Technical University of Crete
Electronic and Computer Engineering

"Using crowdsourcing for grammar induction with application to spoken dialogue systems"

by

Elisavet Palogiannidi

thesis committee

Thesis supervisor:	Assistant Professor Polychronis Koutsakis
Committee member:	Associate Professor Alexandros Potamianos
Committee member:	Professor Euripides Petrakis

Chania, Crete
July 2013

Abstract

Spoken Dialogue Systems are becoming even more common in daily life, supporting information access to the masses in a plethora of domains, such as flight information, restaurant guide, and others. At early stages of the development of a Spoken Dialogue System, there is a chicken-and-egg problem, whereby good quality user data cannot be obtained without a reasonably robust system. Thus, many other ways of mining good quality data for such systems, are used during the development stage.

The data are necessary during the development stage, because they can be used in data-driven grammar induction algorithms, in order to introduce new rules to the grammars that are incorporated to the system for understanding what the users need. This is important, because the performance of Spoken Dialogue Systems, is influenced by the variety of grammar rules that constitute the speech understanding grammar.

One novel way for data acquisition is a new method, called Crowdsourcing. According to its definition Crowdsourcing is "the act of taking a job traditionally performed by a designated agent and outsourcing it to an undefined, generally large group of people in the form of an open call". In Crowdsourcing jobs, a large group of people give answers to tasks that are easy for humans but difficult for computers, in exchange for a small amount of money.

In this thesis the Crowdsourcing method is used, for gathering appropriate for the induction of new grammar rules, relevant to travel flight domain data. Using the Crowdsourcing method, we design both the User Interface and the content of the tasks. Then, the Crowdsourcing workers have to complete the tasks. The major problem in this method is that many Crowdsourcing workers prefer to "cheat", with various ways, for saving time and money. Thus, during the Crowdsourcing process, we use various techniques, in order to reach the best possible quality for the data that we collect and reject users who intent to "cheat".

After the data acquisition, numerous metrics are implemented, in an attempt to quantify data quality and find out whether we achieved our goal. Finally, we reached the conclusions that the data collected using Crowdsourcing method, can be used for grammar induction, but the performance achieved is lower compared with the data collected using web harvesting method. We believe that if we succeed in rejecting "cheaters" in time, the performance of grammar induction will be increased. Moreover, we introduce a kind of methodology for designing Natural Language tasks, based on an array of parameters. Further goal is to deal with an automatic sentences generator applied in the design process.

Keywords: Crowdsourcing, Spoken Dialog Systems, Grammar Induction, Crowdsourcing task

Acknowledgments

Foremost, I would like to express my sincere appreciation and gratitude to my supervisor Alexandros Potamianos, for all I learned from him during this year. His continuous support and encouragement in all stages of this thesis, as well as, his suggestions, ideas and comments were invaluable. Also, I would like to thank my committee members Polychronis Koutsakis and Euripides Petrakis for reviewing my work.

Next, I'd like to thank all the people who contributed to the successful completion of this thesis: Giannis Klassinas for his help and guidance, Dr. Elias Iosif for his guidance, and the participants of the pilot evaluation for their time.

This thesis completion, implies the end of an important part of my life and I don't even know if I would be here today if I hadn't by my side some special persons. So, I thank my best friends, who I met in Chania, for sharing with them beautiful and carefree memories, and for being by my side in the difficult moments. My greatest thanks go to the persons that I could talk about my problems and excitement, and share with them my worries and ideas: Konstantinos Maragos, Georgia Athanasopoulou and Christina Ioannou. Special thanks to Konstantinos for helping me think rationally.

Last but not least, I have no words to thank my family, and especially my parents, Eleftherios and Vasiliki, and my sister Anastasia, for believing in me and supporting my choices.

This work is dedicated to my grandparents, Ioannis and Elisavet Palogiannidi, who wanted so much to see me achieving this goal. Also, to my grandparents Ioannis and Anastasia Kontogianni who, unfortunately, can't see me, but I know that they would be full of pride.

"In God we trust. All others must bring data."

W. Edwards Deming

Contents

List of Figures

List of Tables

List of abbreviations

SDS	Spoken Dialogue System
NLP	Natural Language Processing
SU	Speech Understanding
UI	User Interface
HIT	Human Intelligent Task
AMT	Amazon Mechanical Turk
MT	Machine Translation
LM	Language Model
OOD	Out of Domain
HCI	Human Computer Interaction
MHP	Model Human Processor

Chapter 1

Introduction

1.1 Motivation

Interactive Spoken Dialogue Systems, (SDS) are becoming increasingly pervasive in daily life. To be more specific, in the last years they have moved from research prototypes to real-life commercial applications ([21, 34, 36, 47, 59, 60]). However, there are two key issues for improvement. Firstly, the evaluation of a SDS is an open research problem, which can be categorized into component-based perspective and holistic-based perspective ([20,22,54,58]). The first covers the performance of individual components, in contrast to the second that involves the perceived level of system usability. Still, a major roadblock in SDS prototyping, is the difficulty not only on the development, but also on porting a SDS to new domains or languages. This porting is time-consuming and the Speech Understanding (SU) grammars require expertise in NLP and in-domain data as well ([31]). By now, some of the technologies that make SDS's porting easier and have not penetrated through to the commercial systems yet are ontology enrichment, automatic grammar induction and others.

The main issue that arises when deciding to port a SDS to a new domain or language, is the acquisition of in-domain data. There are several methods for gathering data, such as *web data harvesting*, *crowdsourcing*, *transcription*, and each of them has various benefits and drawbacks. Crowdsourcing is a new and widely growing area of research, based on the idea of the wisdom of the crowds and, actually, tasks that would normally be reserved for experts, are provided to a crowd, consisting of non-experts (*usually*) ([25]). Consequently, using Crowdsourcing for data acquisition is a challenge, which becomes greater and more interesting because of the presence of the human element.

Moreover, data-driven approaches to grammar induction rely solely on corpora ([41, 48]). Thus, using grammar induction as the technology that will make SDS's porting easier is an extra challenge, because its performance will mirror the usefulness of the data that were collected.

1.2 Background

The current thesis is most highly related to previous work falling into three main areas: the development of SDSs, the induction of SU grammars, and data acquisition with Crowdsourcing method.

To begin with, SDS development, in the last decades proved to be growing by leaps and bounds. It is worth to study the "philosophy", the system architecture and the different approaches to the development of SDSs as well. This study offers a better understanding of the main "object" of the present research, posing, at the same time, a strong basis thinking of new ideas in an effort to evolve or improve the existing.

Still, Crowdsourcing, a new method, used for data collection with the use of various micro tasks such as image annotation, transcription, translations and others. Crowdsourcing background work review, provides, not only a better understanding of the method, but a clear image of what is done, as well.

Finally, Grammar Induction, is the part of SDS development that this thesis focuses on. It is useful to research the various grammar induction algorithms and the corpora that were used for this purpose.

On the whole, our background work research, poses a basis of comparison between this thesis's work and previous works. This research is described with more details in chapter 3.

1.3 The task

The main topic of this thesis is the employment of Crowdsourcing for the collection of data for SSDs, that provide travel information, and especially information about flights. More specifically, the aim of this thesis is to collect data in order to use them for inducing new grammar rules. Therefore, the core of this work, is the design of the Crowdsourcing tasks, in such a way that Crowdsourcing workers will be motivated to provide useful responses, related to travel domain. The design of Crowdsourcing tasks contains the generation of a plethora of questions that are part of the Crowdsourcing tasks, and the design of the User Interface (UI). The first is a more theoretical part, while the second part is technical, including the implementation of the UI and the upload of the questions. In addition, when the collection procedure is over, a number of corpora, subset of the corpus that was collected, are going to be created, in an attempt to select the corpus for which the best grammar induction performance is achieved. Apart from grammar induction, various metrics are going to be used, providing an image about the effectiveness of Crowdsourcing tasks's design.

1.4 Contribution

Within the present work, we illustrate that Crowdsourcing can be used for collecting appropriate data for the induction of new grammar rules. It is sufficient to have an initial grammar, and focus on specific parts for collecting in-domain data. This provides a solution to the chicken-and-egg problem that appears during the development or porting phase.

Additionally, we introduce a kind of methodology for designing Crowdsourcing tasks about SDSs, defining the parameters that influence the quality of the collected data. This is important, because we reach conclusions about the characteristics that a Crowdsourcing task should have for achieving an efficient outcome.

Moreover, due to the nature of the tasks, it was impossible to use the quality control mechanism of the crowdsourcing platform. Thus, we explain the way that we took advantage of all the available options of the platform, trying to collect as good data as possible.

1.5 Thesis outline

The next chapters of this thesis address the following issues:

Chapter 2 describes in brief the necessary theoretical background that reader needs to know.

Chapter 3 presents the previous work that has been done on SDS porting and development, Crowdsourcing tasks and Grammar Induction.

Chapter 4 describes the first approach to the task that we had to complete. The most important steps are the definition of the problem with its particularities and the brainstorming phase, which includes the basic design of Crowdsoucing tasks. Also, we describe the pilot process, that took place, mostly, for getting feedback about the design.

In chapter 5, the procedure of the final design of the tasks is described.

Chapter 6, refers to the data collection procedure and the data analysis.

Finally, chapter 7 summarizes all the available information and final conclusions are extracted. Besides, future work is the ending part of the thesis.

Chapter 2

Theoretical Background

In this chapter, we present the theoretical background that reader needs to know in order to better understand the practical aspects of this thesis. Thus, we divide this chapter in three sections: Human Computer Interaction (HCI), language modeling and grammar analysis.

2.1 Human Computer Interaction

HCI is a research area, emerged in the later 1970s, with the emergence of personal computing, and initially, was a specialty area relevant to cognitive science and human factors engineering. Personal computing, made every human a potential computer user, and highlighted deficiencies relevant to usability.([10], [16])

During the interaction of a human and a computer, the human input is the output of the computer and vice versa ([16]). A model that describes the human's side in this interaction is described in [9], and it is introduced with the name Model Human Processor (MHP). In essence, MHP is used for representing the different stages of the processing that a human makes during the interaction with a system. Humans use the perceptual processor for perceiving the various stimuli from the computer, the cognitive processor for more complex tasks, e.g. for solving a problem, and the motor processor, which is the final part of the interaction from the humans' side and it is responsible for the movement response (*arms, mouth e.t.c.*), for providing the input to the computer. The MHP model, from [9], is depicted in figure 2.1

In [8] the various performance metrics and user kinds of a user - computer interaction system are described. The performance of the user - computer system is based on the following quantities : time (*that takes a user to complete a task using the system*), the errors (*during the interaction and how serious are they*), learning (*the time that takes a user to learn how to use the system*), functionality (*the range of the tasks that a user can accomplish within the system*), recall(*how easy is to recall how to use a system that the user has used in the past*), concentration (*how many things has the user to keep in mind during the interaction with the system*), fatigue (*after interacting with the system for an extended period*), and acceptability (*that comes from users' subjective evaluation of the system*). Regarding the users, it is mentioned that they have different experiences and knowledge of other tasks and systems, motor skills (*such as typing speed*) and technical abilities from each other.

Additionally it is mentioned that a user, in order to manipulate large tasks, breaks

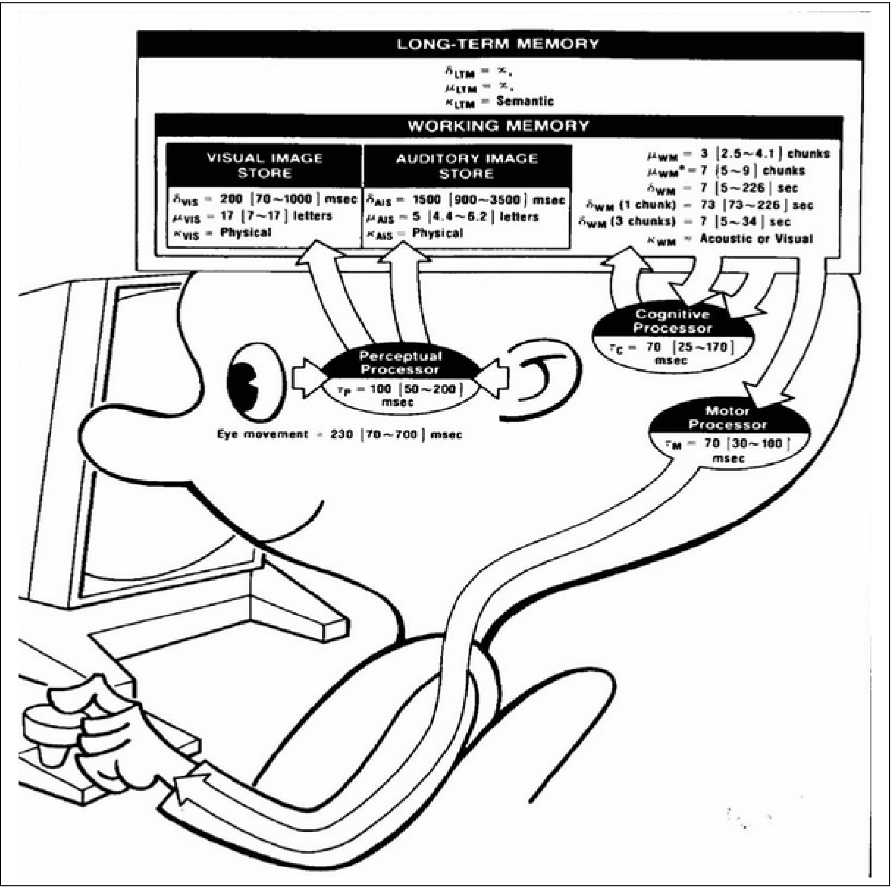

LONG-TERM MEMORY
$\delta_{LTM} = \infty,$
$\mu_{LTM} = \infty,$
κ_{LTM} = Semantic
WORKING MEMORY
VISUAL IMAGE STORE
δ_{VIS} = 200 [70~1000] msec
μ_{VIS} = 17 [7~17] letters
κ_{VIS} = Physical
AUDITORY IMAGE STORE
δ_{AIS} = 1500 [900~3500] msec
μ_{AIS} = 5 [4.4~6.2] letters
κ_{AIS} = Physical
μ_{WM} = 3 [2.5~4.1] chunks
μ_{WM}^* = 7 [5~9] chunks
δ_{WM} = 7 [5~226] sec
δ_{WM} (1 chunk) = 73 [73~226] sec
δ_{WM} (3 chunks) = 7 [5~34] sec
κ_{WM} = Acoustic or Visual
Cognitive Processor
τ_C = 70 [25~170] msec
Perceptual Processor
τ_P = 100 [50~200] msec
Eye movement = 230 [70~700] msec
Motor Processor
τ_M = 70 [30~100] msec

Figure 2.1: Model Human Processor

them into small tasks that are cognitively manageable, called unit tasks. (*Note that Crowdsourcing is based on this idea*). Unit tasks, consist of two basic parts: acquisition, in which the user builds a mental model and execution, in which the user utilizes the facilities of the system for completing the task. The total time a user spends to complete a task, is the sum of the time that he dedicates to each of the above parts. According to the difficulty of the task the acquisition time may be higher, beginning from 2 to 3 seconds. The execution time of a unit should be around 20 seconds, but if it is higher, the user, likely, breaks it into smaller units.

It is worth to say that the designer of the system, also builds a mental model and if the mental model of the user meets the mental model of the designer, then there is a successful interaction, which means success of the task design.

In the Crowdsourcing system that we are going to create, we design tasks for users, whose majority wants to complete them fast, so we need high performance in time and learning, while performance in functionality, recall and fatigue is not important for us, since for each task we have an independent system and it is available for a small time period. Performance in concentration is also important, because we want to create simple and not complex tasks for the users. Acceptability was the basic performance metric in pilot process, which is described in chapter 4. Regarding the users that we address, the majority of them has Crowdsourcing experience and satisfactory motor abilities.

The design that we create is based on the principles described in this section.

2.2 Language modeling

A statistical model of word sequences is known as language model. A LM can be used for giving the probability P(s) of a sentence s. The majority of the language models decomposes the sentence probability, P(s), into a product of conditional probabilities. In the simplest possible model, any word can follow any other word and this makes all the words to have equal probability to follow any other word. In a more complex LM we take into account the frequency, with which a word appears. However, for specific sequences, words with lower frequency may be more reasonable than words with higher frequency and this observation considers the conditional probability of words given the previous words instead of the relative frequency.

The N-gram language model considers the language as a Markov process of order $N-1$.

$$P(w_i|h_i) = P(w_i \mid w_{i-N+1}, \dots, w_{i-1}) \approx P(w_i \mid w_{i-N+1}^{i-1}) \tag{2.1}$$

Equation 2.1 states that the probability of word w_i given all the previous words of the sentence can be approximated by the probability given only the previous $N-1$ words.

In bigram LM, the probability of a word given the previous is approximated by the probability given the preceding word. Trigram is the same as bigram instead of that the condition goes on two previous words, and so forth as the N increases.

The major problem of LM is that they must be trained from a corpus, and since there aren't infinite corpora, some N-grams is possible to miss from the LM, and this means that they have zero probability. Smoothing is one technique that assigns a non zero probability to a zero probability, for avoiding calculating problems. Backoff is an other technique that is used, in which, if we meet a zero probability for an N-gram, we "back-off" to lower N-grams.

For understanding better what we described, consider the sentences "I want to travel all over the world", "I like traveling with airplanes". The total number of these words is 13 and according to the simplest model, the probability of each word is $\frac{1}{13}$. But taking into account the relative frequency of each word, the probability of each word is different, according to the number of appearance, e.g. $p(I) = \frac{2}{13}, p(like) = p(world) = \frac{1}{13}$.

The probabilities of the sentences according to the bigram and the trigram, is nothing else than the multiplication of the appropriate probabilities. We show one example for bigram and one for trigram with each of the sentences, beginning with the first sentence and the bigram.

· p(I want to travel all over the world) $= p(I| <s>$ [1]$) \cdot p(want|I) \cdot p(to|want) \cdot p(travel|to) \cdot p(all|travel) \cdot p(over|all) \cdot p(the|over) \cdot p(world|the)$

· p(I like traveling with airplanes) $= p(I| <s_1><s_2>) \cdot p(like| <s_2> I) \cdot p(traveling|$I like$) \cdot p(with|$ like traveling$) \cdot p(airplanes|$traveling with$)$

Perplexity

Perplexity and entropy are the most common metrics used for evaluating N-gram systems. Entropy is a measure of information and it can be computed after establishing

[1] means the start of new sentence

a random variable X that ranges over what we are predicting which has a probability function $p(x)$. Entropy of a random variable X is shown in equation 2.2.

$$H(X) = -\sum_{x \in \chi} p(x) log_2 p(x) \tag{2.2}$$

Intuitively, we can think of entropy as the lower bound of the bits that we need for encoding a piece of information. Moreover perplexity, which is defined in equation 2.3, can be thought as the weighted average number of choices a random variable has to make ([37]) and is a metric for expressing how well a statistical model matches to a test corpus. The higher the N the lower the perplexity and this is logical because, the language model has less possible choices, since it takes into account higher tuples of words.

$$perplexity := 2^{H(X)} \tag{2.3}$$

For calculating perplexity in practice, we need a LM and a test corpus and we can use the SRILM toolkit [51] . It is important to build the LM without any knowledge of the test corpus, because this will lead to very low perplexity.

The perplexity is calculated from SRILM toolkit by the following formula:

$$ppl = 10^{(\frac{-logprob}{words - OOVs + sentences})} \tag{2.4}$$

where

1. sentences is the total number of sentences (*in perplexity per sentence calculation is 1*)
2. logprob gives the total logprob, ignoring OOV word tokens.
3. words is the total number of words
4. OOV is the number of unknown word tokens, i.e. tokens that appear in test corpus but not in train corpus from which the language model was generated.
5. ppl is the geometric average of 1/probability of each token, i.e., perplexity.

2.3 Context - free Grammars

In 1956 Chomsky introduced in [12] the hierarchy of the languages which are depicted in figure 2.2.

Context-free grammars belong to the realm of formal languages - they sprang out from linguistics as a way of understanding the syntactic regularities of natural languages. In equation 2.5 we view the definition of a context free grammar with its basic components.

$$G = (V, \Sigma, R, s) \tag{2.5}$$

where

1. V is the non-terminal vocabulary and each of its elements represents a different type of phrase or clause in the sentence.
2. Σ is the set of terminals, from which the actual content of the sentence consists of.

