot of projects can use the simulation methods, like the study of new developed network protocol. To test this protocol you have to build a huge network with a lot of expensive network tools, but this network can't be easily achieved. For this reason we can use the simulation method.

3.4 Theoretical Method

The theoretical approaches to CS are based on the classical methodology since they are related to logic and mathematics. Some ideas are the existence of conceptual and formal models (data models and algorithms). Since theoretical CS inherits its bases from logic and mathematics, some of the main techniques when dealing with problems are iteration, recursion and induction. Theory is important to build methodologies, to develop logic and semantic models and to reason about the programs in order to prove their correctness. Theoretical CS is dedicated to the

design and algorithm analysis in order to find solutions or better solutions (performance issues). Encompassing all fields in CS, the theoretical methodologies also tries to define the limits of computation and the computational paradigm. In other words we can say that we can use the theoretical method to model a new system. However the theoretical method can help in finding new mathematical models or theories, but this method still needs other methods to prove the efficiency of the new models or theories. For example when a student need to develop a new classifier in AI by using the mathematical representation and theoretical method, he need to prove the efficiency of this model by using one of the previous methods.

3.5 why experimental method?

This system follows experimental methodology, because its the methodology that can be used in computer Science. In experimental method the human-computer interaction is mandatory and this system rely on human-computer interaction. The independent variable is the sentence on which question is going to be generated and the dependent variable is the question(s) that are generated by this system.

3.6 Business process model:

Business process modeling in business process management and system engineering is the activity of representing process of an enterprise, so that the current process may be analyzed, improved and automated. BPM is typically performed by business analysis, who have specialized knowledge of the process being modeled; or more commonly by a team compromising both. Alternatively, the process model can be derived directly from events log using process mining tools.

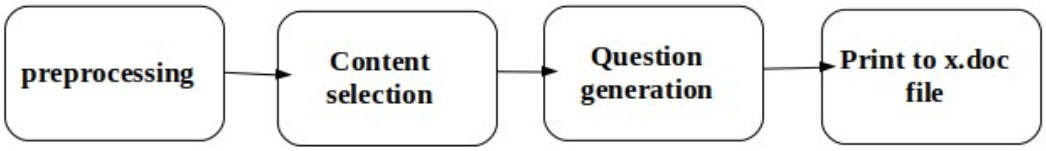

Figure 3-1: Business Process model annotation

Chapter 4: Implementation

4.1 Implementation

In this section, the generation of question from the given sentence is explained in detail. This system is designed just for normal Persian text to generation not for fiction, idioms and etc. This system focus on generating questions from sentences which are no longer then 3000 characters but the questions from a paragraph also can be implemented such that each paragraph is divided into separate sentences and then the question on single sentence will be applied. This operation has four steps which include pre-processing, content selection, question generation and writing in the word file with the given name for the file. This system can generate three kind of questions. First true/false questions, blank space questions, and short descriptive questions. Each step will be explained by detail in this chapter.

4.2 Getting data set

The data set for this system can be a normal Persian text and I choosed the first, second and third class Dari subject of the school book. And for testing I have choose text from Wikipedia, google and text-mining website.

4.3 Preprocessing

The sentence which is given to the system may not be normalized and may have spacing issues and Arabic or English numbers. For example some words in Persian need single space between words and some words need half space between words, this step will rectify the space issue and convert the Arabic or English number to Persian numbers and the sentence will be standardized. This will be done by using API of text-mining website. Because there is no available library for this purpose.

At the end of normalization of the sentence the dots from the end of the sentence will be removed. Example:

“خانم احمدی هر روز ساعت 8 در صنف حاضر میشود.”

Here the Arabic or English number(8) will be changed to Persian number format “۸” and the “میشود ” will be written as “می شود” with half space between them. At the end the dot(.) will be removed from the end of the sentence. Here is the python code for it.

4.4 Content Selection

Content selection means selection of a sentence for generating question on it. If the given sentence is capable of generating question it will select the sentence, if not it will not select the sentence for generating question. If our input is a paragraph, the content selection is also able to identify the sentence or sentences that are capable of generating question on it. Such that first the paragraph will be split into separated sentences and the this operation will be done on each sentence of the paragraph. The selection is based on NER(named entity recognition) parse of sentence, POS(parts of speech) tagging and defined rules. For English language semantic role labeling parser is the most used parser for sentence selection but unfortunately for Farsi or Persian language we don’t have that available library or any other alternative so for implementation of this system I contentment for POS tagging and NER libraries which are available for Farsi/Persian language.