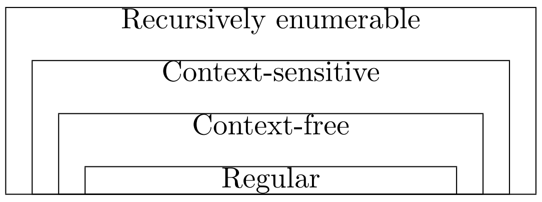


Figure 2.2: Venn diagram of languages in Chomsky Hierarchy

3. R is the set of rules of the grammar and, in essence, is a finite relation from V to $(V \cup \Sigma)^*$.

4. S is the start symbol.

We can see an example from [37] of a context-free grammar in figure 2.3 for the context-free language $L : a^n b^n$

S → aSb
S → e

$$\begin{tikzcd}
 & & S \arrow[d] \arrow[dddll] \arrow[dddrr] & & \\
 & & S \arrow[d] \arrow[ddl] \arrow[ddr] & & \\
 & & S \arrow[d] & & \\
a & a & e & b & b
\end{tikzcd}$$

Figure 2.3: Rules of the context-free grammar for the language L and context-free parse tree for "aabb"

ABNF grammar format

Augmented Backus–Naur Form (ABNF) is a format used for representing context-free grammars, based on BNF, and consists of its own syntax and derivation rules. In ABNF format a rule is a set of characters that can be letters, numbers or hyphens enclosed in brackets. Some of the useful operations that are defined between the rules are concatenation, for a rule that is defined from a sequence of rules, alternative for a rule that is defined from a sequence of alternative rules and incremental alternatives for adding to a rule more alternatives. Also, numeric values can be used in a rule, enclosed in "‚ ". We can see an example of a grammar in ABNF format, used in real applications in figure 2.4, where the brackets [ ] mean that the rule that they enclose can used optionally.

```
<TIME>= <HOUR>[<MINUTE>/<O'CLOCK>] [<ampm>]
<TIME>=/ <HOUR><MINUTE><O'CLOCK>[<ampm>]
<TIME>=/ <HOUR>"HUNDRED" ["HOUR"/"HOURS"]
```

Figure 2.4: example of a part of an ABNF grammar

2.4 Natural Language Parsing

A parser is a program used as a first step on working out the meaning of an input text and, actually, reveals the grammatical structure of the text. There are two methods for parsing, bottom-up, in which the text's lowest-level details are identified first and the highest level is identified last, and top-down, that uses the opposite approach. The parser that we are going to use for the data analysis is bottom-up and it is better to understand its functionality though an example.

Suppose that we have a parser with an incorporated grammar, and test text file that we want to find its parse tree. In figure 2.5 we show the grammar that the parser uses and the text which we want to find its grammatical structure and in figure 2.6 the parse trees that are created for each sentence.

```
                    GRAMMAR
<CITY-DAY>→ <city>"on"<day >
<city>→ ATHENS / CHANIA
<day >→ MONDAY / TUESDAY / WEDNESDAY

                TEXT FOR PARSING
Sentence 1: I go to Athens
Sentence 2: I go to Athens on Monday
Sentence 3: I leave at eight o'clock
```

Figure 2.5: Hypothetical grammar and test text for a parser

In figure 2.6, we depict the parse tree for beginning from the sentence 1 and ending in sentence 3. Note that sentence 3 does not have a parse tree, because there isn't a grammar concept for assigning any word of the sentence. We consider this case as a failure, while the other sentences were parsed with success.

2.5 Summary

In this chapter we introduced, in brief, some special issues from the theoretical background that the reader needs to know for the remainder of the thesis. HCI is necessary for the part of the design and the interaction of the humans with the system that we are going

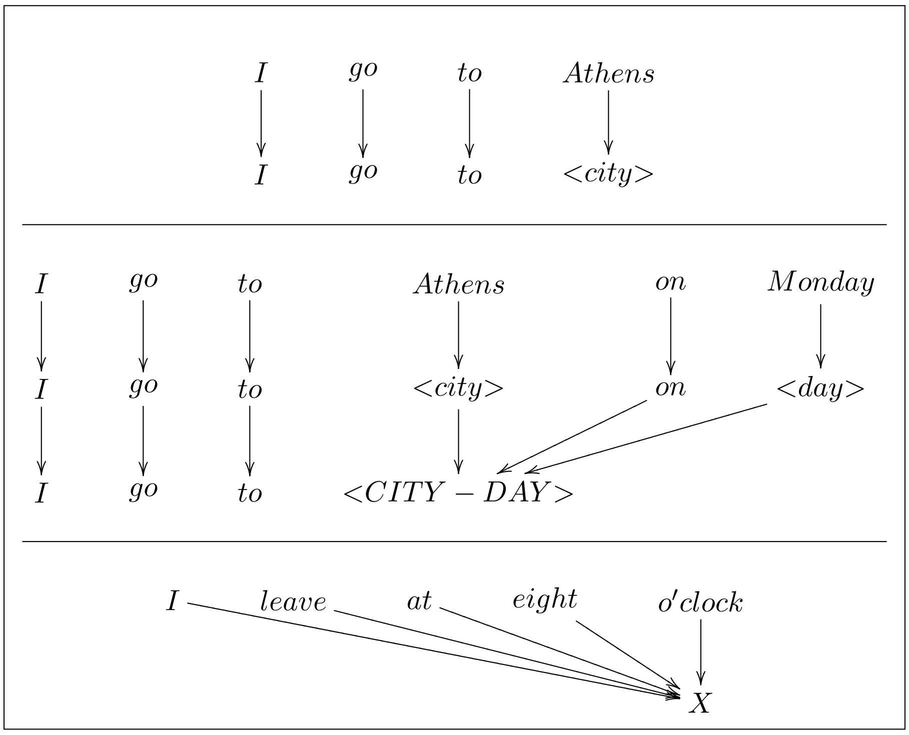


Figure 2.6: Parser trees for the test sentences based on the hypothetical parser

to build, while LM and NT parsing are necessary for the analysis phase, after collecting the data.

Chapter 3

Previous Work

3.1 Spoken Dialogue Systems

3.1.1 Introduction

Looking back in 1950, Alan Turing in [53] considered the question "*Can machines think?*". However, due to the ambiguity of the terms "machine" and "think", for the computer, it was not possible to give an answer, thus he introduced an empirical test, played from two people and one machine, in which language is used for determining whether a machine can think. If the machine could win the game, which means that it could fool the people and make them believe that is human, then it would be characterized as intelligent and this is known as the famous Turing test. According to Turing, it is sufficient to make machines use the language as the humans do, as an operational test for intelligence.

Nowadays, SDS, are being developed with rapid rhythms. We can consider the development of a SDS as an attempt to grant "intelligence" to a machine. An intelligent machine, should act in a dialogue like a human would do, viz., to understand language, use language and interact with a human in a dialogue.

To begin with, as it is expressed in [37] a dialogue is a conversation between at least two participants, taking the roles of hearer and speaker, who alternate periodically, and consists of communication types such as asking questions or giving answers. A dialogue differs from a common monologue, due to the turn-taking of the dialogue, and the critical point is for each participant to know when is the proper time to contribute his turn. Besides, dialogue is a collective act between hearer and speaker, with the need of establishing common ground, and acknowledge the other user's utterances. The final big difference, is that, in many cases the participant does not answer clearly to an utterance, but the semantics of this statement are produced from a semantic interpreter.

A dialogue between humans is an easy issue, in contrast to a dialogue between a human and a computer. What SDSs try to achieve is to provide to humans natural conversations during the interaction with the systems.

3.1.2 Characteristics of a SDS

Spoken Dialogue Systems are automatic systems, derove on the grammar induction module to take advantage of bootstrap grammars if available (in addition to web data), and 3) combine parsing with statistical semantic relatedness metrics to aling, usually, with the dialogue of a user and a machine. The majority of SDS is telephone-based and enable users to conduct transactions or ask information about the kind of application that SDS is about. The different purposes that a SDS can be used for and the different languages, make SDS a constantly growing area of research. Figure 3.1 (*from [28]*) depicts the basic modules that constitute a SDS and the interconnection with each other. As we observe, SDSs are the integration of various aspects from the fields of speech recognition and synthesis, NLP, information retrieval and dialog modeling.

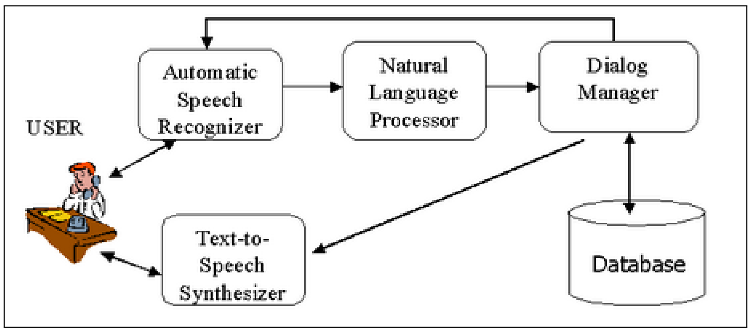

USER
Automatic Speech Recognizer
Natural Language Processor
Dialog Manager
Text-to-Speech Synthesizer
Database

Figure 3.1: Diagram of SDS's basic modules

Automatic Speech Recognizer (ASR)

The role of ASR as a module of SDSs is to translate spoken words, that take as input from the users, into text. Speech recognition is a very complex problem, since different pronunciation, accent, articulation, roughness, nasality, pitch, volume, and speed can can lead to low performance outcomes easily. Likewise, background noise can incommode speech recognition process. These are some of the reasons that lead to various types of recognition errors.

Natural Language Processor (NLP)

Input of NLP is the outcome of ASR, and its basic process is to extract the meaning/keywords of the user through the semantic and syntactic analysis, taking domain information into account. It is a very sensitive component, since different wordings of saying the same thing to the user will yield very different results.

Dialog Manager (DM)

DM has been assigned to establish the connection between the parts of the dialogue system. In other words, it controls the data flow, taking as input the NLP's outcome, getting the corresponding user information from the database and sending the dialogue response to to TTS. In figure 3.2 a finite-state automaton architecture for a DM is depicted.

Text to Speech Synthesizer (TTS)

During this process artificial human speech is produced. In this case, the response that DM has decided to give as answer to user's input is converted into acoustical signal and played back to humans.

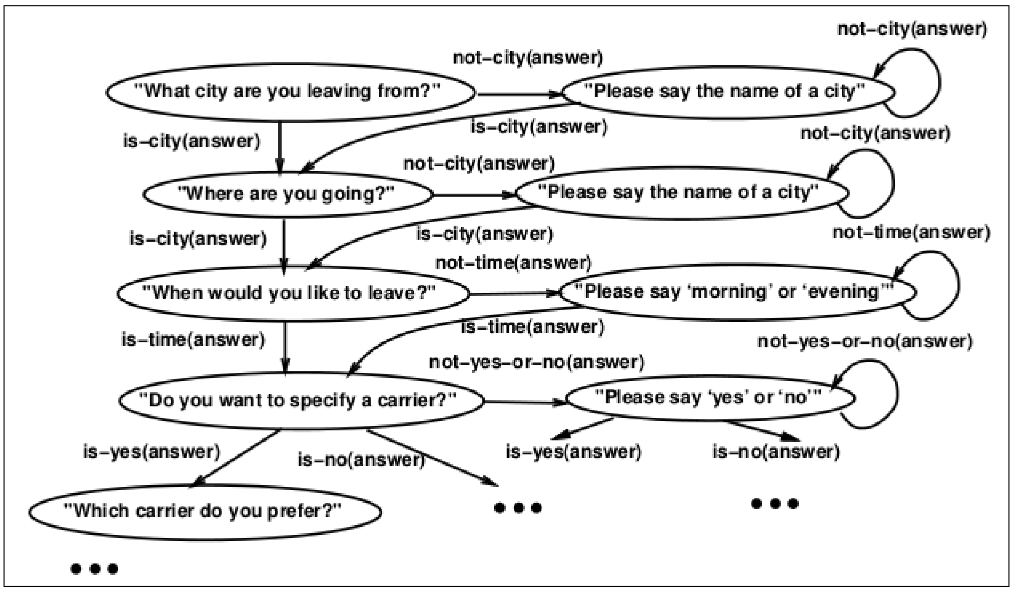

"What city are you leaving from?"
"Please say the name of a city"
"Where are you going?"
"Please say the name of a city"
"When would you like to leave?"
"Please say 'morning' or 'evening'"
"Do you want to specify a carrier?"
"Please say 'yes' or 'no'"
"Which carrier do you prefer?"
not-city(answer)
is-city(answer)
not-time(answer)
is-time(answer)
not-yes-or-no(answer)
is-yes(answer)
is-no(answer)

Figure 3.2: Example of a finite-state automaton, from [37], that a DM can use

3.1.3 Previous works on Spoken Dialogue Systems

In [33] a stochastic model for SDS using Markov Decision processes is introduced. A strategy is defined as good if, given a current state of the system, the next action can be invoked. The main assumption that this research is based on is that a good strategy minimizes the costs of all the important dialogue dimensions. [47] describes the use of reinforcement learning for the automatic design of dialogue strategies between human and computer, using an error simulation tool and a detailed user. The PARADISE general framework for evaluating SDS is presented in [54]. It is used for evaluating dialog strategies and allows comparisons among different tasks, agents that perform different tasks and performance calculation over sub-dialogues and whole dialogues as well, as a function of tasks success and dialogue cost. The research in [44] exposes a probabilistic framework used for realistic simulation of SDS, whose components are modeled with respect to independent data or expert knowledge aiming to evaluate the dialogues and learning optimal strategies. Also, an automatic speech recognition model and a user model were developed while the experiments were done for two different tasks and indicated potentially problematic scenarios.

Except from the works that aim for the best strategies, many researches have been done for the development of SDS. SpeechBuilder is a utility described in [21] and its purpose is to facilitate the development process of a SDS. More specifically it is used for creating domains by accessing structured information from a database, and create spoken language interfaces. Also, [49] represents the CSLU tooklit, which provides an interface for spoken language technology, and supports all stages of development, from prototyping to evaluation.

Jupiter is described in [59], and it is a conversational interface for weather information, based on telephone interaction. It can provide weather information for a large number of countries all over the world and sources for finding information. An attempt for content processing and information selection is described as well. Another SDS, named August (*the* *name of an animated talking agent*) is used in [23] for the collection of spontaneous speech data, with one of its aims to try to semi-automate the extension of the system according to the user's needs. However the main aim of this system was to study the interaction of naïve users with spoken language technologies with several domain. Thus the specialty of this system is that is uses numerous domains instead of one and more complex as the majority of SDS does. In addition, the SDS described in [34], ITSPOKE, has tutoring purposes. A student interacts via natural language text with the system, and its basic purpose is to create an empirical understanding of the branches of adding spoken language capabilities to dialogue tutors based on text. PEGASUS, a spoken language interface that provides the users with the capability of planning travels on line, is described in [60]. In [36] a SDS that developed for dealing with fast-food orders is described, and consists of a dialogue system and a voice interface.

The problem of evaluating SDS, is studied in [58] from a different point of view. More specifically, this study proposes and presents a novel evaluation framework with Finite State Machines (FSMs), that can evaluate SDS in different domains, in which FSM states can be regarded as efficiency measurements. Data were collected from the"Let's Go!" dialog system, keeping dialogs with more than 6 turns and classifying them into three categories. The state in an FSM captures the key information contained in each turn and each dialog can be modeled as a state transition path in the FSM. The key idea about an FSM is that a dialog with a long path and few state transitions reveals low efficiency. The evaluation of SDSs is based on regression models built on the efficiency measurements. The results showed that SDS evaluation with FSMs can be utilized for enhancing dialog system's evaluation methods.

[20] describes data collection and performance evaluation infrastructure in support of spoken dialogue system development. The data were collected from a toll-free phone line from a flight information system[1], which allows users to plan their travels all over the world. Finally, about eleven thousand utterances were collected, and a subset of them was used during the evaluation process, which is based on two understanding metrics, one metric defined per utterance and one defined per dialog. According to the investigation that was done, one outcome was that making the system available to real users is a crucial aspect of system development.

In [43], the main differences of commercial and research SDS development are mentioned. The goals of each are different and the usability needs as well.

Up to now we have presented only some of the existing works in literature, since the relevant list wof research contributions on SDS is too large to be presented.

3.1.4 Comparison with this thesis work

In the last decades, SDSs's development has become an increasingly growing area of research and a part of daily life. Corollary of this, is the development of many SDS that lean on a plethora of daily occupations. The previous works that were represented, were about the SDSs' evaluation and data collection process. The current work, like some of the previous and in contrast to some others, aims to collect data, for the purpose of developing a SDS. Likewise with the majority of the previous works, the SDS belongs to a specific domain, and especially the travel - flight domain.

The main difference, between the previous works that used crowdsourcing for data

[1] called MERCURY

collection for SDSs, and the present thesis, is that in the current work we don't use dialogs that come from SDS. In fact, we get inspiration from utterances collected from SDSs for creating potential scenarios of interaction during the design of the tasks.

3.2 Crowdsourcing

3.2.1 Introduction

The term Crowdsourcing was introduced for the first time in 2006, by Jeff Howe and it is "the act of taking a Job traditionally performed by a designated agent and outsourcing it to an undefined, generally large group of people in the form of an open call". It is based on the "*Wisdom of the Crowds*" idea and in each Crowdsourcing task there are two distinguishable roles. People who design and people who perform the tasks, that are easy for humans but difficult for computers. Quid pro quo for the people who perform tasks is a small amount of money.

The rapid development of Crowdsourcing led to the creation of a plethora of crowdsourcing platforms, some of which are specialized on certain tasks, while some others allow the designers to create any desirable kind of task. The platforms that belong in the second case are of greater interest, since they can be used for a large variety of tasks. Actually, these platforms act as the mediator between the designers and the crowd.

One major drawback of crowdsourcing, is that some people try to cheat with various methods, in an attempt to save time and earn money. This action leads to the collection of useless data. Thus, an integral part of crowdsourcing is the assurance of the best possible quality of the collected data. Methods that tried to achieve this led to sense of Quality Control.

The historical hoax

One of the most famous and reliable platforms is the Amazon Mechanical Turk (AMT) platform (*launched on 2005*), the name of which refers to a historical hoax from the 18th century. The hoax is about a chess-playing machine, invented by the Hungarian inventor Wolfgang von Kempelen, that appeared to be able to beat human opponents. In fact, the machine was fake, since it was controlled by a puny Turk, hiding inside it. Likewise, the AMT web service allows humans to help the machines of today perform tasks, for which they are not suited.

In [30] two experiments were conducted to test the utility of AMT as a user study platform. In the first, users had to rate articles from Wikipedia, according to a set of factors such as how well written, structured, e.t.c., they were and additionally they had to describe what improvements were necessary for the article's improvement according to their opinion. The second experiment, was the same with the first, including an extra procedure before rating, in which users had to complete four verifiable, quantitative answers. The results showed that the correlation between AMT user ratings and Wikipedia admin ratings was marginally significant, providing only weak support for the utility of AMT rating mirroring expert ratings. On the other hand, results from experiment 2, showed that the correlation between AMT and expert ratings was higher, and the fewer responses appeared invalid. The conclusion of this research is that the design of crowdsourcing tasks must be done with special care, and some design ideas are recommended as well.

3.2.2 Crowdflower

Crowdflower is a crowdsourcing service founded in 2009 that has over 50 labor channel partners, one of which is AMT. Since officially only people or institutions located in the US can post work on MTurk, Crowdflower can be used as an interface to AMT workers. Crowdflower offers a number of enterprise solutions, using the largest virtual workforce, consisting from people all over the world, such as categorization, where crowd matches items to their corresponding category, or transcription where images of handwritten text must be converted into digital forms.

Besides, Crowdflower's Builder solution, is a general purpose crowdsourcing platform, where individual tasks can be created, according to the needs of the clients. Clients have to design the tasks's UI, upload data, create instructions, order judgments and collect data. In table 3.1 we list some of the Crowdflower's terms, because we are going to use them in the following parts of the thesis.

Field	Description
Unit	The smallest autonomous module of a crowdsourcing task
Judgment	A single record that a contributor submits
Page	The number of units that a contributor must complete before submitting
Job	The total units that are uploaded
Contributor	One person from the crowd that participates in crowdsourcing
Requester	The designer of crowdsourcing Jobs

Table 3.1: Public information about crowdsourcing on Crowdflower

Crowdflower has been used for the development of many crowdsourcing Jobs. In one of them, it was used for the design of HITs for quality control [7], by the mechanism "Gold Standard". The idea is that in one phase of the design, the Requester can input gold standard data, that include, except the others, all the possible right answers to a unit that a contributor could provide. Then, units that come from gold standard are mingled with the rest of the units and the contributors who fail in a number of gold standard units can be blocked. In brief, contributors come from a training phase when a score is generated according to their judgments, for giving them feedback on how well they are doing.