4.4.1 number existence

If a sentence has a numerical value, the value of that number will be a target to make question on it.

4.4.2 person existence

If a sentence has a person name in its content, the person name will be a target to generate question on it.

4.4.3 location existence

If a sentence has a location name in its content, the location name will be a target to generate question on it.

4.4.5 organization existence

If a sentence has an organization name in its content, the organization name will be a target to generate question on it.

4.4.5 event existence

If a sentence has an event name in its content, the event name will be a target to generate question on it.

4.4.6 date existence

If a sentence has a date in its content, the date will be a target to generate question on it.

4.5 Question Generation

After passing content selection step there is question generation step based on content selection. This system try to generate all possible questions that can be generated from the sentence and its the users choice which one they prefer. This system generates three kinds of question true/false questions, blank space questions and short descriptive questions. Here the process of generating question from the given sentence is explained by detail.

4.5.1 True/False Question

4.5.1.1 number existence

when the POS tagging parser detects a number in the sentence as question target, the value will be changed randomly to a number near that value. The this new value will be replaced with the existing value in the sentence. The code is here:

```
def changing_int_value_TF(text):
    output = tagger.tag(word_tokenize(text))
    for ind, o in enumerate(output):
        payload = u"\"" + text.__add__("؟") +"\""
        try:
            if o[1] =="NUM":
                Matching_value = o[0]
                if int(Matching_value) >= 0 | int(Matching_value) <= 100:
                    Result = int(int(Matching_value) + random.randrange(100))
                    r =str(Result)
                    return payload.replace(o[0], r)
                if int(Matching_value) >100:
                    R = int(Matching_value) /16 + random.randrange(10)
                    Result = int(int(Matching_value) - R)
                    r = str(Result)
                    return payload.replace(o[0], r)
                break;
        except:
            return False
```

4.5.1.2 valuable sentence

when the sentence is passed through NER parser, this parser detects if the sentence contain any person name, location name, organization name, event name or date, if the given sentence contains any of this the sentence is valuable to be as question. This sentence will be sent to other step. The code is here:

```
def Any_TF(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url = baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result = json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-PER":
            return payload
        if phrase['tags']['NER']['item1'] == "I-PER":
            return payload
        if phrase['tags']['NER']['item1'] == "B-LOC":
            return payload
        if phrase['tags']['NER']['item1'] == "I-LOC":
            return payload
        if phrase['tags']['NER']['item1'] == "B-ORG":
            return payload
        if phrase['tags']['NER']['item1'] == "I-ORG":
            return payload
        if phrase['tags']['NER']['item1'] == "B-DAT":
            return payload
        if phrase['tags']['NER']['item1'] == "I-DAT":
            return payload
```

4.5.2 Blank space questions

After the sentence passes NER parser in content selection step it will be sent to the blank space question generation step. The different situations are considered here.

4.5.2.1 number existence

First the sentence will be word tokenize and then POS will be applied to the separated words when the parser detects the existence of a number the number will be replaced with a group of underscores(_). For example:

```
def Int_BS(text):
    text = text.__add__("؟")
    output = tagger.tag(word_tokenize(text))
    for ind, o in enumerate(output):
        if o[1] =="NUM":
            Matching_value = o[0]
            value = u"\"" + text +"\""
            withDash = value.replace(Matching_value, "_____", 1)
            return withDash
```

4.5.2.2 person existence

After the sentence is passed through NER tagging, the person name existence will be marked and the person name will be replaced with dashes.

```
def Per_BS(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url = baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result = json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-PER":
            payload = payload.replace(phrase['wordComment'],"_____", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-PER":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-PER":
            payload = payload.replace(phrase['wordComment'], "_____", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-PER":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
```

4.5.2.3 location and organization existence

The input sentence passes NER tagging which cause tagging of elements according to NER tagging parser labeling. And instead of any location name there will be used dashes which ask the user to add the correct value.

```
def Loc_BS(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url =  baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result =  json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-LOC":
            payload = payload.replace(phrase['wordComment'], "__________", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-LOC":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-LOC":
            payload = payload.replace(phrase['wordComment'], "_________", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-LOC":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
```

4.5.2.4 event existence

The input sentence first passes NER tagging and instead of an event there will be used dashes which ask the user to write the correct values.

```
def Eve_BS(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url =  baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result =  json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-EVE":
            payload = payload.replace(phrase['wordComment'], "_________", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-EVE":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-EVE":
            payload = payload.replace(phrase['wordComment'], "_________", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-EVE":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
```

4.5.2.5 date existence

if we have a date in our sentence instead of the date the system will generate dashes to ask the user write the correct answer.

```
def Dat_BS(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url = baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result = json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-DAT":
            payload = payload.replace(phrase['wordComment'], "__________", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-DAT":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-DAT":
            payload = payload.replace(phrase['wordComment'], "_________", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-DAT":
                    payload = payload.replace(phrase['wordComment'],"", 1)
                    break
            return payload
```

4.5.3 short descriptive question

short descriptive questions represent the questions whose answers are short and actually are recap of an idea. Kinds of this questions are considered here and will be explained here.

4.5.3.1 number existence

By using Hazm the sentence will be word tokenize and it will be pass through POS tagging, if POS tagging recognize the existence of a numeric value, the system will generate question with “چند”.

```
def Int_D(text):
    text = text.__add__("؟")
    output = tagger.tag(word_tokenize(text))
    for ind, o in enumerate(output):
        if o[1] =="NUM":
            Matching_value = o[0]
            value = u"\"" + text +"\""
            withDash = value.replace(Matching_value, "چند", 1)
            return withDash
```

4.5.3.2 person existence

First when the system is parsed through NER if the sentence has a person noun, it will be the target of our system and the system will ask question with “چه کسی”.

```
def Person_Noun(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url =  baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result =  json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-PER":
            payload = payload.replace(phrase['wordComment'], "چه کسی",1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-PER":
                    payload = payload.replace(phrase['wordComment'], "",1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-PER":
            payload = payload.replace(phrase['wordComment'], "چه کسی",1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-PER":
                    payload = payload.replace(phrase['wordComment'], "",1)
                    break
            return payload
```

4.5.3.2 location existence

When the system is parsed through NER if the sentence has a location noun, it will be the target of our system and the system will ask question with “کجا”.

```
def Location_Noun(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url = baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result = json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        print(phrase['tags']['NER']['item1'])
        if phrase['tags']['NER']['item1'] == "B-LOC":
            payload = payload.replace(phrase['wordComment'], "کجا",1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-LOC":
                    payload = payload.replace(phrase['wordComment'], "",1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-LOC":
            payload = payload.replace(phrase['wordComment'], "کجا" , 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-LOC":
                    payload = payload.replace(phrase['wordComment'], "", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "B-ORG":
            payload = payload.replace(phrase['wordComment'], "در کجا", 1)
            for phrase in Result:
                 if phrase['tags']['NER']['item1'] == "I-ORG":
                    payload = payload.replace(phrase['wordComment'], "", 1)
                    break
            return payload
```

4.5.3.3 event existence

When the system is parsed through NER if the sentence has an event noun, it will be the target of our system and the system will ask question with “ کدام رویداد”.

```
def Event(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url =  baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result =  json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-EVE":
            payload = payload.replace(phrase['wordComment'], "کدام رویداد", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-EVE":
                    payload = payload.replace(phrase['wordComment'], "", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-EVE":
            payload = payload.replace(phrase['wordComment'], "کدام رویداد", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-EVE":
                    payload = payload.replace(phrase['wordComment'], "", 1)
                    break
            return payload
#
```

4.5.3.4 date existence

If NER parser finds the existence of date in the given input sentence, there is two possible ways of generating questions. First the date will be replace with the “چه زمانی” and other we can take the date and generate a question that will ask what happened in this specific date. The codes are here:

First possibility:

```
def Date_Noun(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url =  baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result =  json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-DAT":
            payload = payload.replace(phrase['wordComment'], "چه زمانی", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-DAT":
                    payload = payload.replace(phrase['wordComment'], "", 1)
                    break
            return payload
        if phrase['tags']['NER']['item1'] == "I-DAT":
            payload = payload.replace(phrase['wordComment'], "چه زمانی", 1)
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-DAT":
                    payload = payload.replace(phrase['wordComment'], "", 1)
                    break
            return payload
```