In [5] Crowdflower is used for creating preference tests for detecting cheating. In this work, except from *Gold standard data*, *additional information* and *intrinsic metrics* are used for assuring quality control. Intrinsic metrics, use information about which worker answered what to which item. In additional information, pages that were submitted too quickly to have been listened to, are rejected (*listen to speech tests*). Moreover, IP addresses can be used as additional information for cheater exclusion.

3.2.3 Quality control

Since gathering good quality data is the major issue for crowdsourcing, a number of previous works investigate, the way good quality can be assured and how cheaters can be detected.

Apart from [7] and [5], [19] is one of these works, in which speech transcription is used for the development of acoustic models for under-resourced languages, and proves that it is possible to acquire quality transcriptions from the crowd. The languages that were used were African and specifically *Amharic* and *Swahili.* The corpora creation procedure, includes text extraction from news websites and its segmentation by sentences. Then recording was made by native speakers, reading sentence by sentence. For ensuring quality control, descriptions and instructions were given in respective languages, while the sentences read by native speakers were used as gold standard to compare with transcriptions obtained from AMT.

Also, a two-stage transcription task with automatic quality control in each stage was proposed in [32]. Goal of this work is to find an efficient approach for generating high-quality transcripts for long audio recording. In crowdsourcing HITs, workers had to listen to short audio clips and transcribe them (*first phase*). Many transcripts had poor quality, and in an attempt to reduce their number, an automatic classifier that could distinguish between them and good quality transcripts was designed. A benefit of this procedure is that whenever a worker attempts to submit a poor quality transcript, he is warned and asked to improve it. In the second phase, the short audio clips were merged to form longer audio segments, and workers were asked to correct transcript errors. A submitted HIT from this phase will be accepted if the number of changes is more than 80% of the estimated number of errors for the HIT. This estimation is, also, used to provide feedback to workers.

3.2.4 NLP tasks

Many of the crowdsourcing tasks belong to NLP area, including knowledge extraction, textual entailment, word sense disambiguation and others. These tasks, can be used during the building of MT corpora, a procedure that becomes increasingly desired for more and more language pairs and domains.

In the study represented in [55], various performance measures of crowdsourcing process are discussed and semantic correctness, naturalness and biases of the collected data are analyzed, focusing on the problem of collecting natural language expressions that correspond to a given semantic form. The crowdsourcing tasks present three alternative elicitation methods: sentences, scenarios, list-based descriptions, in which workers have to rephrase in their own words.

Machine Translation

In [18], AMT is used for the collection of bilingual(*Chinese - English*) word alignment data to assist automatic word alignment. The AMT workers are given a sentence pair and they have to link words in source sentence in one, or more, or the empty word in target sentence. For quality control, answers are filtered out based on the consensus. Also two pricing strategies were tried, but even after the price raised, the number of workers wasn't high enough, event that indicates the limited workers base of Chinese speakers.

Moreover, [1] explores the effectiveness of AMT for the creation of parallel corpora. The corpora are created with 100 sentences and three translations per sentence for all the language pairs between English, Urdu, Spanish and Telugu. This demonstrates the feasibility of cheap corpora for high and low resource languages. From the received datasets, two basic problems were apparent. Firstly, problems that are easy to identify like blank annotations, copy-paste and misspellings and secondly, problems hard to identify, in which AMT workers who don't understand the task but attempt it anyway. Computation of majority consensus translation using fuzzy matching, is agreed to be an effective solution to detect low quality translations.

Apart from the above work, [3] explores the potential of AMT in the creation of MT parallel corpora, too. This work succeeds to buck the trend of diminishing returns and improving translation quality, keeping, at the same time, annotation costs low. The method used for collecting translations, includes only asking n-grams and not entire sentences. One reason that this happens is because translation model learned from the so-far labeled data will be able to translate correctly the rest words in the sentence.

Work [4] proves that it is feasible to use AMT for reducing MT test sets cost. A previous collected Urdu - English MT test set was replicated, paying $0.10 a sentence (*l*ower price than the typical annotator cost). Handling of undesirable behaviors, includes the manually blocking of the AMT users, while simple mistakes such as misspellings were uploaded for correction from the AMT users, with a total cost of $44.80.

Goal of [27] was the use AMT for annotating translation lexicons between English and a large set of less commonly used languages. First, small existing dictionaries were used to induce additional lexical translation pairs, and the AMT users were payed for checking and correcting the system's output. Then, the same procedure continues with the updated lexicons and so on. In the AMT tasks, for each English word within a HIT, ten candidate translations in the foreign languages were posted and AMT workers were asked to check the boxes beside any and all of the words that were translation of the English word. For ensuring the quality of the answers, positive and negative controls were used, if the seed dictionary included an entry for the English word, or with a random word in the foreign language, respectively.

Furthermore, [24] investigates whether undiscovered nicknames could be successfully collected with AMT to added to existing Name Entity Lexicons. First nicknames are gathered from AMT, by asking users to enter an Arabic nickname that they have heard, where they heard it and their country of residence, using a variety of payments. Then they are validated, with the help of five AMT users, three experts and a Google check. Finally, the verified names are compared against the available list of names in the database of Arabic names, to determine if they represent new additions to the lexicon. The goal of this work was achieved, and extra conclusions were that increasing pay improves collection speed and using bilingual directions and requiring typing in Arabic, participation from Arabic speaking countries was able to be increased.

Also, [57] explored how AMT could be used to improve a MT grammar. The collecting procedure relies on the comparison of candidate translation pairs, that are generated based on a number of features. This approach was applied to an Urdu - to - English translation task, where 12 features were used for characterizing each grammar rule. AMT users were provided examples and their feedback was used to re-score grammar productions.

Paraphrasing

The following works use a very common NLP task, named "*Paraphrasing*", which is nothing else, than the recast of a phrase or a sentence, using different words but maintaining the initial meaning.

In [15], a semi-automatic paraphrasing technique for creating additional reference translations (*English-Arabic*) is presented. Paraphrases are automatically extracted from a large parallel corpus. AMT users are shown an original phrase and its paraphrase and are asked to answer with "yes" or "no", if the two phrases have the same meaning or not, respectively. Then, they are shown the original sentence and its paraphrase, and they are asked the same question. The key idea is that the users who assign "yes" to the sentence pair should always assign "yes" to the phrase pair. After the end of data collection, a revised version of the task was designed, containing positive and negative controls. Result of this research was 728 paraphrases for Arabic - English translation, that can also be used for other tasks in MT and NLP.

One other crowdsourcing task that uses paraphrasing is [6], in which paraphrases are used to simplify, potentially, segments of text that are difficult to translate. Paraphrases are being obtained only for the parts that seem to be problematic for the translation system. During the paraphrases obtaining phase, if it was impossible for AMT workers to think of a paraphrase, they had the opportunity to mark an "Unable to paraphrase" checkbox. Totally, from the 1780 errors, 4821 responses contained actual paraphrase data, out of the 5340 that were collected. Next is the verification phase, in which the workers are given an original sentence and they are asked to compare it with five alternatives. The results show that good paraphrases of the problematic spans of translation could improve translation performance.

Speech related tasks

One other widespread area of crowdsourcing tasks, is speech area. Famous speech related tasks are speech acquisition, transcription, annotation as well as the assessment of speech technology.

In [38], a photo annotation HIT was created, and AMT workers were asked to record a spoken description of photos. This, in conjunction to a transcription task is used to grow a spoken language interface. WAMI toolkit is used in order to provide all the necessary plumbing to get audio from client side to a recognizer running server side. The transcriptions in this tasks are iterative procedures, continuing with a maximum number of five workers per utterance. In addition, photo annotation and photo search HITs were created and workers must meet some requirements in order to work on a task. This work proved the feasibility of analyzing a dynamic spoken language system deployed on AMT. Growing trigram models were explored and shown that improvements can be achieved without expert guidance.

[2] asks multiple workers to transcribe speech, and use the audio for acoustic model adaptation. More specifically, this work's goal is the development of a speech - to - speech translation system for a doctor - patient interaction scenario, where both are proficient in different languages. The AMT task requests workers to hear a set of the two languages audio files and transcribe whatever they hear, while they had to answer some verification and personal questions, too. The contribution of this work is the evaluation of the effect of incorporating transcription information reliability information on the Word Error Rate (WER) of an automatic speech recognition system with adapted acoustic models.

The study in [13] aimed to compare the outcomes of traditional and web-based perception tests in which listeners identify words in noise. During the web experiment, listeners heard monosyllabic English words in 12 different types of noise. The performance of a selected subset of listeners is being explored, based on subjective and objective criteria as well. Finally, a list with the confusions during the crowdsourcing procedure is represented.

In [17] naïve annotators are asked to provide prosodic annotation of non-native speech, and this attempt proved to be a cost-efficient way for obtaining such data. Workers had to reach a specific performance limit before participating in the two-phase final experiment, with the second phase being the same with the first and taking place for potential performance improvement. An additional outcome of this study is that extra training is able to improve the performance of naïve annotators.

Annotation methods for collecting data and traditionally-collected corpora, for training LM for speech recognizer, are compared in [35], resulting that crowdsourcing users can be used in several aspects of a SDS's development. A crowdsourcing task for obtaining speech data takes place, in which, AMT workers generated potential interaction scenarios of actual spoken interactions.

The research conducted in [22], investigates how far the performance of a of speech recognition and full-text search can be improved by getting recognition errors corrected through the cooperative efforts of many workers. Apart from that it demonstrates how speech recognition can be put to use in situations where a speech corpus is almost impossible to be prepared in advance. Anonymous users are enabled to correct recognition errors, through the PodCastle[2] web service, which provides the full text of speech recognition results, and the audio files.

Other crowdsourcing tasks

Apart from the above described types of crowdsourcing tasks, one can meet crowdsourcing in games, too. One example of Crowdsourcing through games is represented in [11], where a novel data - driven approach to behavior generation for interactive robots based on a data collection method is proposed.

Moreover, crowdsourcing is used for gathering data for the development of SDSs. The reason why many crowdsourcing tasks related to SDS, take place is either the development of a SDS where data are necessary to be collected, or the evaluation phase where utility of SDS is checked. Previous works about crowdsourcing and SDS are described in the following section.

3.3 Spoken Dialogue Systems & Crowdsourcing

The research areas of SDSs and Crowdsourcing can be combined, leading to some interesting results. The relationship between the two areas is that the second is used during the development procedure of the first. Specifically, when the development of a system begins, a classic chicken-and-egg problem appears: training data are necessary for building the system, but collecting in-domain training data needs the system. Thus, crowdsourcing can be used for simulating the system in order to collect in-domain data. Furthermore, data used for utility evaluation of SDSs can be gathered using crowdsourcing.

In [56] crowdsourcing is used for the collection of user judgments on SDSs through AMT. In this study, two types of HITs were designed, the first targets to collect fast ratings from a large number of dialogs, while the second aims at assess the reliability of the AMT users. Additionally, a set of approval rules developed to take care of the quality of rating from AMT. The results showed that crowdsourcing is a suitable and effective

[2]Podcastle is a social annotation web service that supports searching, reading and annotating.

method for collecting such data, however, AMT workers are not the real users of the SDS and this is a main drawback. The input for the evaluation procedure transcribed from ASR of "*Let's Go*" dialog system ([45]) and next, the dialogs were classified into five categories, according to dialog's characteristics. Then, the evaluation of the dialogs is based on some questions and selection of the desired from predefined answers, and for assurance of the quality of ratings, which impacts on the credibility of SDS evaluation, an approval mechanism was developed.

One more study in [29] describes a framework of spoken dialogue systems and compares the obtained results with a trial in which the systems were tested by locally recruited users, in a controlled environment, that did a series of task in one hour. The framework combines a telephone infrastructure used for connecting users with the dialogue systems and a web interface for presenting tasks and collecting users's feedback. The comparison of the results showed that ranking between the two populations was consistent, resulting that the use of Crowdsourcing was more efficient, because less effort and cost was needed in contrast to the controlled test. The drawback, as in the previous described study, is that the crowdsourcing workers are not real users of the system.

3.3.1 Comparison with this thesis work

Crowdsourcing, as it is obvious from previous work, has become an accepted method for gathering data. Crowdsourcing's peculiarity is that its main advantage, which is the great diversity of people groups that participating is, also, its main disadvantage. This happens, because humans have the tendency to cheat, whenever it is possible, especially if anonymity is preserved. However, a number of previous works, aim to find efficient ways of Crowdsourcing design in order to collect useful data.

The innovative element that the current work introduces, is the definition of an array of parameters that are used during the design process, in an attempt to quantify senses such as freedom, politeness, specificity, e.t.c.. If we manage to find a correlation between these metrics and the quality of the workers' responses we will be able to design tasks that will yield higher quality corpora.

3.4 Grammar Induction

3.4.1 Introduction

The term "Grammar" is used for expressing the set of the rules that are used for producing all the admissible compositions of words or phrases that belong to a specific natural language. Admitting that a language has a specific structure, grammar can be viewed as a theory of this structure ([12]).

Speech understanding grammar is a necessary module of a SDS, used for understanding what a human says. In this case, we need grammars that understand a specific natural language such as English, Greek, e.t.c., and their development is difficult because of the huge size of all possible phrases that can be used in these languages. This is one of the reasons that SDS are used for specific domains and purposes. However, talking about natural language understanding of a human's input from a machine, we can never say that we have enough grammar rules. For this reason grammar induction is a process used for introducing new rules for an existing grammar. The grammar induction principle is a machine learning process and an important part of natural language understanding.

Moreover, the number of rules that a grammar contains can be one of the factors that are reliable for the performance of a SDS.

History

Etymologically, the word grammar derives from the greek phrase “γραμματική τέχνη” which means “grammatical art”. The first grammar that was created and is still saved today, is originated in Iron Age India, with Yaska (*6th century BC*), Pãṇini (*4th century BC*), a Sanskrit grammarian, with most famous the ASṭādhyāyī (*“Eight Chapters”*), where the morphology rules of Sanskrit are coded up to 3959.

3.4.2 Previous work in Grammar Induction

The area of research relevant to Grammar Induction counts a plethora of works. Most relevant to this thesis is the work presented in [31] where a data-driven approach for grammar induction is used. The data that they used have been harvested from the Web, and numerous filters are applied to the collected corpus, in an attempt to find the corpora with the best performance. Finally, they reached the conclusion that unsupervised web harvesting can be as good as manually collected corpora.

Another work is presented in [41], in which an unsupervised, iterative procedure for inducing semantic classes was proposed. The suggested system consists of a *lexical phraser*, that identifies frequently co-occurring words by applying a weighted point-wise mutual information measure (*e.g., consecutive words such as “New York” are chunked into the entry “New_York”*), a *semantic generalizer*, which generates rules that map words (*or previously induced classes*) to semantic classes and a corpus parser. The core idea of semantic generalizer is the distributional hypothesis of meaning. The similarity between two words is computed according to their contextual distributions. In this work they experimented with four different metrics: (a) Kullback-Leibler, (b) Information-radious, (c) Manhattan-norm, and (d) Cosine similarity. A comparative study of these metrics about automatic induction can be found in [42]. During a system iteration a metric outputs an ordered list of pairs that is ranked according to semantic similarity. Each pair is assumed to form a semantic class. A predefined, fixed number of top pairs are used for the induction of semantic classes during for every system iteration. After every generation of induction classes the module of corpus parser is run. All instances of the generated classes are substituted in the corpus with the corresponding semantic label. It should be noted that the class labels are artificial tags, without denoting the class concept. The generated by the lexical phraser chunks are retained if they were assigned to an induced class by the semantic generalizer. In 3.3 the whole process is depicted.

In addition, in [39], an algorithm for unsupervised semantic class induction which is based on the hypothesis that similarity of context implies similarity of meaning, is described. Two semantic similarity metrics that are variations of the cosine similarity distance were used in order to measure the semantic distance between words and to automatically generate semantic classes. The first metric computes “wide-context” similarity between words using a “bag-of-words” model, while the second metric computes “narrow-context” similarity using a bigram LM. A hybrid metric was proposed as the linear combination of the wide and narrow-context metrics, using fixed weights, estimated over held out data during an a priori experimental procedure.

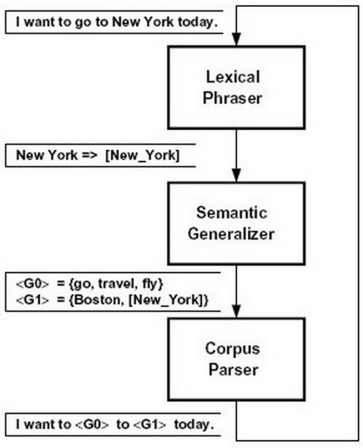


Figure 3.3: Auto-induced semantic classes system from [41]

3.4.3 Comparison with this thesis work

The approach that is followed in [31] for grammar induction, is the same with this that is used in this thesis. The main steps that are followed are the named entity recognition (*"Athens"*), induction of semantic classes (*concepts*) of terminals (*<CITY >= ("Athens", "Chania",..)*), extraction of grammar fragments (*chunks* e.g. "departing"<CITY >or "leaving"<CITY >) and induction of grammar rules (*<DEP_CITY >= ("departing"— "leaving")<CITY >*). The basic idea is the automatic creation of clusters that include semantically similar terminals. In more details, we depict the basic steps of the algorithm in figure 3.4.

As we can see in figure 3.4 the algorithm used begins from a corpus, which does not have any linguistic information, and the only hypothesis that it takes into account is that the named entities (*multi-word, e.g. New York → "New_York"*) are known.

$$\boxed{\text{Corpus}} \longrightarrow \boxed{\text{Similarity Computation}} \longrightarrow \boxed{\text{Clustering Algorithm}}$$

Figure 3.4: Grammar Induction's algorithm basic steps

Next step is the similarity computation between all words of the corpus. In 3.4.2 we mentioned four metrics used for similarity computation: Kullback Leibler, Information-radious, Manhattan-norm and cosine similarity.

The *Manhattan-norm (MN)* metric, which is used in our approach, can be considered as a geometric distance (*it is closely related with Euclidean distance*) and is, also, a true distance metric as the term "norm" indicates. It is computed as

$$D_{MN}(P\|Q) = \sum_{y \in Y} \mid P(y) - Q(y) \mid \tag{3.1}$$

The *MN* metric computes the absolute difference between the bigram conditional probabilty distributions W_1 and W_2. Due to the absolute function the *MN* metric is symmetric:

$$D_{MN}(P\|Q) \equiv D_{MN}(Q\|P)$$

The contextual distance between the bigram conditional probability distributions W_1 and W_2 is

$$D^R_{MN}(W_1 \| W_2) \equiv D^R_{MN}(w_1, w_2) = \sum_{v_{1,R} \in V} \mid P(v_{1,R} \mid w_1) - P(v_{1,R} \mid w_2) \mid \tag{3.2}$$

for the right contexts. In similar manner, the distribution distance for the left context is defined as

$$D^L_{MN}(W_1 \| W_2) \equiv D^L_{MN}(w_1, w_2) = \sum_{v_{1,L} \in V} \mid P(v_{1,L} \mid w_1) - P(v_{1,L} \mid w_2) \mid \tag{3.3}$$

Last, the left and right contextual distance (hence, dissimilarity) between words w_1 and w_2 is calculated as

$$D^{L,R}_{MN}(w_1, w_2) = D^L_{MN}(w_1, w_2) + D^R_{MN}(w_1, w_2) \tag{3.4}$$

Each of both terms of Equation 3.4 has a lower and upper bound of zero and two, respectively. Thus, two words of identical contextual distributions will have a zero value of *MN* distance, while a distance score equal to 4 indicates absolute dissimilarity.

Next, the induction of semantic classes is performed by applying agglomerative clustering, which is fed with the number of clusters and the pairwise semantic distances of the words of interest, using the CLUTO toolkit ([50]). The algorithm of agglomerative clustering, belongs to the the realm of hierarchical clustering, and forms clusters in a bottom-up manner. Initially each word is putted in its own cluster. Next, among all current clusters, the two with the smallest distance are picked and replaced with a new cluster, formed by merging the two original ones. This process is repeated until there is only one remaining cluster.