Second possibility:

```
def Date_D(text):
    baseUrl = "http://api.text-mining.ir/api/"
    tokenKey = APISetup.tokenkey()
    url =  baseUrl + "NamedEntityRecognition/Detect"
    payload = u"\"" + text.__add__("؟") +"\""
    Result =  json.loads(APISetup.callApi(url, payload, tokenKey))
    for phrase in Result:
        if phrase['tags']['NER']['item1'] == "B-DAT":
            firstP =  phrase['wordComment']
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "I-DAT":
                    secondP = phrase['wordComment']
                    complete = firstP + secondP
                    payload = "در" +  complete + "چه اتفاقی افتاد؟ "
                    return payload
        if phrase['tags']['NER']['item1'] == "I-DAT":
            firstP =  phrase['wordComment']
            for phrase in Result:
                if phrase['tags']['NER']['item1'] == "B-DAT":
                    secondP = phrase['wordComment']
                    complete = firstP + " " + secondP
                    payload = "در" +  complete + "چه اتفاقی افتاد؟"
                    return payload
```

4.5.3.5 Writing Question to a word file

When the question generating processes have been finished, its time to show the questions to the user. Questions will be outputted to a word file. There will be a title for true/false question, a title for blank space questions and a title for short descriptive questions.

4.6 Flow chart of system

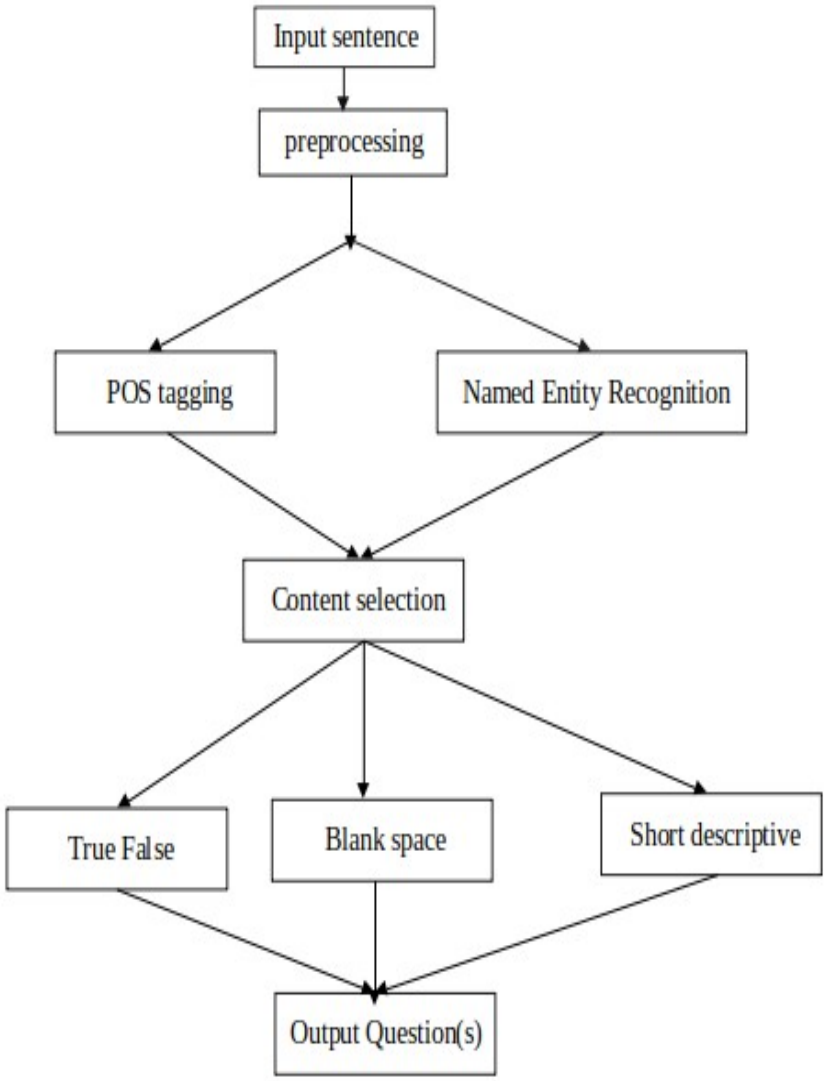

figure 4.1 flowchart

Chapter 5: Result

Automatic question generation is in its first steps. One of limitations is that there isn't enough available Parser for Dari language. Some available parsers are trained with Farsi language and it can't detect Dari keywords which are special words in Dari language. For now only Hazm library and the API of text-mining website provide some parser which are used in this system.

This system can generate question that contains when, who, where, how many questions in three kind of questions true/false questions, blank space questions and short descriptive questions. Because there isn't any available semantic role labeling parser for Persian or Farsi languages this system can' generate questions like why and how questions.

This system focus on generating questions from sentences which are no longer than 3000 characters. These sentences don't include idioms, Dialogs and fictions. If the given input is a paragraph, the system will change it to separated sentences and then behave with each sentence as a single input.

Because the system focus is on generating questions from a single sentence it can't detect the pronouns related to a name in previous sentences, which is a big issue for automatic question generation systems. Till now, there isn't any solution for this purpose not even in English language.

5.1 Accuracy values

For checking accuracy of this system I preferred human evaluation. It means Human judges rated the generated questions, such that I gave 15 sentence-question pairs to 20 human judges to rate in 1-5 range. Which shows acceptable result though automatic question generation systems for Farsi/Persian language which is in first steps of evolution.

Rating	Responses
5 - Very Good	139
4 - Good	55
3 - Okay	71
2 - Bad	25
1 - Very Bad	10

Table 5.1 accuracy values

5.2 Samples of questions generated by this system and evaluated by human judges

S1: "در ۲۸ اسد روز استقلال وطن عزیز ما افغانستان است که در زمان امان الله خان به دست آمد"

Q1: "در 118 اسد روز استقلال وطن عزیز ما افغانستان است که در زمان امان الله خان به‌دست آمد؟"

S2: "اولین کسی که از مردان، پیامبری را تصدیق نمود و ایمان آورد، حضرت ابوبکر صدیق بود "

Q2: "_____ کسی که از مردان، پیامبری را تصدیق نمود و ایمان آورد، حضرت ابوبکرصدیق بود؟"

S3: "حضرت ابوبکر صدیق از جمله یاران حضرت محمد بود"

Q3: "چه کسی صدیق از جمله یاران حضرت محمد بود؟"

S4: "حضرت محمد(ص) آخرین پیامبران است"

Q4: "چه کسی آخرین پیامبران است؟"

S5: "حضرت محمد(ص) پیامبر ما است"

Q5: "حضرت محمد کیست مختصرا معلومات دهید؟"

S6: "حبیب بورقیبه در ۶ آوریل ۲۰۰۰ در سن ۹۶ سالگی درگذشت. ـ"

Q6: "چه کسی در ۶ آوریل ۲۰۰۰ در سن ۹۶ سالگی درگذشت؟"

S7: "حبیب بورقیبه در ۶ آوریل ۲۰۰۰ در سن ۹۶ سالگی درگذشت. "

Q7: "حبیب بورقیبه در ۶ آوریل چند در سن ۹۶ سالگی درگذشت؟"

S8: "احمدعباسی در ایران، انگلستان و آمریکا سفر کرده است"

Q8: "احمدعباسی در ـــــــ، ـــــــو ـــــــ سفر کرده است؟"

S9: "احمدعباسی در ایران، انگلستان و آمریکا سفر کرده است"

Q9: "احمدعباسی" کیست مختصرا معلومات دهید؟"

S10: "روزنامه سیمرغ در سال ۱۳۵۴ تاسیس شد"

Q10: "روزنامه سیمرغ در چه زمانی تاسیس شد؟"

S11: "افغانستان در قرن ۲۱ عضویت سازمان ملل متحد را بدست اورد. ـ"

Q11: "افغانستان در قرن ۴۸ عضویت سازمان ملل متحد را بدست اورد؟"

S12: "افغانستان در قرن ۲۱ عضویت سازمان ملل متحد را بدست اورد. ـ"

Q12: ""افغانستان در چه زمانی عضویت سازمان ملل متحد را بدست اورد؟""

S13: "وزارت مخابرات در مرکز شهر کابل موقعیت دارد.ـ"

Q13: "وزارت مخابرات در مرکز کدام شهر موقعیت دارد؟"

S14: "دیروز هوا ابری بود. ـ "

Q14: "چه زمانی هوا ابری بود؟"

S15: "افغانستان در قرن ۲۱ عضویت سازمان ملل متحد را بدست اورد. ـ"

Q15: ""افغانستان در ـــــــــ عضویت سازمان ملل متحد را بدست اورد؟"

Chapter6: Discussion and Outlook

As I searched automatic question generation for Persian/Farsi language using natural language processing isn’t done or I couldn't find evidence to show that researches have been done in this field. But for English language its almost 20 years that researchers work in this field and they reach to higher performance and accuracy values.