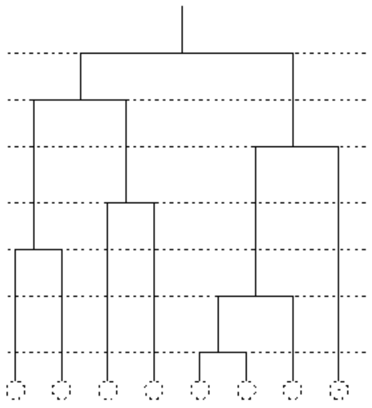


Figure 3.5: Example of agglomerative clustering

3.5 Summary

In this chapter we introduced to the reader the three basic areas of research that this thesis is based on. It includes a brief description of previous works and a comparison of them with this thesis's work. Its aim is to make the reader familiar with the state-of-the-art, and help him comprehend the contribution of this thesis. [3]

[3]Also used : [25], [40], [52], [26]

Chapter 4

Definition of Crowdsourcing tasks & Pilot study

The first approach to this thesis's task, consists of two phases. The first one is about the selection of the tasks that are going to be used and their abstract design, while the second is a pilot process for getting the necessary feedback from the users, before continuing with the final design.

The goal of this first approach is to help designers comprehend the purpose of the tasks and, consequently, succeed in an effective design. An effective design should provide the users with a usable interface, and achieve its basic goal, which is the collection of useful data from the users.

4.1 Introduction to the design of Crowdsourcing tasks

As we mentioned in 3.2 there are plenty of ways for collecting data through Crowdsourcing. The common characteristics among all these methods is that they should be accompanied with the appropriate background information, in order to help users to complete each task with success. According to [14] a task should minimize cost and completion time and maximize quality. In other words, a task should be cheap, fast and good. The payment of each task defines whether it is cheap or expensive. There is a trade-off among the high and low payment, because tasks with low payment are completed slower, but they do not attract contributors that want to "cheat", and on the other hand, tasks with high payment are completed faster, but they attract contributors that want to cheat.

Also, a crowdsourcing task has specific form that must be followed during the design. Firstly, every crowdsourcing task has a title, which must be descriptive and attractive to the contributors. Title and payment are the first information that contributors take for a task before deciding to complete it, thus the more attractive these information are, the larger the number of the contributors that want to complete it. Secondly, a task should have instructions, for helping users to understand exactly what they have to do for completing it with success. The instructions should provide all the necessary background information in clear and concise way. Thirdly, the main part of the task that is used for collecting data, must be driven of the nature of the data that we want to acquire.

Starting with the design of the tasks, we must consider a number of intermediate steps ([14]) before ending up with the final design. First and foremost, the reason for using Crowdsourcing e.g., what type of data we want to collect. Secondly, the way of obtaining these data. Last but not least, the way we present the tasks, including the information that we are going to provide them. We must keep in mind, that, these tasks are intended for humans. This makes their design more interesting and difficult at the same time, because of the subjectivity introduced by the human element. In figure 4.1 we depict the abstract format of each Task.

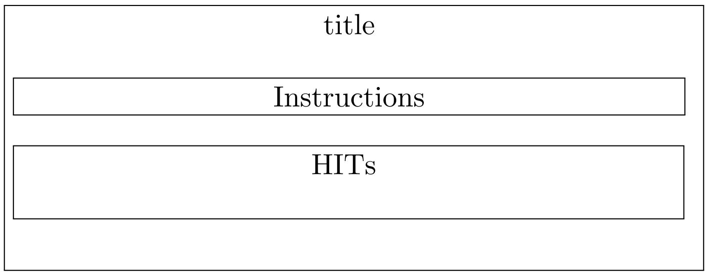


Figure 4.1: Abstract format of a Crowdsourcing task

4.2 Design of Crowdsourcing tasks

Bearing in mind all we described up to now, we have to define the content of the tasks. The data we want to collect are queries and answers that could be part of a real dialog during the interaction of a user and a SDS about travel-flight information. Furthermore, since a conversation for traveling with flights has a large range of concepts, it was more feasible to focus on a low number of concepts, in order to collect enough data for each of them.

Therefore, the tasks that were created should reflect the environment of a SDS, in order to help users provide more realistic and useful data. When we think of a SDS, the first that comes in mind, is the interaction between the system and the user and in table 4.1 we can see some examples of the data that we want to collect. Thus, we decided to create Question - Answer tasks. Furthermore, since we want to collect specific parts of travel domain, it was meaningful to create tasks, that the contributors were able to give only a part of a sentence, according to the context that they were given. Finally, we ended with five Crowdsourcing tasks.

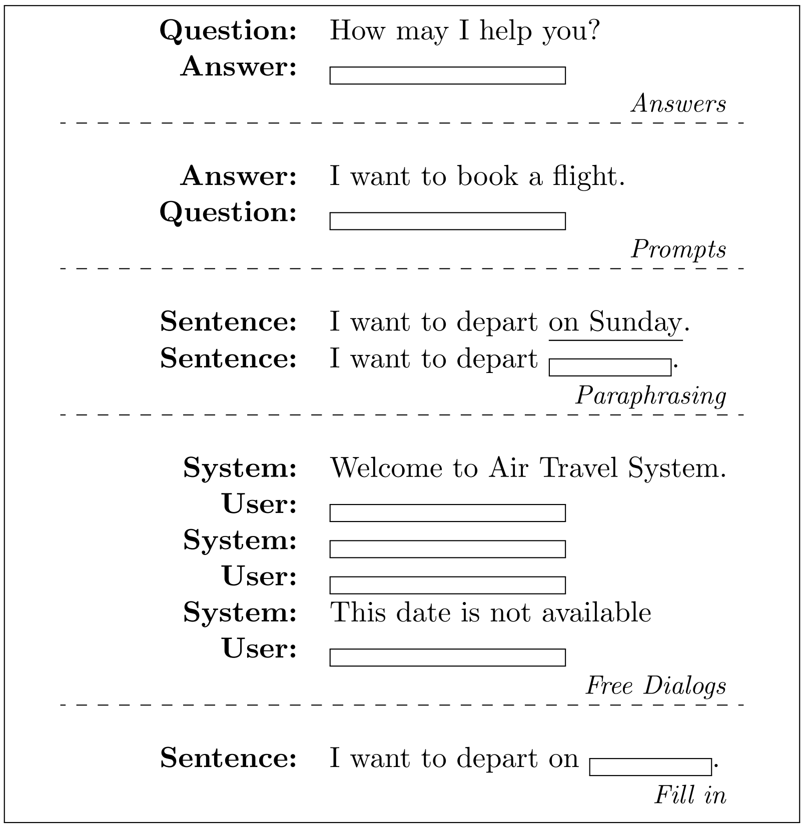
The first task, whose name is "Answer the questions" is composed of questions that we give as input, and empty fields, where humans must fill in with a suitable answer. In the second task, "Provide questions for the given answers", we give as input answers and ask people to give a corresponding question. "Provide phrases with the same meaning with the underlined", is a paraphrasing task, where we give as input a sentence with an underlined phrase and ask humans to give a phrase with the same meaning with the underlined. We selected to underline only a phrase and not the whole sentence, because it is more useful, since the workers will paraphrase only the parts of the sentence that we want to collect. The fourth task, with name "Complete the dialogues", is a kind of free task, where we have a dialogue with 3 turns[1] of user and system. Some of the system prompts and the user responses are given, and people have to complete the rest, in order to conclude to a dialogue that makes sense. In the last task, "Fill in the empty fields with the most appropriate word(s)", we give a sentence with a missing part and people have to fill in

[1]Each turn consist of a system prompt and a user response

this part. For the next document we are referring to the tasks described above with the following names: “Answers”, “Prompts”, “Paraphrasing”, “Free Dialogs” and “Fill in” respectively. Figure 4.2 depicts an example for each task, where the users must put their response in the empty boxes.

Examples of data that we want to collect	
SYSTEM:	And What Day In May Did You Want To Travel?
SYSTEM:	Traveling On What Date?
SYSTEM:	And You Said Returning On May Fifteenth?
USER:	Okay I Need To Be There For A Meeting That’s From The Twelfth To The Fifteenth
USER:	Leaving On February The Second
USER:	Yeah At The End Of The Day

Table 4.1: Examples of System Prompts and User Responses



Question:	How may I help you?
Answer:	☐
	Answers
Answer:	I want to book a flight.
Question:	☐
	Prompts
Sentence:	I want to depart on Sunday.
Sentence:	I want to depart ☐.
	Paraphrasing
System:	Welcome to Air Travel System.
User:	☐
System:	☐
User:	☐
System:	This date is not available
User:	☐
	Free Dialogs
Sentence:	I want to depart on ☐.
	Fill in

Figure 4.2: The form of the tasks that were designed

Parts of Grammar that we focus on

As we have already mention, we should apply our approach to a whole grammar about travel domain, but this is not feasible with one crowdsourcing procedure. The parts of the grammar that were selected are Date rules, Departure-city rules and top-level Prompts (*at the beginning of the dialog*). Date rules are used for expressing the possible ways that someone can describe a date. Departure-city rules, are used for describing the possible ways that someone can express the city that he is departing from. In Table 4.2 we represent a part of the grammar Date and Departure-city and in Table 4.3 some top-level prompts and the corresponding user-responses.

Examples of grammar rules

<DATE>	[“ON”/“ON THE”/“ON A”]<day><month >[<year>]
<DEPAR.CITY>	<attribute_departcity>“IS”<CITY>

Table 4.2: Examples of grammar rules about date and departure city

System prompts and corresponding user responses
Hello! this is Air Travel system, how may I help you?
Welcome to Air Travel system!Please give me your trip information.
Welcome to Air Travel system!Where would you like to fly to?
Hello I would like to make a reservation I’d like to leave Sunday.
February eighth and I would like a round trip to Dulles.
I’d like to fly to Seattle.

Table 4.3: Examples of Top-level prompts and corresponding responses

4.3 Pilot process

Before launching the on-line crowdsourcing process, and since we had no previous experience, it was necessary to begin the implementation phase with a pilot process. The feasibility of this process is that errors or ambiguous parts of the design can be detected and be corrected. Also, one outcome of this process is a dataset that can be used for applying several metrics, and be prepared when the final corpus will be collected.

4.3.1 Method

The pilot process was nothing else than an internal Crowdsourcing experiment that took place in Technical University of Crete. The participants were sixteen undergraduate and graduate students, with one of them being an expert user of English language(*resident of England*). Most of them were coming to the lab and were given sheet of papers with the tasks. They had as time as they needed for completing all five tasks and the order that they were given the tasks was random. The initial intention was to explain them the

Crowdsourcing process, and then let them complete the tasks without extra help, but, in some cases they were given a help clue, if they were really confused with the task. This, introduces a kind of discrepancy among the pilot and the real experiment, but the main purpose of the pilot was the evaluation from the users, and as a result this is not a big problem. The users except from the answers that they were requested to give, they also had to write down comments about the fuzzy parts or proposals for improvement.

4.3.2 Analysis of collected data

When the data collection phase of the pilot was over, we continued with the analysis of the dataset that had been collected. At first glance, we observed that the data that were collected from each task had some special characteristics. More specifically, it is worth to refer that in the "Paraphrasing" task the data that we collect depend completely from the data that we give as input, while with the "Free Dialogs" task we can collect data from various dialog parts. Next, we forwarded with the analysis of these data.

The main purpose of this analysis is to find out whether we achieved our goal, collecting utterances relevant to travel-flight domain and belong to specific dialog parts. Thus, we continued with an automatic evaluation which includes relevance metrics and a manual analysis that has the role of an authentication metric for being sure that the automatic analysis works correctly.

Corpora

The data we collected were organized into various corpora. The classification that we used was one corpus per task. As we observe from table 4.4 the corpora are too small, to do further classification, since we will not have enough data for extracting conclusions.

Corpus	Words	Sentences
Answers	716	127
Prompts	911	128
Paraphrasing	2119	236
Free Dialogs	3464	489
Fill in	1415	160

Table 4.4: Corpora that created after pilot process

The corpora have different size from each other and this happens because we designed different number of questions in each task. We decided to do so, because the tasks belong to different difficulty levels and we wanted to normalize this dissimilarity. We followed an approximate design in which we wanted to achieve a ratio of the time reading the instructions to completing the task(*acquisition and execution time respectively*) almost 0.1. This ratio means that the users spend the most of their time for completing the task, and not learning how to do it. Since we want to create "fast" tasks, the instructions should be proportional to task's difficulty and help users understand with clear and concise way what they have to do. Remember from 2.1 that a unit with acquisition time around 2 seconds has execution time around 20 seconds.

During the pilot process we collected these time measurements per user and we calculated the above ratio, whose value inform us how time-consuming was each task for the users. Tasks with ratio bigger than 0.1 were less time-consuming than tasks with ratio smaller than 0.1. We calculated the average for all users and we list the results in table 4.5 and the conclusions that we extract is that tasks “Answers”, “Prompts” and “Paraphrasing”, which are close to 0.1. On the other hand, the “Free dialogs” task was the most time consuming, thus the number of dialogs in the task should be smaller. In contrast to the “Free dialogs”, the “Fill in” task was the less time consuming, and according to the ratio, the users could bear almost the double number of HITs.

Task	Ratio
Answers Prompts	0.12
Paraphrasing	0.09
Free Dialogs	0.06
Fill in	0.22

Table 4.5: Ratio of reading instructions time to task completion time

Relevance metrics

The purpose of relevance metrics is to help us find out how relevant the gathered data are with travel-flight domain. One of the relevance metrics is a parser's success percentage.

We parsed the corpora created from each task, and we list the success percentages in table 4.6. We want the success percentage to be the highest possible, because this means that the data we collected, can be produced from parser's grammar. As a result, the data we collected are relevant to travel-flight domain.

Next, we used another metric, in order to find out how many from the sentences that were parsed, were about the grammar parts that we are interested in, in this case Date, Departure-city and prompts. The implementation of this metric takes as input the output of the parser that contains the parser tree of each sentence of the corpus that was parsed and finds how many of them were about the grammar parts that we focus on.

In manual evaluation, we examined all the answers that pilot participants provide us and we rejected some of the answers. Then we counted all the answers that were about Date or Departure-city or Prompts. It is the same metric with the previous one, with the deference that in this case the process is done manually. The results with manual and automatic evaluation have a divergence, because, the assumptions that were taken in the two cases were different. For example, in the automatic analysis it is not possible to find out if a prompt is top level. Also, the automatic analysis introduces errors in contrast to the manual.

From table 4.6 we observe that all the tasks collected in domain data. Also, the Date-Depar.city-Prompts metric provide a clue about the success of each task. It is obvious that the “Prompts” task has greater success collecting data that belong to specific parts of the grammar, possibly, because this task collects prompts. Also, the “Free dialogs” task produce the smallest percentage of utterances about the grammar parts that we want. The same happens to the “Answers” task, because users have the freedom to provide any utterance they think that is a logical answer, in contrast to the “Paraphrasing” and the “Fill in” task in which users' answers are restricted from the parts of the answers that are already given.

Task	Parser Success %	Sentences about Date-Depar.city-Prompts% (*Manually*)	Sentences about Date-Depar.city-Prompts% (*Automatically*)
Answers	89	70	76
Prompts	95		**89**
Paraphrasing	96	**91**	83
Free Dialogs	87	37	66
Fill in	**99**	75	86

Table 4.6: Relevance metrics for the data collected from pilot

Next, we measured in-domain success with a more specific way for each task. We separated the questions from which we expected to get answers about date and measured how many of the answers were date. We did the same for Departure-city and Prompts. The results are shown in table 4.7. This metric quantifies the success of task's design.

Task	Date %	Depar.city	Top-level Prompts %
Answers	70	54	25
Prompts	n.a	n.a	52
Paraphrasing	**83**	**62**	**81**
Free Dialogs	n.a	n.a	n.a
Fill in	56	44	63

Table 4.7: Design success metric for the data collected from pilot

We have to mention that in the "Prompts" task we have no results for Date and Departure-city. The reason is because with this task we only collect prompts. Furthermore, this metric cannot defined for the "Free dialogs" task, due the free form of the task. We have satisfactorily success percentages, and this means that the design of the tasks is good enough to give as answers in the domain that we want. As we observe, the "Paraphrasing" task has the greater in-domain percentages, and this happens because in this task we show to the users what we want type of data we want to collect.

Language richness metrics

Next step in data analysis, was perplexity metric. Perplexity inform us how well a language model matches to a test corpus and how rich (*linguistically*) is. Thus, it can be used as a language richness measure, expressing how "interesting" an utterance is.

When the perplexity value is low, the sentence is close to the language model, and consequently, this sentence, doesn't appear extra interest, since, it doesn't provide new information.

Perplexity is computed per sentence, so maximum and minimum perplexity belongs to a sentence, average perplexity is computed over all the sentences (*a mean value*) and total perplexity is computed from the toolkit over all sentences of the test set.

In tables 4.8, 4.9 we list perplexity statistics for unigram and bigram LMs. Note that maximum, minimum and mean perplexity is calculated per sentence, while total perplexity is calculated per corpus. We observe that bigram LM has lower perplexity than unigram LM, because it takes into account pairs of words which is an closer approximation to language. Thinking with the same way, bigger N-grams are expected to have lower perplexity.

High perplexity introduces suspicions for OOD, and this can be verified by examining mean and median perplexity, because mean metric is sensitive to the outliers. For example, in "Prompts", for the bigram LM, median metric is almost the 1/4 of the corresponding mean metric.

Corpus	Maximum	Minimum	Mean	Median	Total
Answers	2585	7.2	363.8	265	276.4
Prompts	8659	50.1	475.1	309	338.9
Paraphrasing	2205	37.8	397.1	261	296.5
Free Dialogs	3790	7.2	384.9	266	322.5
Fill in	1426	51.06	355.4	224	256.8

Table 4.8: Perplexity statistics for the various tasks using unigram LM

Corpus	Maximum	Minimum	Mean	Median	Total
Answers	9804	5.6	426.1	103	118.5
Prompts	25035	6.8	474.5	97	111.1
Paraphrasing	10882	6.0	561.6	116	113.7
Free Dialogs	10794	4.6	410.8	110	130.5
Fill in	6071	5.5	654.3	86	122.9

Table 4.9: Perplexity statistics for the various tasks using bigram LM

4.4 Conclusions

The pilot process can be characterized as a successful experiment since we collected numerous comments. Most users found the "Free Dialogs" task the most difficult and time consuming, in contrast to the"Fill in" task. The majority of the participants found the "Free Dialogs" task the most difficult among all the others, due to the many things that they had to write and the creation of large dialogs. One more difficulty, was that they had to think both from system's and user's side, and in some cases they had not only to provide answers or prompts, but also to create the concept of the dialog. The difficulty becomes higher counting in the need of imagination and the freedom that they were given in this task. Besides, one major problem is that if users have the ability, they write the same dialog turns. This problem can be solved be reducing the freedom that we allow to the users to have. On the other hand, the "Fill in" task, was the easiest for the most users, because it requires short answers and more specifically, only a part of a sentence. The rest tasks were almost in the same level of difficulty.

Moreover, the participants gave us some very helpful comments about fuzzy parts of the tasks and they were used for creating more useful and specific instructions. Furthermore, we observed, (*by examining the answers, and without quantified this observation in some* *way*) that the users were influenced from the answers that they had already seen, and they could not give enough variety. One additional problem, that shows the need of better instructions, was that they were given some answers relative to travel domain, but out of flight domain. However, the biggest drawback of the pilot process was that the majority of the participants are not expert users of English language.

By examining all the results, the appearance of trade - off, relevant to the freedom that we allow the users to have is obvious. More specifically, we notice tasks with almost the same behavior, had as a common characteristic the freedom that users were allowed to have. With the sense freedom, we mean the information that we give to the users in a HIT. When we give enough information the freedom of the task is low, because we restrict the possible answers that a user can give, in contrast to the tasks with high freedom, where the HITs have not enough information.

In both cases the data that we collect have advantages and disadvantages. When we give freedom to the users we get some interesting responses, but also many duplicates, because the users prefer to complete the task with the less possible effort. On the other hand, when we do not give freedom we get some “boring” responses, that don’t contain some new information, but we do not have duplicates. In an attempt to design a task in which the advantages will predominate over disadvantages, we should find a middle ground, to the freedom that we allow in each task. So, in the final design, we have to experiment with the information that we give in each task.