6.1 Similarities

Similarities between this system and existed systems in English language are that, the existed systems can generate questions that can question through who, when, where and how many. And this system is also able to generate questions that starts with who, when, where, and how many questions. Not only who, when, where and how many questions this system is able to generate questions that has which organization and which event. That may not be in exists English language systems. In English language one kind of question is yes/no question, but the same concept is applied for this system with true/false question.

6.2 Differences

First difference will be the languages that, this system is for Persian/Farsi language and the papers I read are in English language. Second the libraries which are used for parsing purposes to tag the elements of the sentences. One difference is that, this system is based on Hazm and NER parser but other English existed systems are based on NER, SRL, Dependency parsing. The existed English systems can generate questions that has why, and how but this system can’t generate this kind of questions because for Persian/Farsi language we don’t have SRL parser which is needed for this purpose. One big difference is that, automatic question generation for English language has improved and they get higher accuracy and they also have evolution workshops after some years to discuss and share their idea and improve

existing systems. But this kind of systems for Persian/Farsi language is in its first steps. It needs more effort to improve and get higher accuracy. One positive point of Persian language is that, it doesn’t have auxiliary verbs and there is no need of using auxiliary verbs to generate questions. But in English language most times we need auxiliary verbs to generate questions. For example:

S : John won the football world cup in 2018.

Q: who did won the football world cup in 2018.

Chapter7: Conclusion and Future Work

This research presented a rule based system for generating question using natural language processing for Persian language. This system can generate questions from a single sentence and a paragraph. At the end for checking accuracy we have evaluated this system by human evaluators which shows significant result.

This system rely on parsers named Hazm and API of text-mining website for parsing the sentence elements and using these elements for generating question. Hazm is a library for Farsi language in natural language processing field. Hazm contains sentence tokenizer, word tokenizer, pos tagging and some other functions for Farsi/Persian language And because till now we don't have any standard library for NER parsing, for NER parsing purpose I used API of text-mining website which serve this application.

As this system focus is generating questions from sentences, it can't recognize the pronouns that relate to a specific noun. Till now its also not possible for English language to detect the pronouns that relate to specific nouns.

Currently, this system can't achieve best performance for sentences that have Persian specific words question because the available libraries and APIs are trained with Farsi language and don't contain special words, name of places and many other terminologies.

Now question generation for Farsi/Persian language is in its first steps of evolution and it need many effort to improve the performance and accuracy of this research area.

As mentioned this system can generate questions from sentences and paragraphs. The sentences shouldn't be longer than 3000 characters. And for future work my plan is to develop a website that delivers the service which can generate questions

not only for sentence and paragraphs but also from a pdf file. And output the questions to a word file with greater accuracy and performance.

References

[1] Yates, T. (2016). Automated generation of questions from factual, natural language sentences.

[2] Keklik, O. (2018). *Automatic question generation using natural language processing techniques* (Master's thesis, Izmir Institute of Technology).

[3] Rakangor, S., & Ghodasara, Y. (2015). Literature review of automatic question generation systems. *International Journal of Scientific and Research Publications*, *5*(1), 1-5.

[4] Mannem, P., Prasad, R., & Joshi, A. (2010, June). Question generation from paragraphs at UPenn: QGSTEC system description. In *Proceedings of QG2010: The Third Workshop on Question Generation* (pp. 84-91).

[5] Le, N. T., Kojiri, T., & Pinkwart, N. (2014). Automatic question generation for educational applications–the state of art. In *Advanced Computational Methods for Knowledge Engineering*(pp. 325-338). Springer, Cham.

[6] Rakangor, S., & Ghodasara, Y. (2015). Literature review of automatic question generation systems. *International Journal of Scientific and Research Publications*, *5*(1), 1-5.

[7] Heilman, M. (2011). Automatic factual question generation from text. *Language Technologies Institute School of Computer Science Carnegie Mellon University*, *195*.

[8] Chali, Y., & Baghaee, T. (2018, November). Automatic opinion question generation. In *Proceedings of the 11th International Conference on Natural Language Generation* (pp. 152-158).

[9] Lindberg, D. L. (2013). *Automatic question generation from text for self-directed learning* (Doctoral dissertation, Applied Sciences: School of Computing Science).

[10] Kalady, S., Elikkottil, A., & Das, R. (2010, June). Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation* (Vol. 2). questiongeneration. Org.

[11] https://api.text-mining.ir/swagger/index.html