To sum up, the pilot results show that with low freedom we collect accurate data, but with high freedom we collect more “interesting” data but OOD as well. However, we can’t be sure, because of two drawbacks of the pilot process. The first is that the users were not native users of English and this means that they can’t think variety of ways of expressing their selves. The second is that it was not a representative experiment of Crowdsourcing, due the method that we followed and because the users hadn’t the the motivation of payment and, definitely, their motivation was in every aspect different than this of the Crowdsourcing workers.

4.5 Summary

In this chapter we analyzed the first steps of our work, including decisions about the design and the metrics of the analysis. The feedback from the pilot participants was the main aim of this first approach. Also, we collected data that we analyzed in respect to several metrics and this gave us an experience that we are going to use in the final design.

Chapter 5

Design

The next step of this thesis, is the re-design of the Crowdsourcing tasks. The re-design includes changes to the UI and the generation of the sentences that will be provided in each task and in each HIT.

5.1 User Interface

The UI is one of the most important parts of the Crowdsourcing tasks. The contributors (*Crowdsourcing workers*) interact with the system through the UI, and more specifically, the answers they submit are the data we collect. A usable UI not only makes the interaction of the system and the contributors easier, but also guides them to provide effective answers, and consequently, useful data. Thus, UI largely defines the success of the task.

The UI of the tasks can be divided into three branches:

1. The title part, which is very important, because it is the first information that a contributor gets about the task. The title should attract the contributors, in order to make them select it among numerous other tasks. Designing a usable task with a nasty title, should be avoided. Exaggerations must be avoided, as well.

2. The instructions part, is, maybe, the boring part of the task from the side of contributors, but, also, the most important for helping them learn the steps they have to follow in order to complete it with success. For these reasons, instructions must be brief but comprehensive, providing all the necessary information to the contributors in the less possible time. Instructions's purpose is to help contributors overtake the disambiguation points of the HIT.

3. HIT design is the main UI of the Crowdsourcing tasks. There is not guarantee that the instructions will be read, so the UI of the HIT should guide contributors to provide "correct" answers. Also, it should not contain unnecessary information, because this confuses the contributors.

The UI of the Crowdsourcing tasks

For each one of the Crowdsourcing tasks, we selected a title, which expresses the main purpose of the task with the lowest possible number of words.

The instructions we created were in the same style for all the tasks. We divided the information into bullets, because it is easier for the contributors to read them, in contrast to make them read the same information represented in one paragraph. Also, we classified instructions' content into small sections that make contributors act with a specific way, and we assigned one representative title to each of them, in the form of a prompt, for creating an environment characterized of immediacy, giving to the users the sense of an interactive system (*e.g. "What to do", "What to avoid"*). Also, for giving extra emphasis and attracting users' attention we used a coding technique based on colors. Humans are used to correlate the colors with behaviors. More specifically, green gives a good impression while red gives a bad impression. So, we used red, for the behavior that the contributors should avoid and green for the behavior that contributors should adopt. Furthermore, for the important parts of the instructions, that we wanted to be read, definitely, we increased font size, in contrast to the extra information that were clues for awesome answers.

The HIT design was different for each task, however, some characteristics were common for all of them. We tried to design tasks familiar to the contributors, giving the sense of a SDS even if there is not voice interaction. First of all, we wanted to be conspicuous to the contributors that the HITs (*units*) were independent of each other. For this reason, we inserted a delimiter in each task, according to the unit type, which denotes that a new unit begins. The unit types are: answer-question pairs, sentences and dialogs. Also, empty fields were part of the UI, used for the contributors' responses, with a default instruction inside them, about what they should add into these fields (e.g. *"Enter here a system prompt"*).

More specifically, in both tasks "Answers" and "Prompts", we provide first the question and next the answer, because for getting an answer one question must have been preceded. In paraphrasing task, we underline the phrase that we want to be paraphrased, and the same sentence is represented, with an empty field in the place of the underlined phrase. In "Free Dialogs", we represent in each dialog turn whether it comes from the system or the user side, and finally "Fill in" task consists only of one sentence with a missing part, represented with an empty field, that the contributor must fill in.

In Appendix C we show the titles and the UI of the tasks, as they uploaded to Crowdflower.

5.2 Design of the questions

The units design, including the questions and the piece of information that we are going to provide to the contributors is equally important with the UI design. The information that contains each unit, and especially the content of the questions, determines the quality of the data that we are going to acquire. Thus, we have to find out what is "this" that the input sentences have, and its change could increase or decrease the quality of the data that we collect. Still, we have to think how we can quantify it, in order to create an array of parameters, for each input sentence, per task, and then we have to find a correlation among the quality of the collected data and the parameter values.

5.2.1 Parameter definition

The tasks we designed try to achieve the same goal -collect data that belong to specific grammar parts- with different ways from each other. This means that some characteristics are common for all, while some others vary analogous the task. In essence, launching the parameters definition, for the input sentences, means that the first we must do, is to examine carefully the available information. Next, we take into account that each sentence that we provide as input into a unit:

1. Could be used during an interaction between a user and a SDS.
2. Has a defined length.
3. Is expressed more or less politely.
4. Is assigned into different parts of a dialog.
5. Expresses one or numerous concepts.

Furthermore, in some tasks we can experiment with the piece of information we provide to the units. For example, in "Free dialogs", we can make a unit free by decreasing the number of system prompts or user responses that we provide in each unit (*as input sentences*). In general, the less information we provide in each unit, the higher the freedom of the unit, and vice versa. Thus, the characteristics of the units (*and in most cases of the input sentences*) that determine the freedom of the units, can be expressed with the use of various parameters.

Except from the freedom of each unit, as discussed in the previous chapter, we reached the conclusion that each task allows more or less freedom to the contributors. In table 5.1 we list the tasks with decreasing freedom order.

$\Uparrow$ *High Freedom*	**Task**
	Free Dialogs
	Answers - Prompts
	Fill in
Low Freedom	Paraphrasing

Table 5.1: Crowdsourcing tasks in decreasing freedom order

Parameters defined for all tasks

The common, for all tasks, parameters are about the content of the sentence, in contrast to the parameters that are defined only for a subset of the tasks, and depend on the form of the task and the freedom that it allows to the contributors.

$\diamond$ **Question Specificity** : Qs and its values are the layers of the question. Big value means big specificity and value 0 means no specificity.

$$Qs = x : x \in S = \{0, 1, ..., n\}, n = \max\{levels\}, levels \in \mathbb{Z} \tag{5.1}$$

Where the levels are defined according to the dialog part that a sentence belongs to (*For our design n was 4*). Assuming that in the beginning of the dialog we ask more abstract information and the specificity increases as the dialog is being continued, we express this information, with Qs parameter, where the value increases with the specificity. An example of the level layers and representative sentences is depicted in figure 5.1.

What can I do for you?	LEVEL 0
When do you want to travel?	LEVEL 1
What date do you want to travel?	LEVEL 2
Give me the day and month you want to travel. Give me the month and year you want to travel.	LEVEL 3
Give me the day you want to travel. Give me the month you want to travel.	LEVEL 4

Figure 5.1: Example of sentences in various levels

□

◇ **Question Length** : Ql its values are numbers of words of each sentence.

$$Ql = x : x \in S = \{1, ..., n\}, n \in \mathbb{Z} \tag{5.2}$$

□

◇ **Number Of Concepts** : Noc and its values are the numbers of concepts that are expressed in each sentence.

$$Noc = \begin{cases} 0 & \text{defined for ``free dialogues''}\dagger \\ 1 & \text{if one concept is expressed} \\ 2 & \text{if two concepts are expressed} \\ 10 & \text{if more than three concepts are expressed} \end{cases} \tag{5.3}$$

Give me the date and the time you want to travel	Noc = 2
I want to fly to Athens	Noc = 1
I want to fly to Athens on Monday afternoon	Noc = 10
This is AirTravel system. How may I help you?	Noc = 10

Figure 5.2: Example of sentences in various Noc values

In figure 5.2 we can see some examples of sentences with various Noc values. Note that Noc takes the value ten, both in third and fourth example. In the first case, the sentence contains three concepts, but in the second case the sentence is a (*top-level*) system prompt. For reasons of fairness, when we assign values to the parameters, and we meet a system prompt, we think of a logical user response that could be an answer to the prompt and then we assign the value to the parameter according to this response. In fourth sentence, the value of Noc is 10 because, this prompt is too free to define what we are expecting to get as response.
† In "Free dialogues" separating the dialogues into pairs of question and answer, if the sentence is not given, Noc value is 0 and the same value have the rest parameters.

□

⋄ **Question Politeness** : Qp and its values are numbers among 0 and 5 that express the politeness of the sentence.

$$Qp = x : x \in S = [0, 5] \tag{5.4}$$

Where we can assign the cases of low, medium and high politeness to the values (0,1), (2,3) and (4,5) respectively. □

Parameters defined for a subset of the tasks

⋄ **Question paraphrase's length** : Qpl and its values are the numbers of words, that the phrase we ask to be paraphrased in each sentence, contains. This parameter is defined only for the "Paraphrasing" task.

$$Ql = x : x \in S = \{1, ..., n\}, n \in \mathbb{Z} \tag{5.5}$$

□

⋄ **Question's empty field position** : Qep and its values are the positions of the empty field in the sentence. In the case of "Paraphrasing" task, expresses the position of the underline phrase and in the case of the "Fill in" task, the position of the missing part of the sentence. This parameter is not defined for the rest tasks.

$$Qep = \begin{cases} \text{“left”} & \text{if the empty field is in the beginning of the sentence} \\ \text{“right”} & \text{if the empty field is in the end of the sentence} \\ \text{“center”} & \text{if the empty field is between two phrases} \end{cases} \tag{5.6}$$

□

⋄ **Dialogue's Freedom** : Df and it is defined only for the "Free dialogues" tasks. In essence, expresses the freedom that we allow to the contributors in each dialogue.

$$Df = \frac{empty_{fields}}{total_{fields}} \cdot 100 \tag{5.7}$$

Where $total_{fields}$ is the total number of fields in each dialog, and $empty_{fields}$ is the number of fields that contributors have to fill in. In our design $total_{fields}$ is always 6, thus this parameter takes specific values because we can provide from zero up to five dialogue fields. Thus the possible values are {100, 83.3, 66.6, 50, 33.3, 16.6}. □

To sum up, we depict in the next table, the parameters that are defined for each task.

Task \ Parameter	Qs	Ql	Noc	Qp	Qpl	Qep	Df
Answers	✓	✓	✓	✓			
Prompts	✓	✓	✓	✓			
Paraphrasing	✓	✓	✓	✓	✓	✓	
Free dialogues	✓	✓	✓	✓			✓
Fill in	✓	✓	✓	✓		✓	

Table 5.2: Tasks with their parameters

5.2.2 Sentences generation

In this section we analyze the process of designing a plethora of questions based on the data that are relevant to travel - flight domain and the parameters that we described in the previous section. In essence, we discuss how we think that each parameter can be used for providing data with the best possible quality and we explain the choices we made.

The "Grammar domains" that the collected data should belong to

The "Grammar domains" that the collected data should belong to, are Date and Departure-city, as well as, top level prompts. Thus, we have to define the number of questions that target to each of them. We list the design percentages (*calculated after defining for each sentence its target grammar domain*) in table 6.9, and as we observe, the grammar domain can be defined for all the tasks except from the "Free dialogues" because it is too free. All the questions of the "Prompts" task, as it is obvious, target to collect prompts, and all the questions of the "Answers" task target to collect user responses, thus half of them target to Date and the rest of them target to Departure - city. In the "Paraphrasing" and the "Fill in" tasks, we design only a low number of questions that target to prompts. We took this decision, because we wanted the majority of the data that we were going collect, to be used for grammar induction and the prompts are not suitable for this purpose.

Connection of parameters and questions

Next, we designed a number of sentences, system prompts and user responses, aiming to collect answers that belong to the target grammar parts. The inspiration for the these sentences were corpora collected with various ways from SDSs. (*From "Human to human" interaction, "Human to system", "Wizard of Oz", "Question - Answers"*). We list these questions in appendix A).

Next, we connected the initial questions with the parameters. First, we assigned values to the parameters in each sentence and then we created four new sentences keeping some parameter values the same and changing some others (*This is not always possible, because some parameters aren't independent of each other*). Totally we created 375 sentences and 40 dialogs.

The generation of the questions was done manually, and during this process we make some choices, trying to reach an effective design.

Using the parameters

The various values of each parameter have different affect on each sentence. Dividing the sentences to each parameter value uniformly, could be a way of using the parameters. However, we decided to follow an other technique, and more specifically, to create more questions with the values that will lead contributors to provide more effective data. The key idea behind this technique, is to try handle the freedom that we allow users to have in each unit. We believe that the contributors could give more effective answers when the freedom of the unit is neither too high, nor too low. The problem of high freedom is the collection of OOD utterances and the problem of low freedom is the collection of trivial utterances that don't carry new information. More specifically:

· Low values of **Qs** allow more freedom to the contributors. The more specific the question that we ask a contributor, the more the restriction that we introduce. Thus we prefer to design sentences with neither low nor high values.

· In the case of the “Prompts” task, the distribution of **Qs** changes, because, one characteristic of top-level prompts is that they have general concept, which means low specificity. Thus, the number of questions decreases with the increase of Qs value.

· For **Ql** parameter, we try to keep a uniform distribution, avoiding very large or low number of words. We can’t tell with certainty if questions with large number of words lead to answers with low or high information, because both scenarios are possible.

· The most preffered value of **Noc** is 1, because, using this means that we ask contributors to focus on a specific concept. This will lead contributors to provide answers that belong to a specific grammar rule with bigger probability.

· Furthermore, we avoid creating very polite or very rude questions, because this is more realistic for a SDS. Thus, the **Qp** values we prefer are 2 and 3. In theory, we expect from questions with high Qp to give answers expressed in formal way, due to the positive inspiration and questions with low Qp to give more straight and sharp responses.

· In a sentence with high **Qpl** parameter the contributors have to paraphrase a large phrase. Paraphrasing the important part of the sentence, which in this case, is the phrase that expresses either the date or the Departure - city is enough. As a result, it is satisfactory to avoid paraphrasing phrases with large number of words.

· We can change the **Qep** value according to the kind of answer that we want to collect. For example if we want to collect a greeting of the System, the most usual position is left. If we want to collect data about Departure - city, we prefer “left” and maybe we should give a city/country name. eg ”.......... is/to London”. This will drive users to give in domain data with various ways. Also, we should contain both “is” and “to”, for getting results in passive and energetic voice. On the other hand, if we want to collect data about date “right” maybe is a better choice. eg: “Departure date is......” or “I would like to travel (on)...”.

· Parameter **Df** expresses the freedom that we allow contributors in each dialogue. Freedom is defined from the system prompts or the user responses that we provide in each dialogue, so, the more utterances we provide the less free the dialog is. We don’t want to end up with an answer-question task, thus we prefer to provide the less possible utterances in order to avoid cheating and collect useful data.

In appendix B we depict the histograms of the parameters distributions, as we used them in the design stage, and in table 5.3 we list the parameter values according to the freedom that allow to contributors.

Freedom category	Qs	Ql	Noc	Df
Low Freedom	3,4	>13	2	33.3
Medium Freedom	2	>6&<=12	1	50, 66.6
High Freedom	0,1	<=6	0,10	83.3

Table 5.3: Assignment of parameter values to a freedom category

5.3 Summary

In this chapter we analyzed the process of the final design. We tried to quantify with numerous parameters, the basic characteristics of the sentences, we provide in each unit, that influence the quality of the data the we collect. Key point during the design is the freedom that we allow to the contributors, due to the trade - off we observed from the pilot study. A free design leads to the collection of useful data, with large diversity, but allows cheating behaviors as well. On the other hand a less free design, makes cheating behavior less possible, but the data collected lack diversity.

Chapter 6

Data Collection and Data Analysis

Up to the previous chapter, we have completed all the necessary processes for launching the data collection through Crowdsourcing. In this chapter we analyze the final step; the upload of the sentences we generated and the gathering process.

6.1 Collection process

The data we collect are the answers that the contributors submit during the task completion. At this point we must remind that contributors submit pages, and each page contains a number of units. The number of the units per page, was selected keeping in mind two key clues: the ratio of acquisition and execution time (*which had to be around 0.1*) and, the fact that we want contributors to provide high variety, and this means that we wanted to allow the less possible influence from the provided units. The mixture of these lead us to allow 5 units per page for the "Answers", the "Prompts" and the "Paraphrasing" task, 7 units per page for the "Fill in" task and 2 units per page for the "Free Dialogues" task.

The uploading process, as well as, some issues regarding Crowdflower are discussed in appendix C. The major problem that we have to face, is that we can't use the quality control system that Crowdflower provides. As we have already mention in previous chapter, Crowdflower uses "gold standard data" and in brief, the way it works is the following: Except from the questions we upload, we upload a few more with all the possible right answers. Then, in each page appears one of the gold questions, and the contributor is not allowed to continue unless he answers it correctly.

When a contributor selects a Job, he completes it in pages for a reward of some cents, and there is not restriction to the answers. Thus we can characterize the Jobs free enough to collect both useful and useless data. In total, 250 Crowdsourcing Jobs were ordered, 50 for each task, with total cost \$454.07 and data collection began on 13-04-2013 and ended on 27-04-2013 (*not equal data collection each day*).

In the case of NLP tasks, the contributors are called to create sentences, thus it is impossible to find all the possible right answers, due to the numerous permutations. This inserts an important problem, because there is not quality control for the data that contributors provide but simultaneously it is a challenge, since we have to think of alternative ways that could work as quality control. Finally, we ended up with two alternative methods: Crowdflower's flagging mechanism and Crowdflower's additional information, that were combined for achieving the best possible outcome.

Flagging mechanism

In Crowdflower, the requesters can monitor the answers the contributors provide, as soon as, one page is completed. Thus, if the answers are not what we are looking for, we can use the flagging mechanism, with which we denote the reason why the contributor flagged and we prevent him from completing other Jobs uploaded from us. The drawback is that the answers that he already provided can't be erased, and one more is that requires the monitoring all the active Jobs, as soon as, the pages are being completed. The benefit is that when a cheater is detected he is not allowed to complete other pages of the Jobs.

Additional information

Crowdflower has many options, that we try to take in advance for creating an alternative method for quality control. The main difference is that using additional information, in essence, we try to decrease the possibility of cheating, and not to reject answers from cheaters. We used the following additional information.

1. According to [14] **payment**, takes important role for the task completion time. When we want a task to be completed fast, we should provide a high payment to the workers. During our first uploading attempt we payed \$0.21 for the "Answers", \$0.23 for the "Prompts",\$0.22 for the "Paraphrasing" and the "Fill in" and \$0.33 for the "Free Dialogues" task. The truth is that the completion time was extremely low, but the data were not the expected. More than 50% of the acquired data was irrelevant data and copy paste. Thus, the first that we thought was to decrease the payment to the half for attracting fewer cheaters. The data we collected with this payment had less irrelevant data, while the time was still low. We continued with experiments and we ended up with a payment at around \$0.06 per page. The final conclusion is that with the low payment we collected better quality data than we did with high payment. The data were collected fast enough, and with less cost.

2. Acquiring the first data we realized that cheaters were not stopping submitting pages and as a result we were collecting more and more irrelevant data. Hence, we decided to use the Crowdflower's option for fixing the **maximum number of submitting pages per task** to 2. We decided so, because we have two possible scenarios to handle: either a cheater or a trusted worker submits a task. In the case of the trusted worker, we have the drawback that he can't provide more useful data and in the case of the cheater the benefit of preventing him from providing garbage. The cheater completes faster and more units instead of the trusted worker, so avoiding cheater's data is more important and this is the reason why we selected 2 pages.

3. The research that was done in [46], showed that the crowdsourcing workers that come from India found AMT money necessary to make basic ends meet in much higher percentage than Americans and the majority's annual income was less than \$10000. Workers that depend on Crowdsourcing money, usually want to collect the more possible, completing a large number of tasks in few hours. This could be a good reason for cheating and as a result we selected to **Exclude India** from participating in our tasks.

4. In addition, after examining the information of the first data we gathered, we noticed that contributors from some crowdsourcing channels, were providing garbage, while

contributors from other channels were providing useful data. This observation lead us to **exclude crowdsourcing channels** that their contributors were not providing useful data. An extra experiment was to allow the jobs only to the channel from which we collected the most useful data, AMT in our case. The problem was that the frequency of submitting tasks was very low, and some of these jobs never completed, and was the reason why we used and other channels except from AMT.

5. One extra strategy that we tried from preventing users from cheating was to **put warnings into the instructions**. We based on the assumptions that some of the workers that have the intention to cheat will not do it if they be warned about the consequences. We selected to give emphasis on the warning for two reasons, first, for attracting the contributors' attention and second for giving the sense to the contributors that this is really important and they should not ignore it.

The above were done experimentally, according to the data we were collecteing. Finally, from the data we collected we created five corpora, one per task, and one with their union, whose size we list in table 6.1. We have to mention that the corpora created from the "Answers", the "Prompts" and the "Free Dialogues" task, consist of data exclusively created from contributors, while in the rest tasks, the entries of the corpora are a mixture of data inserted from contributors and input sentences. In these tasks in contrast to the previous, contributors insert only a phrase that is a part of a sentence, so the entry in the corpus is the sentence that is being created from the part provided to the contributors and the part provided from the contributors.

Corpus	Words	Sentences
Answers	16339	3435
Prompts	21721	3515
Paraphrasing	41108	4803
Free Dialogs	41500	7543
Fill in	62775	5996
Total	189443	25289

Table 6.1: Corpora that created from Crowdsourcing

6.2 Data analysis

Goal of data analysis is to find out whether the design of the Crowdsourcing tasks was effective. An effective design leads to the collection of useful for grammar induction data. For the data analysis we use various metrics and numerous corpora. We create corpora according to the parameter values, the grammar domains that we focus on and the filters. Then we observe for which of the corpora better performance is achieved.

6.2.1 Filters

In an attempt to improve the quality of the total corpus that we collected, we used two filters. The first filter is the removing of irrelevant data (*garbage: sentences that created from words with random characters*) and it is applied to the corpus manually. The second filter includes the automatic removing of the answers that were provided by flagged contributors.

We parsed the corpora before and after the filters, in order to get an idea about the filters' influence on the corpora, and we list the results for the total corpus in table 6.2. In table 6.3 we list the tasks with decreasing freedom order, the percentages of answers that came from flagged contributors and the percentages of irrelevant data (*garbage*). Note that, providing irrelevant to travel - flight domain answers or copy - paste answers are the reasons for flagging users. Thus, garbage filter implies flag filter.

Corpus	Size(lines)	Parser Success %
Initial	25289	86
After Garbage filter	23620	90
After Flag filter	20745	**91**

Table 6.2: Comparison of the corpus we collected and the corpora created after filtering

Task	Size before filters(lines)	Parcer success before filters %	Answers from flagged users %	Garbage %
Free Dialog	7540	78	18	9.7
Answers - Prompts	3453, 3515	83, 80	18, 22	3.9, 6.2
Fill in	5996	96	15	5.6
Paraphrasing	4803	94	19	4.9

Table 6.3: Parser success and percentages of garbage data and answers from flagged users

From the above tables, we observe that the filters increase the parser success. However, the difference is low, especially among the corpora created from the filters. Thus, in the analysis that follows, only the flag filter is used, since it is automatic. However, the parser success is high enough to say that the majority of the data we collected was relevant to travel - flight domain. In addition, we note that, the higher the freedom, the lower the parser success. This happens because the contributors can't always handle the freedom that they are allowed in order to provide useful data. Therefore, we were expecting higher parser success and lower flagged percentages for the "Paraphrasing" task. One factor that influences these results is the order that the tasks were uploaded. The later uploaded tasks had the benefit that some contributors had already been flagged, after the monitoring of the previous uploaded tasks. The order most times was "Answers", "Prompts", "Paraphrasing", "Free Dialogues", "Fill in".

6.2.2 Grammar Induction

Grammar induction was not the object of this thesis, but the purpose it was based on. Thus, in table 6.4 various grammar induction performance metrics are listed, and their purpose is to help us find out whether we achieved our goal. A comparison with corpora created using other methods, such as web harvesting ([31]), provides a total image about which method is preferable for collecting data for grammar induction.

Corpus	Words	Precision	Recall	Fmeasure	Terminal Concepts	Terminal Instances
Initial	189443	**0.51**	**0.38**	**0.44**	**11**	**167**
Flag Filter	164356	**0.51**	**0.38**	**0.44**	**11**	164
“Answers”	16339	0.51	**0.43**	0.47	7	**127**
“Prompts”	21721	0.41	0.20	0.27	2	16
“Paraphrasing”	41108	0.42	0.31	0.36	8	46
“Free Dialogues”	41500	0.56	0.42	**0.48**	**10**	121
“Fill in”	62775	**0.57**	0.40	0.47	8	95
Date	58272	**0.54**	**0.44**	**0.49**	8	109
Depart-city	57048	0.45	0.38	0.41	**12**	**123**
Prompts	43194	0.41	0.25	0.31	2	26
Qs = 0	49788	0.55	0.41	0.47	7	106
Qs = 1	44303	0.46	0.37	0.41	**9**	**125**
Qs = 2	46140	**0.57**	**0.43**	**0.49**	**9**	110
Qs = 3	12449	0.53	0.42	0.47	7	75
Qs = 4	17501	0.51	0.41	0.46	7	75
Noc = 0	21813	**0.58**	0.40	**0.47**	8	88
Noc = 1	64870	0.51	0.37	0.43	**11**	**139**
Noc = 2	44542	0.53	**0.42**	**0.47**	8	100
Noc = 10	38956	0.56	0.38	0.45	7	71
Qp = 0, 1	27923	0.56	**0.41**	0.47	9	98
Qp = 2, 3	117769	0.51	0.37	0.43	**10**	**147**
Qp = 4, 5	24489	**0.57**	**0.41**	**0.48**	7	79
Ql<=6	59683	**0.53**	**0.42**	**0.47**	**9**	133
6<Ql<=12	92116	0.49	0.39	0.43	**9**	**138**
Ql>=13	18382	0.40	0.33	0.36	6	69
Qep = -1	25010	0.50	0.37	0.43	7	62
Qep = 0	47479	**0.55**	**0.38**	**0.45**	**9**	**79**
Qep = 1	40404	0.48	0.37	0.42	7	60
Qpl<=4	19590	0.35	0.23	0.28	**8**	27
Qpl>4	28577	**0.45**	**0.30**	**0.36**	4	**30**
Df>=66.6	34232	**0.56**	**0.41**	**0.48**	**7**	**111**
Df<66.6	7261	0.52	0.38	0.44	5	58

Table 6.4: Grammar Induction performance metrics (using a subset of the grammar)

The results of table 6.4 were produced from the grammar induction algorithm of [31] and includes various performance metrics, assessing how easy is to mine information from the text. An example of a terminal concept is : <cityname>and an example of terminal instance is <cityname>→ NEW YORK. Precision expresses the accuracy, recall the coverage of the algorithm and fmeasure is a combination of the two metrics. The values they take belong in [0,1] and good performance is indicated from values close to 1. Terminal concepts inform as how rich are the collected data, with the sense that they include numerous concepts. We must note that the results of grammar induction were produced using only the parts of the grammar that the data we wanted to collect should belong to. We observe that the more free task, “Free dialogues”, is the richest, since it collects the higher number of terminal concepts. Also, as we expected, the “Prompts” task, which designed for collecting system prompts, is an outlier, because the number of the terminal concepts that it “finds” is extremely low. The results for the Prompts corpus confirm this state. Looking the corpora that were created from the various parameter values, we observe that the richer outcomes were achieved for the values that allow neither high nor low freedom, with the richest being for the corpus Noc = 1 and Qp = 2 or 3.

In table 6.5 we list the same metrics with table 6.4, but we have kept only Date concepts. This analysis helps to examine individually the grammar induction with respect to rules about Date. The results are similar with the results in table 6.4, but we, also, noticed that we can extract information about the effect of the design to the grammar induction algorithm. For the initial corpus the grammar induction algorithm found 64 terminal instances, while for the corpus about Date domain found 63. An experiment in which we kept only concepts about Departure-city was done, and 116 terminal instances detected in the initial corpus, while 101 were detected in Depart-city corpus. These results indicate that the design of the tasks was successful, because the number of terminal instances and concepts in the domain corpora is almost the same with the corresponding numbers in the initial corpora.

Also, the grammar induction algorithm includes a clustering method, in which a number of clusters is being created and in the next figures we depict the Fmeasure as a function of them. In figure 6.1 we observe that the difference between the initial corpus and the corpus after flag filtering is very low. However, the best Fmeasure is achieved for the corpus after flag filter. We represent the Fmeasure for various filters for some of the corpora, in figures 6.2, 6.3, 6.4, 6.5, 6.6, 6.7. The result that we did’t expect was the low performance of the “Paraphrasing” task.

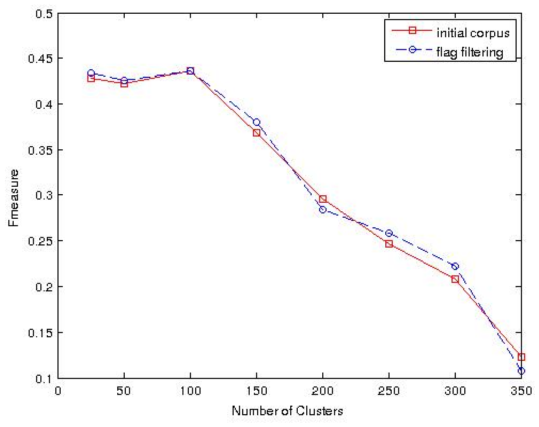


Figure 6.1: Fmeasure for initial corpus and corpus after flag filter for various number of clusters

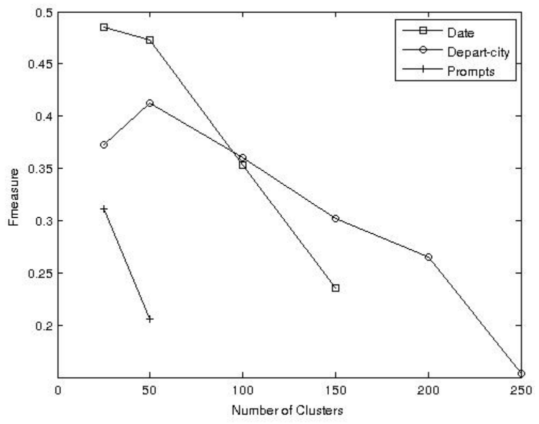


Figure 6.2: Fmeasure for domain corpora for various number of clusters

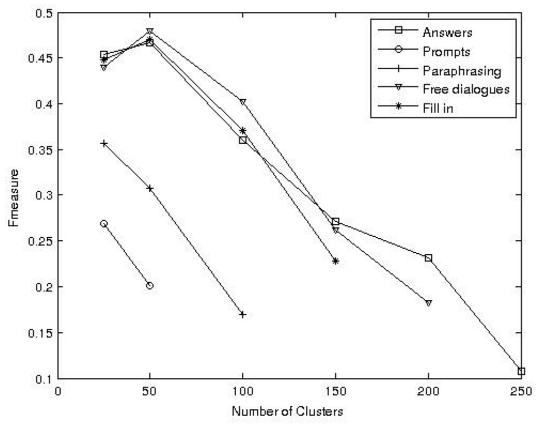


Figure 6.3: Fmeasure for task corpora for various number of clusters

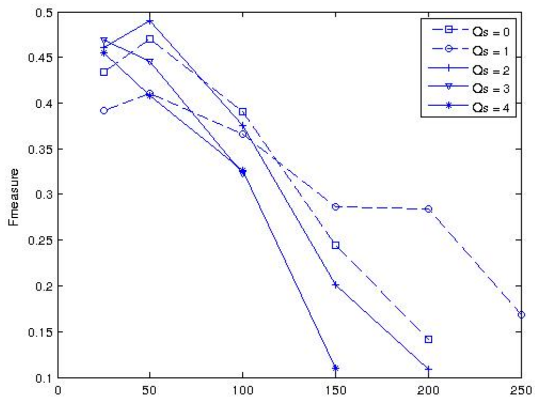


Figure 6.4: Fmeasure for Qs corpora values for various number of clusters

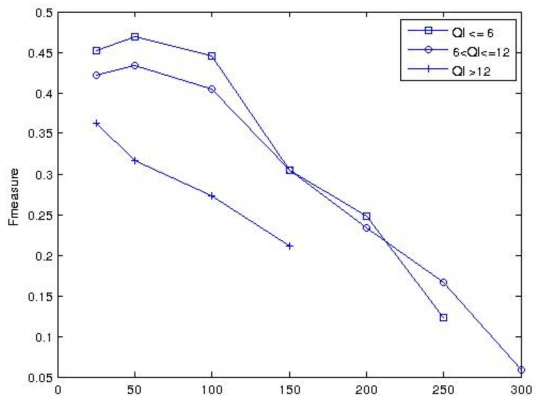


Figure 6.5: Fmeasure for Ql corpora for various number of clusters

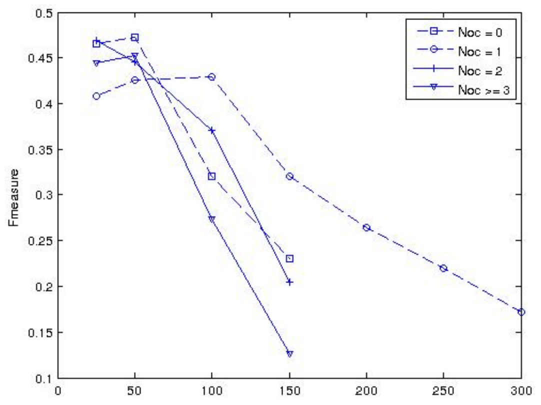


Figure 6.6: Fmeasure for Noc corpora for various number of clusters

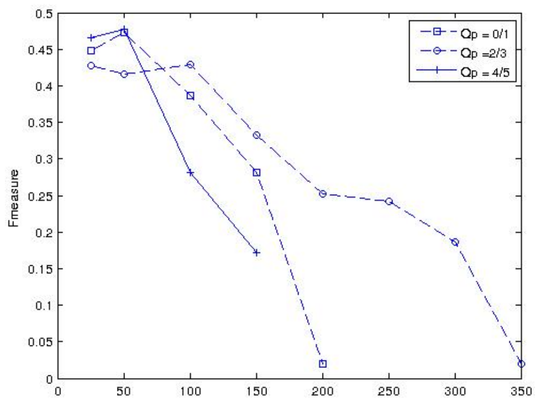


Figure 6.7: Fmeasure for Qp corpora for various number of clusters

Corpus	Words	Precision	Recall	Fmeasure	Terminal Concepts	Terminal Instances
Initial	189443	0.54	**0.36**	**0.43**	6	**64**
Flag Filter	164356	**0.58**	0.34	**0.43**	**8**	59
"Answers"	16339	**0.62**	**0.43**	**0.50**	4	40
"Prompts"	21721	0.50	0.08	0.14	2	4
"Paraphrasing"	41108	0.41	0.25	0.31	6	28
"Free dialogues"	41500	**0.62**	0.40	0.49	**8**	**55**
"Fill in	62775	0.57	0.37	0.45	7	**55**
Date	58272	**0.56**	**0.39**	**0.46**	**7**	**63**
Depart-city	57048	0.41	0.23	0.30	5	24
Prompts	43194	0.40	0.10	0.16	2	6
Qs = 0	49788	0.55	0.33	0.41	6	39
Qs = 1	44303	0.55	0.36	0.44	**8**	**52**
Qs = 2	46140	0.57	**0.37**	**0.45**	6	**52**
Qs = 3	12449	**0.61**	0.34	0.44	5	31
Qs = 4	17501	0.56	0.28	0.38	5	22
Noc = 0	21813	0.54	0.31	0.40	**7**	35
Noc = 1	64870	**0.55**	**0.37**	**0.44**	5	**59**
Noc = 2	44542	**0.55**	0.33	0.41	5	42
Noc = 10	38956	0.54	0.35	0.43	6	36
Qp = 0, 1	27923	**0.58**	**0.34**	**0.43**	**8**	41
Qp = 2, 3	117769	0.52	0.34	0.41	7	**56**
Qp = 4, 5	24489	0.49	0.30	0.37	3	30
Ql<=6	59683	**0.60**	**0.39**	**0.47**	**6**	**56**
6<Ql<=12	92116	0.51	0.34	0.41	**6**	54
Ql>=13	18382	0.44	0.26	0.33	5	24
Qep = -1	25010	**0.49**	0.30	0.37	6	33
Qep = 0	47479	0.46	0.28	0.34	6	36
Qep = 1	40404	**0.49**	**0.31**	**0.38**	6	**38**
Qpl<= 4	19590	**0.40**	0.20	0.26	**5**	17
Qpl>4	28577	0.38	**0.21**	**0.27**	4	**19**
Df>=66.6	34232	**0.63**	**0.39**	**0.48**	6	**52**
Df<66.6	7261	0.57	0.32	0.41	**7**	30

Table 6.5: Grammar Induction performance metrics (using a subset of the grammar :only Date)

6.2.3 Parser analysis

Parser based analysis assesses the information that is available in a corpus. The parser that we use has been described in 2.4. The performance metrics of this analysis are listed in table 6.6 and for reasons of consistency with the grammar induction performance metrics, the analysis was done using a subset of the grammar that contains only rules about Date and Departure-city. Grammar rules are the higher level rules like <TOCITY>and they form grammar fragments like <ARR.CITY>→<TO <cityname>. The results are similar with the results of grammar induction, however the less free tasks have higher percentages in parser success, in contrast to the more free tasks that are richer regarding the grammar concepts, rules and fragments. From the values of the parser analysis performance metrics we can mention that the tasks that allow less freedom to the contributors have higher parser success comparing with the more free tasks, but they lack linguistic richness. Generally, some of the values that we end up with, are comparable with the corresponding values from [31].

Corpus	**Sentences Partially Parsed** (%)	**Terminal Instances** (per word)	**Grammar Fragments** (per word)	**Terminal Instances** (unique)	**Terminal Concepts** (unique)	**Grammar Fragments** (unique)	**Grammar Rules** (unique)
Initial	67.8	**0.16**	**0.11**	**425**	**24**	**108**	**11**
Flag Filter	**72.1**	**0.16**	**0.11**	421	**24**	95	**11**
"Answers"	72.0	**0.21**	0.12	274	20	49	8
"Prompts"	44.7	0.09	0.03	71	19	22	10
"Paraphrasing"	82.2	0.17	0.12	146	**23**	68	**11**
"Free Dialogues"	53.2	0.15	0.07	**280**	22	**73**	**11**
"Fill in"	**85.7**	0.18	**0.15**	222	**23**	71	**11**
Date	81.8	0.19	0.13	235	**23**	**78**	**11**
Depart-city	**85.6**	**0.21**	**0.18**	**304**	19	55	9
Prompts	50.5	0.09	0.02	99	19	26	10
Qs = 0	57.4	0.14	0.07	246	19	62	10
Qs = 1	77.8	0.19	0.15	**301**	**23**	76	**11**
Qs = 2	77.5	0.18	0.12	242	**23**	**77**	**11**
Qs = 3	**93.0**	**0.26**	**0.22**	164	21	53	**11**
Qs = 4	72.6	0.14	0.09	173	18	39	8
Noc = 0	50.5	0.14	0.07	210	18	58	10
Noc = 1	69.6	0.15	0.08	**354**	**24**	**80**	**11**
Noc = 2	**93.4**	**0.20**	**0.17**	222	23	71	**11**
Noc >= 3	74.4	0.19	0.14	183	20	52	9
Qp = 0, 1	52.8	0.14	0.06	229	20	63	10
Qp = 2, 3	**79.2**	**0.19**	**0.14**	**383**	**24**	**89**	**11**
Qp = 4, 5	67.1	0.14	0.06	183	21	56	**11**
Ql<=6	59.8	0.15	0.08	309	23	78	**11**
6<Ql<=12	80.0	**0.19**	**0.14**	**332**	**24**	**84**	**11**
Ql>=13	**85.8**	0.18	0.13	198	22	60	10
Qep = -1	86.4	0.18	0.13	157	**22**	52	**11**
Qep = 0	**87.9**	**0.20**	**0.17**	**189**	**22**	**67**	**11**
Qep = 1	77.9	0.14	0.09	159	21	64	10
Qpl<= 4	**85.1**	**0.20**	**0.15**	**111**	21	47	**11**
Qpl>4	79.6	0.14	0.09	110	**22**	**57**	10
Df>=66.6	51.9	0.14	0.07	**258**	19	**67**	**11**
Df<66.6	**59.5**	**0.17**	**0.09**	141	**21**	52	9

Table 6.6: Parser analysis performance metrics using a subset of the grammar

6.2.4 Perplexity

The perplexity as evaluation metric has been discussed in 2.2. For calculating the perplexity, we built the language models, using a corpus consisting of four travel domain corpora viz.: "human to human", "human to system", "wizard of oz systems"[1] and"questions and answers". Then, we can calculate the N-grams and we list the various statistics of perplexity for bigram LM in table 6.7. First, we observe that the very large values, that indicate OOD bigrams, come from the more free tasks, or parameter values.

Corpus	Size(Words)	Mean	Median	Max	Min	Total
Total	189443	861	166	42751	5	222
"Answers"	16339	406	57	37283	4	72
"Prompts"	21721	1,236	520	28821	5	519
"Paraphrasing"	41108	691	167	27120	7	193
"Fill in"	41500	814	122	42751	5	191
"Free Dialogs"	62775	970	204	33161	6	248
Date	58272	281	88	37283	5	112
Depart-city	57048	624	101	37283	4	138
Prompts	43194	756	353	14886	7	346
Qs = 0	49788	931	169	42751	5	243
Qs = 1	44303	502	96	38707	5	129
Qs = 2	46140	1030	170	27120	5	261
Qs = 3	12449	483	156	18603	5	181
Qs = 4	17501	791	187	24186	5	230
Noc = 0	21813	792	114	42751	5	169
Noc = 1	64870	660	119	38707	5	163
Noc = 2	44542	1065	184	27120	7	247
Noc = 10	38956	915	272	33161	5	268
Qp = 0,1	27923	825	122	42751	5	188
Qp = 2,3	117769	752	145	38707	5	200
Qp = 4,5	24489	1029	207	33161	5	250
Ql<= 6	59683	872	158	42751	5	218
6<= Ql<= 12	92116	802	152	34261	5	213
Ql>= 13	18382	492	97	22683	5	133
Qep = -1	25010	1259	318	27120	7	353
Qep = 0	47479	545	114	33161	6	138
Qep = 1	40404	969	338	17172	7	328
Qpl<= 4	19590	610	135	27120	7	159
Qpl>4	28577	823	244	18603	7	241
Df>=66.6	34232	823	122	42751	5	189
Df<66.6	7261	769	134	21416	5	199

Table 6.7: Perplexity statistics for the various corpora, for Bigram LM

.

[1]In this case the users think that they interact with the system, but actually, they interact with a human from system's side

Also, it is obvious (*and with this metric*) that the “Prompts” task is an outlier, and for this reason it is excluded from the rest corpora. On the other hand, the “Answers” task has the lowest median and mean perplexity. These values indicate the effect that the two tasks have on the contributors, and it is obvious that the contributors provide more useful data, when they take a “role” that they are familiar with (*being the users*), in contrast to the case that they take the “role” of the system (*in “Prompts” task*). In total corpus the very big difference between the mean and the median value, indicates the existence of OOD, but majority of the corpora seems to be in-domain.

Correlation between input and output perplexity

The curiosity to find out whether there is correlation between the sentence we provide as input to the tasks and the response that the contributors provide, drive us to create a correlation metric based on perplexity. More specifically, we create two vectors with elements the perplexity of the input sentence and the corresponding phrase provided by the contributor. The way we continued is based on the following observations:

1. We can’t find possible correlation for “Free dialogs” task, because, many of the input sentences are blank.
2. In the “Answers” and the “Prompts” tasks, we find the possible correlation between the input sentence and the response provided by the contributors.
3. In the “Fill in” task we provide the contributors only a part of the sentence and they provide us a phrase that completes the missing part. Trying to find the correlation between the two sentences, the one with the missing part and the one that is being created after the contributor fills in the empty field, we are going to produce a “fake” correlation, because the first sentence is one part of the second. Thus, we find the correlation between the sentence with the missing part, and the phrase that the user provides. This is going to give us an idea about how the contexts influence the the answers provided by the contributors.
4. In the “Paraphrasing” task, instead of the missing part, we had in the “Fill in” task, we have an underlined phrase. So, we discern three cases and in each of them we find the correlation between the phrase that the contributor provides and the whole sentence given, the underlined phrase, and finally the context of the underlined. The last is done for comparing with the “Fill in” task.

The correlation outputs are listed in table 6.8. The values we ended up with, prove that there is not high correlation between the given and the output phrases. When only the phrase that comes from the contributor is used, the correlation is much smaller. Note that the correlation on the “Paraphrasing” task is higher between the underlined phrase and the paraphrase. Also, in the “Fill in” task the contexts have higher correlation with the phrase provided by the contributors, than in the “Paraphrasing” task.

6.2.5 Design success metric

This metric informs us how effective the design of the input to the tasks sentences was. In essence, we examine how many of the questions that were designed for collecting

Task	Correlation	Correlation between
Answers	0.039	given system prompt & user response provided by the contributor
Prompts	0.004	given user response & system prompt provided by the contributor
Paraphrasing	0.114	whole sentence given & paraphrase provided by the contributor
	0.130	underlined phrase given & paraphrase provided by the contributor
	0.022	context of the underlined phrase given & paraphrase provided by the contributor
	0.378	whole sentence given & union of the context of the underlined phrase given and the paraphrase provided by the contributor
Fill in	0.073	part of sentence given & phrase provided by the contributor
	0.591	part of sentence given& union of part of sentence given and phrase provided by the contributor

Table 6.8: Correlation per task, based on perplexity and calculated between the data that we provide and the data that are provided by the contributors

answers that belong to each of the target domains lead contributors to provide answers that belong to this domain.

In table 6.9 we list the percentages of the questions that were designed and uploaded for collecting data that belong to Date or Departure-city domain. Note that the "Prompts" and the "Fill in" tasks, are not contained in this design because the first is used for collecting prompts, and the second is too free to define what we expect to collect for each sentence.

Task \ Questions for collecting	Date %	Departure-city %	Date & Departure-city%
"Answers"	38	44	18
"Paraphrasing"	40	40	0
"Fill in"	35	37	9

Table 6.9: Percentages of the answers that target to collect data that belong to each domain, per task

The implementation of this metric is based on the parser output and more specifically, we examined the grammar rules that the parser assigned to the answers of these questions. We assume success to assign to Date and the question to be about Date and so forth. In table 6.10 we list the values of this metric.

Task	Date %	Departure-city %
Answers	21	19
Paraphrasing	**49**	**42**
Fill in	39	38

Table 6.10: Data that belong to Date and Depar-city domain and were provided as answers to questions that were targeted in the corresponding domains

As we observe, tasks with lower freedom have greater success in this metric, because it is easier for strict tasks to drive contributors to provide a specific type of answers. In addition it is easier to collect answers about Date than Departure - city, due to the form of the tasks and some key words that bias contributors to provide specific kinds of answers.

6.2.6 Meta-data analysis

With the term Meta Data analysis, we mean the analysis of data about the data that were collected. In this analysis we calculate some statistics about the information that we were provided by Crowdflower, regarding the contributors who submitted the tasks. In table 6.11 various statistics about the contributors are listed. Also, in table 6.12 the percentages of contributors that submitted a number of units per task, that belongs to a specific range, are listed.

"Answers"	
Total units	3435
Contributors(unique)	208
Flagged Contributors	39
Cities(unique)	173
Crowdsourcing channels(unique)	28
"Prompts"	
Total units	3515
Contributors(unique)	155
Flagged Contributors	43
Cities(unique)	135
Crowdsourcing channels(unique)	26
"Paraphrasing"	
Total units	4803
Contributors(unique)	213
Flagged Contributors	48
Cities(unique)	179
Crowdsourcing channels(unique)	26
"Free Dialogues"	
Total units	1874
Contributors(unique)	187
Flagged Contributors	45
Cities(unique)	158
Crowdsourcing channels(unique)	26
"Fill in"	
Total units	5996
Contributors(unique)	204
Flagged Contributors	33
Cities(unique)	176
Crowdsourcing channels(unique)	26

Table 6.11: Various statistics about contributors per task

From table 6.12 it is obvious that the "Free dialogues" task was the more labored,

Task \ Units submitted per contributor	<10 (%)	15-25 (%)	30-50 (%)	>50 (%)
Contributors of “Answers”	**71**	16	5	6
Contributors of “Prompts”	**63**	18	7	11
Contributors of “Paraphrasing”	**63**	19	7	6
Contributors of “Free Dialogues”	**80**	10	4	4
Contributors of “Fill in”	19	**52**	15	12

Table 6.12: Percentages of contributors that submitted a number of units per task, that belongs to a specific range

since the majority completed up to 10 units, and the “Fill in” task was the most facile, since the majority of contributors submitted between 15-25 units. The variety of the contributors who submitted the tasks implies the variety of the collected data, thus we prefer to have small percentages in the large number of units and, as we observe, this is achieved for the majority of the tasks.

6.3 Summary

In this chapter we described the data collection process and the analysis of the data we acquired, with respect to several metrics. In brief, all metrics ended up with the same conclusion: the free tasks provide larger variety in the data that they collect, but the strict tasks provide better accuracy. Creating corpora trying to balance these two limit cases, we can collect data that combine both variety and accuracy. Also, from these metrics we get a feedback about the success of the tasks’ design. From the corpora that were created from the parameter values we can see the effective values of each parameter. Bearing in mind that the tasks didn’t include an automatic quality control during the collection phase, we can tell that we achieved a good design, and the data that were collected were useful enough.

Chapter 7

Conclusions and Future Work

In this chapter we sup up the results and the conclusions of this thesis, and also, we recommend some ideas for future work.

7.1 Conclusions

The conclusions of this thesis come from the results about the design of the Crowdsourcing tasks, and the performance of the grammar induction algorithm for the various corpora. Finally, comparison with previous work, is included in order to find out which method for collecting data works better.

Crowdsourcing tasks

Since the data we wanted to collect are going to be used for the development of a SDS, the tasks that we designed aimed to reflect the environment of a SDS. For this reason we designed tasks that contained system prompts and user responses, from dialogue parts from a potential interaction between a system and a user. These tasks (*"Answers"*, *"Prompts"*, *"Free Dialogues"*) allow high freedom to the Crowdsourcing participants, since they can provide whatever they want without restriction. Thus, we created two more tasks, in which the users were allowed less freedom, and they had to complete logically a part of a sentence. So, we have two basic type of tasks, classified according to the freedom that they allow to the contributors. The results showed that the freedom of a task influences the contributors and the corpora created have different characteristics. The free tasks give the users the opportunity to provide variety of responses, in contrast to the other tasks, that offer, in-domain and accurate data. The more free tasks seem to work better, because they can provide variety of data. Also, the design of the tasks, as we explained, was successful enough.

Also, the design of the tasks was based on a plethora of parameters. We showed that the values of the parameters have specific affect on contributors and lead them to provide more or less useful responses.

To sup up the conclusions we reached regarding the design decisions are the following:

1. We prefer free tasks, that allow users to express themselves.
2. We prefer questions that don't confuse the users and don't create conflicts about the possible "right" answers (Noc = 1).

3. We prefer questions with medium specificity, because it seems to provide a middle ground to the trade-off of low and high freedom in each unit (Qs = 2).

4. We prefer to keep a normal (*realistic*) behavior to the users. Too polite or rude questions don't seem to work the same well with the polite neutral questions (Qp = 2, 3).

5. We prefer questions with low or medium length (Ql <= 12). Questions with a large number of words don't seem to have good results.

6. Focusing on the less free tasks, we observe that the position of the phrase that they have to provide, influences the users. Maybe it is easier to provide more useful data, when they are given both left and right context (*Qep = "center"/0*).

7. For the "Paraphrasing" task, we can't tell for sure, if it is better to paraphrase long or short phrases. Maybe the most important is the content of the phrase, however the short phrases end up with better results (Qpl <= 4).

8. Finally, for the "Free dialogues" task, dialogues that allow high freedom to the contributors have better results regarding the variety.

Comparison with Web Harvesting

Comparing the performance metrics of grammar induction and parser analysis with [31], we observe that the method used in this thesis for collecting data is inferior the method used in [31]. First, we must note that the sizes of corpora are much larger than the corpora that we created and this affects the performance of the induction algorithm, since the approach used is data-driven. Also, the method used in [31], targets to any rule of travel-flight domain, and not only to specific parts, like our method. From the grammar induction results we observe that web harvesting corpus "finds" a larger number of terminal instances than the corresponding with Crowdsourcing corpus (*399 Vs 167*) and the terminal concepts are less than the half (*26 Vs 11*). Then we ran again the grammar induction algorithm using only the subset of the grammar rules and we noticed that they are close to the results from Crowdsourcing: 356 and 14, the terminal instances and the terminal concepts, respectively. Regarding the Fmeasure, we note that the values of crowdsourcing method are comparable with web harvesting and in some cases higher, and a possible reason that this happens is due to the design of the tasks. Regarding the parser analysis, the crowdsourcing values are low compared with the web harvesting, and the reasons are the sizes of the corpora, the grammar parts that the two methods target and the cheating behavior of some of the contributors. We believe that if a better quality control system was able to be used, the acquired data would lead to higher performance.

Quality of the acquired data

The quality of the acquired data was good enough to help us achieve our initial goal, which was the grammar induction. We showed that we can control the quality of the data according to a number of parameters and the use of various type tasks. However we collected useful data even though we did not have an automatic quality control system.

7.2 Future Work

The experience and the conclusions that we end up with provide us an advanced tool for continuing with the same area of research, and improving the work presented on this thesis.

First of all, we should continue with the analysis of the collected data by applying various filters, in an attempt to create corpora with which the best possible performance is achieved. The filters can be based on the parameters, or we can introduce new kinds of filtering, e.g. filters based on the perplexity.

Moreover, we can continue with a machine learning approach. Ideal goal of this expansion is to be able to generate sentences, with semi-supervised or unsupervised way, that lead to the collection of useful data. For the completion of this goal, maybe some thousands of sentences, as well as a larger range of grammar domains (*except from Date, Departure-city and top-level prompts*) are necessary. This implies that more Crowdsourcing experiments must take place. Also, it is important to find an effective way of ensuring quality control in the next crowdsourcing experiments. By this way we collect more useful data spending less money.

Since we have reached accurate results about the parameter values that lead to useful data, we can design an interface for generation of sentences for the Crowdsourcing tasks. However, it is not easy to make an unsupervised approach since we need to design sentences with the appropriate concept.

A future extension is to create Crowdsourcing tasks that target to other grammar domains of travel - flight domain, e.g. time, booking tickets or arrival city. In this case we can use the same Crowdsourcing tasks, and apply the, based on parameters, methodology that we introduced in this thesis. Also, the generation of the sentences for each task can be done with semi - supervised way. In essence, we can take advance of the conclusions we reached about the parameter values, and after the design of the basic sentences (*based on concept as we described in appendix A*) we can forward with an automatic generation of the rest of the sentences.

One more expansion is the use of crowdsourcing for collecting data in other languages. In this case, it is possible to create a design much different from the design that we used in this thesis, since every natural language has different characteristics.

Appendix A

Initial Questions designed per task

In this appendix we show the first questions that we designed for each task. In each table we have delimiters for distinguishing the questions that target to Date, Departure - city or prompts. We tried to create questions that lead contributors to provide answers that belong to the above domains, allowing enough freedom. The process of generating the questions is explained with the following example from "Answers" task where

- Qs : Question specificity.
- Qp : Question politeness.
- Ql : Question length in words.
- Noc : Number of concepts expressed in a sentence.

	Qs	Qp	Ql	Noc
Initial Sentence: "When do you want to depart?"	1	2	6	1
Sentences generated by changing Qs				
"What date do you want to depart?"	2	2	7	1
"Tell me details regarding your departure	0	2	6	1
Sentence generated by changing Noc				
"What date and time do you want to depart?"	2	2	9	2
Sentence generated by changing Qp				
"When do you want to depart please?"	1	5	7	1

Note that: Ql changes with the change of other parameters, Qs is the parameter with the most changes, because it defines the freedom of the question.

No	Sentence for "Answers" task	For
1	When do you want to depart?	Date
2	Hello, how may I help you?	
3	This is AirTravel System and is a pleasure to serve you. When does your travel begin?	
4	Could you please inform me about the departing date?	
5	When do you want to arrive?	
6	Give me the date.	
7	Have you got any preference regarding the return day?	
8	Excuse me, can you repeat the date?	
9	From where does your travel begin?	Departure - city
10	Give me the starting location first, please.	
11	Can you tell me details about your departure?	
12	Where would you like to fly to and from?	
13	From where does your travel begin?	
14	Are there any preferences regarding the airport you want to leave from?	
15	I cannot tell you if there is a flight to London unless you tell me your departure place.	

Table A.1: Initial Questions for the"Answers" task

No	Sentence for "Prompts" task	For
1	I want to book a flight.	Prompts
2	On May two thousand thirteen.	
3	I want to travel from Rome to London on Monday.	
4	I would like you to give me some information about afternoon flights.	
5	I need to travel in the next two hours.	
6	Paris.	
7	From Chicago to Mexico city, on November twenty first, in the afternoon.	
8	Hello, is there an available flight to Madrid today?	
9	Good evening, can you find me the cheapest ticket to Bangladesh?	
10	I need to change the ticket that I booked yesterday.	
11	Oops, sorry, I got the wrong number!	
12	Yes, I want to know the available flights to Tokyo.	
13	I need to book tickets for a group of fifteen people.	
14	I don't want to travel, I just want to ask some questions.	
15	From Athens to Istanbul	

Table A.2: Initial Questions for the "Prompts" task

No	Sentence for "Paraphrasing" task	For
1	Hello, this is AirTravel System.How may I help you?	Prompts
2	Welcome to AirTravel System. Please give me your trip information.	
3	Good morning! Where do you want to travel?	
4	You have called AirTravel System. How can we serve you?	
5	I would like to depart on November second.	Date
6	I need to leave on December thirty first.	
7	I want to be back on Wednesday the twenty fifth of February.	
8	I prefer at the beginning of the next week.	
9	I want to travel at the end of two thousand thirteen.	
10	I have to be back on Monday night.	
11	Middle of summer could be the best.	
12	I want to begin between Friday and Sunday from Cape Town	
13	Starting location is Athens, Greece.	Departure-city
14	Last flight is leaving from Puerto Rico in an hour.	
15	Please tell me the start location.	
16	I want to depart from Boston to Sidney.	
17	We prefer to take off from Orly Airport.	
18	I would like to depart from Hawaii after two weeks.	
19	I have no preference on destination airport, but I want to depart from Athens airport .	
20	Departing city is Tokyo and destination is Ottawa.	

Table A.3: Initial Questions for the "Paraphrasing" task

No	**Sentence for “Fill in” task**	For
1	Hello from AirTravel system.$\boxed{\qquad\quad}$?	Prompts
2	Welcome to AirTravel System.How$\boxed{\qquad\quad}$?	
3	Give me $\boxed{\qquad\quad}$.	
4	Hello, $\boxed{\qquad\quad}$ details about your trip?	
5	Welcome to AirTravel System, what$\boxed{\qquad\quad}$?	
6	I want to leave on $\boxed{\qquad\quad}$.	Date
7	I want to arrive $\boxed{\qquad\quad}$.	
8	I want to be back before $\boxed{\qquad\quad}$ in the morning.	
9	$\boxed{\qquad\quad}$ is the perfect date.	
10	I prefer $\boxed{\qquad\quad}$ instead of weekend.	
11	I must be there $\boxed{\qquad\quad}$ before night.	
12	My plan is to come back $\boxed{\qquad\quad}$.	
13	I want to begin $\boxed{\qquad\quad}$and return after two months.	
14	I need to change the existing date of my flight to $\boxed{\qquad\quad}$.	
15	Neither of these dates fits me. Is there an available flight $\boxed{\qquad\quad}$?	
16	I would like to $\boxed{\qquad\quad}$ New York.	Departure-city
17	$\boxed{\qquad\quad}$ is Dubai.	
18	Hello, I would like $\boxed{\qquad\quad}$ to Panama.	
19	I $\boxed{\qquad\quad}$ from Milan, Italy.	
20	I want to $\boxed{\qquad\quad}$ London to Athens.	
21	I have to depart $\boxed{\qquad\quad}$ before Monday night.	
22	I want to $\boxed{\qquad\quad}$ and return after two days.	
23	I depart $\boxed{\qquad\quad}$ today.	
24	$\boxed{\qquad\quad}$ is London and destination is Paris.	
25	I want $\boxed{\qquad\quad}$ from Oslo.	

Table A.4: Initial Questions for the “Fill in” task

No	Sentence for "Free dialogues" task	
1	System :	
	User :	On Friday January the third.
	System :	
	User :	
	System:	
	User:	I prefer the first.
2	System :	
	User :	
	System :	
	User :	Is this the only available date?
	System:	
	User:	
3	System :	
	User :	Yes, are you there?
	System :	
	User :	
	System:	
	User :	Boston.
4	System :	
	User :	
	System :	
	User :	
	System:	
	User:	I will take the first.

Table A.5: Initial Questions for the "Free dialogues" task (a)

No	**Sentence for “Free dialogues” task**	
5	System :	
	User :	I want to change the date of my flight.
	System :	
	User :	My flight number is three five double zero six.
	System:	
	User:	
6	System :	
	User :	
	System :	
	User :	
	System:	
	User:	Any airport In Paris.
7	System :	What is your starting point?
	User :	
	System :	And what date do you prefer?
	User :	
	System:	
	User:	
8	System :	This is AirTravel System. How may I help you?
	User :	
	System :	One moment please...Yes, there is.
	User :	
	System:	Please give an alternative.
	User:	

Table A.6: Initial Questions for the “Free dialogues” task(b)

Appendix B

Distribution of the Parameters

In this appendix we depict the distribution of the parameters that we designed in each task. Remember that:

· Qs : Question specificity.

· Qp : Question politeness.

· Ql : Question length in words.

· Noc : Number of concepts expressed in a sentence.

· Qpl : Question paraphrase's length.

  – It is defined only for the "Paraphrasing" task.

· Qep : Question's empty field position.

  – It is defined only for the "Paraphrasing" and "Fill in" task.

· Df: Dialogue's freedom.

  – It is defined only for the "Free Dialogues" task.

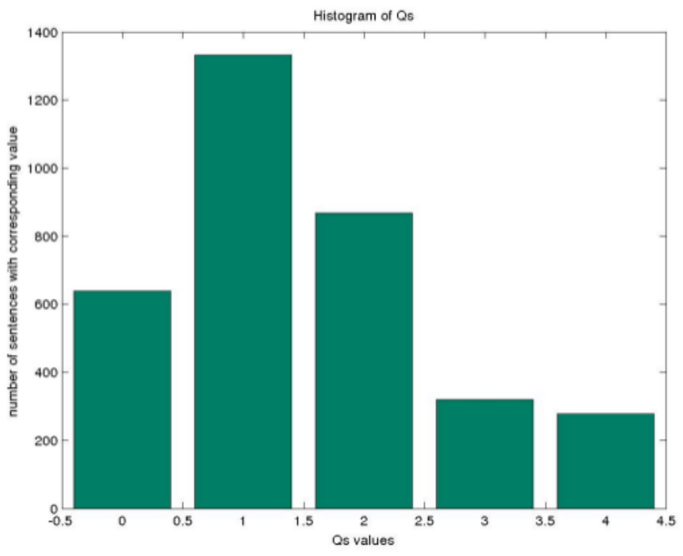


Figure B.1: Qs distribution for the“Answers” task

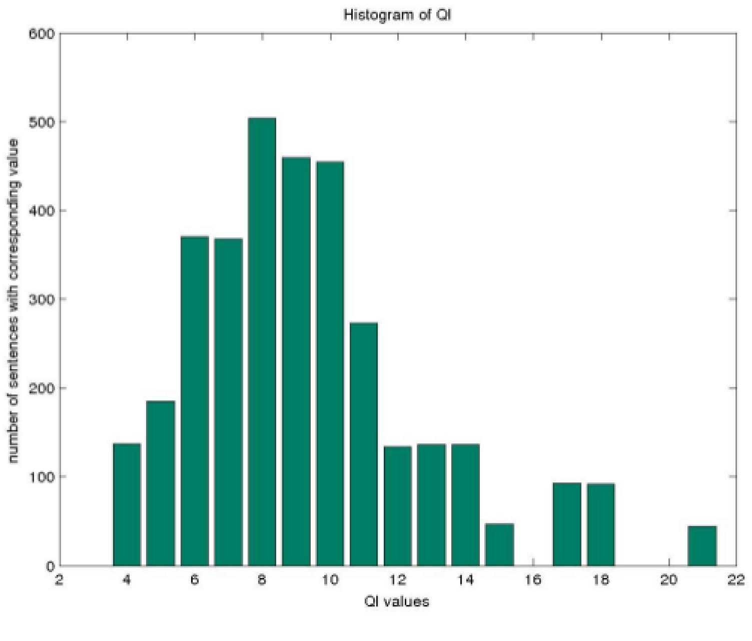


Figure B.2: Ql distribution for the “Answers” task

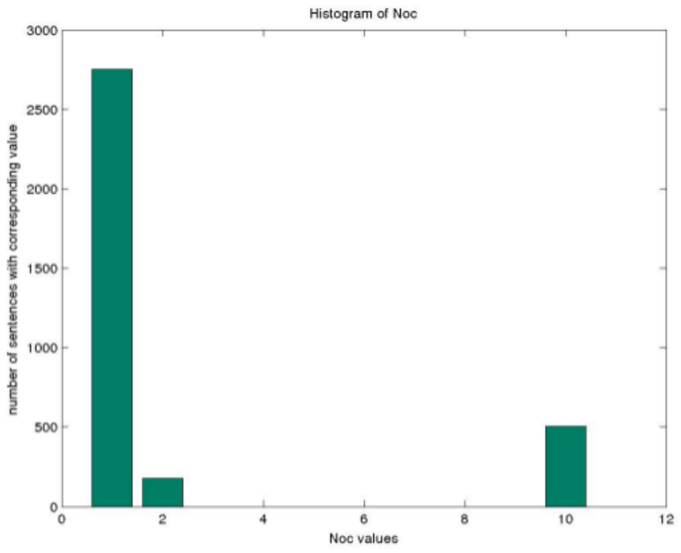


Figure B.3: Noc distribution for the “Answers” task

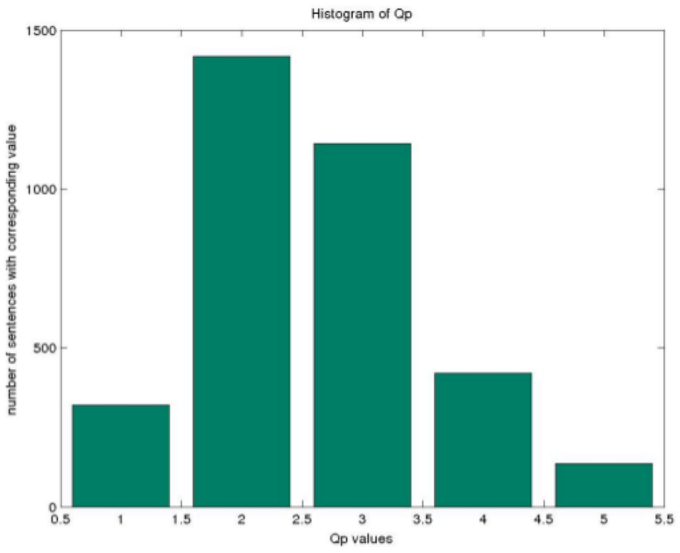


Figure B.4: Qp distribution for the “Answers” task

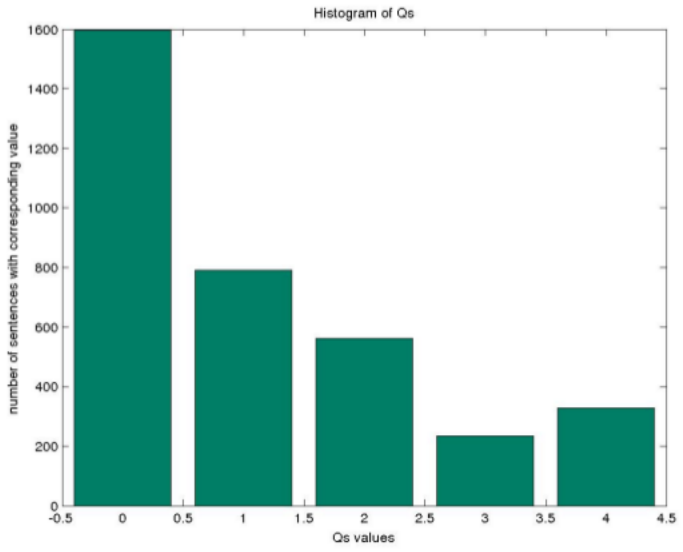


Figure B.5: Qs distribution for the “Prompts” task

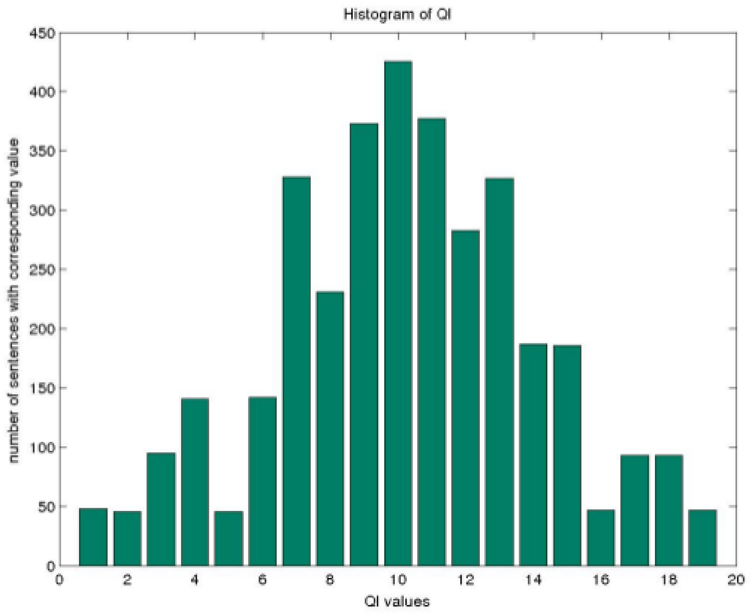


Figure B.6: Ql distribution for the “Prompts” task

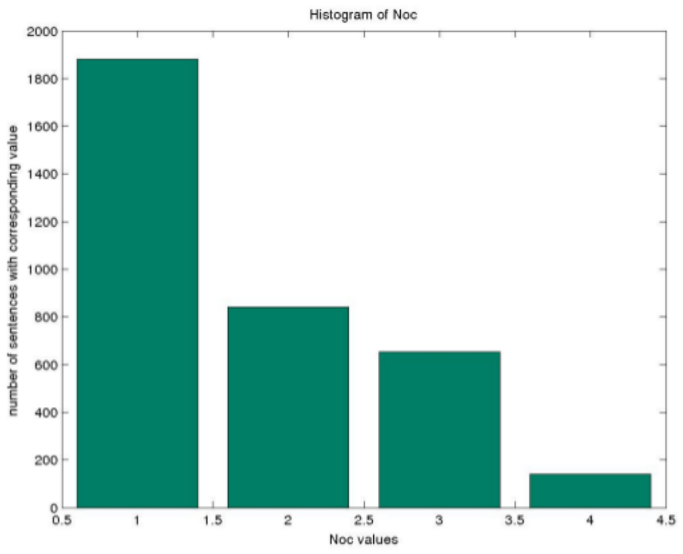


Figure B.7: Noc distribution for “Prompts” task

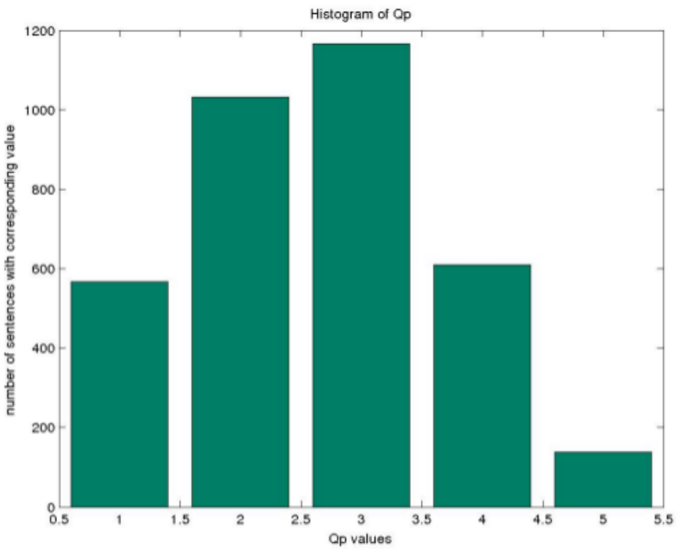


Figure B.8: Qp distribution for the “Prompts” task

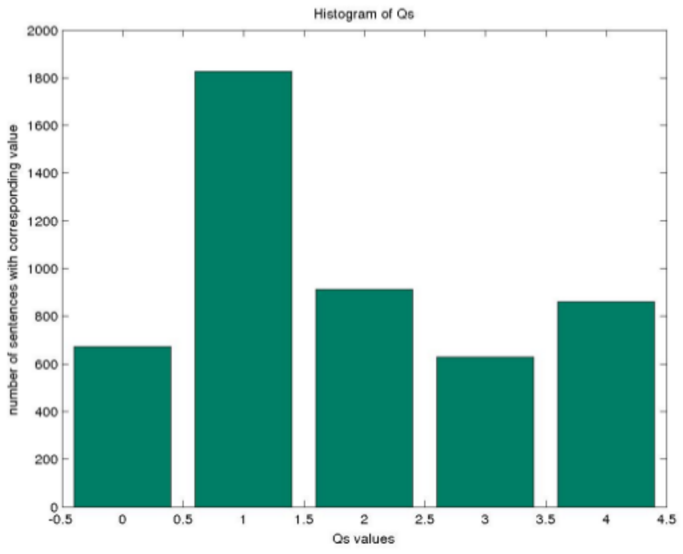


Figure B.9: Qs distribution for the “Paraphrasing” task

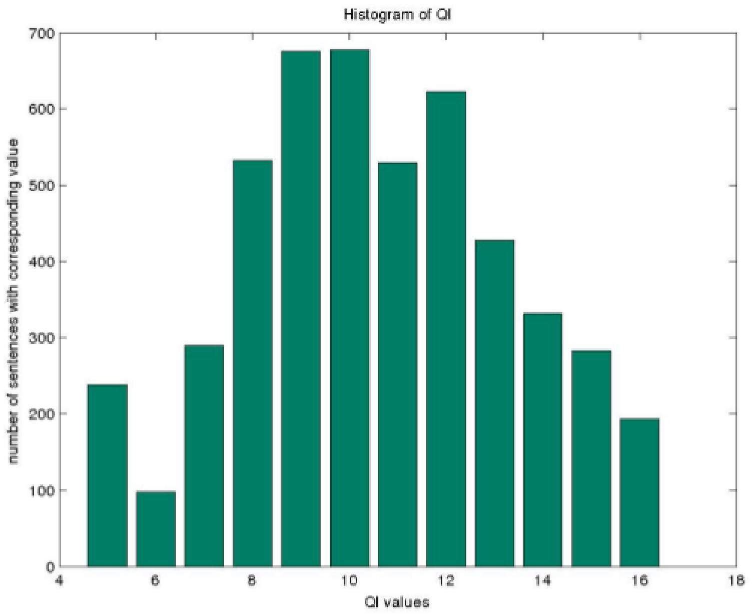


Figure B.10: Ql distribution for the “Paraphrasing” task

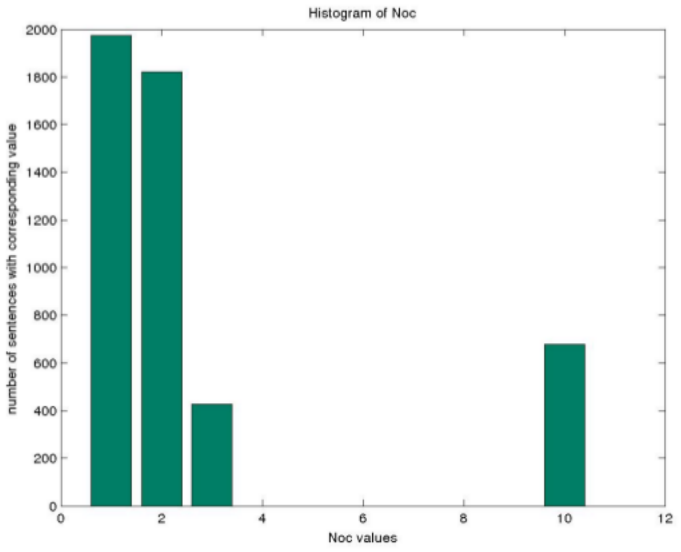


Figure B.11: Noc distribution for the “Paraphrasing” task

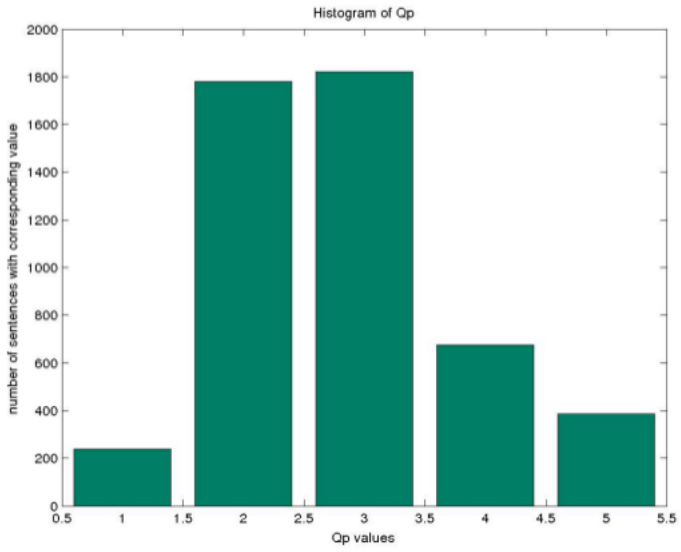


Figure B.12: Qp distribution for the “Paraphrasing” task

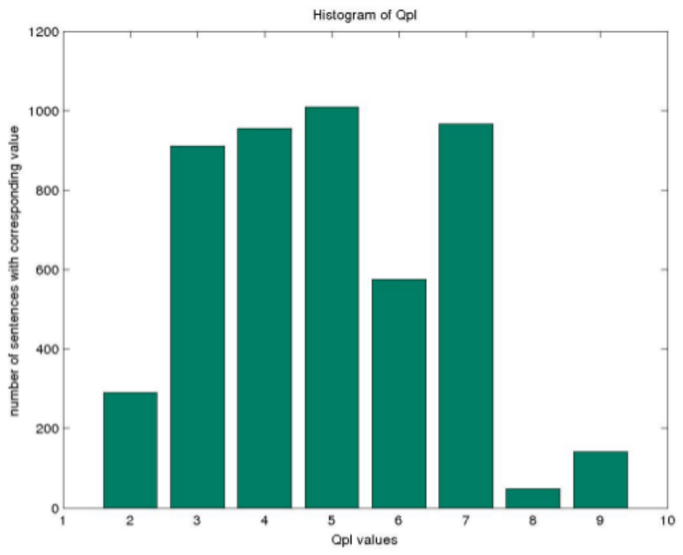


Figure B.13: Qpl distribution for the “Paraphrasing” task

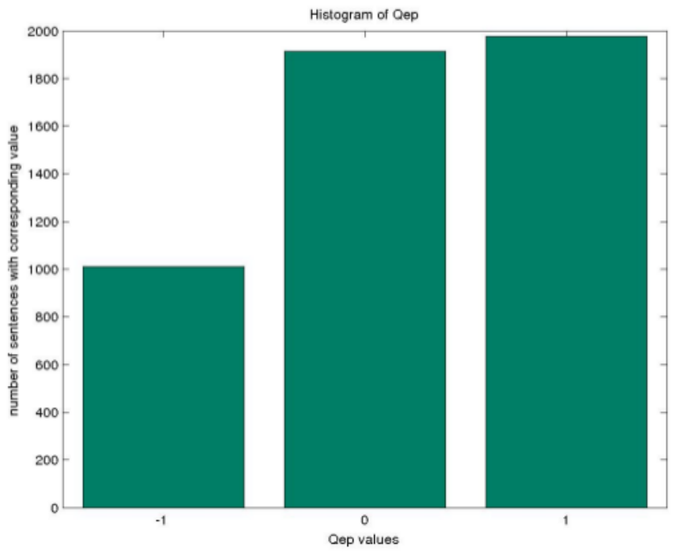


Figure B.14: Qep distribution for the “Paraphrasing” task

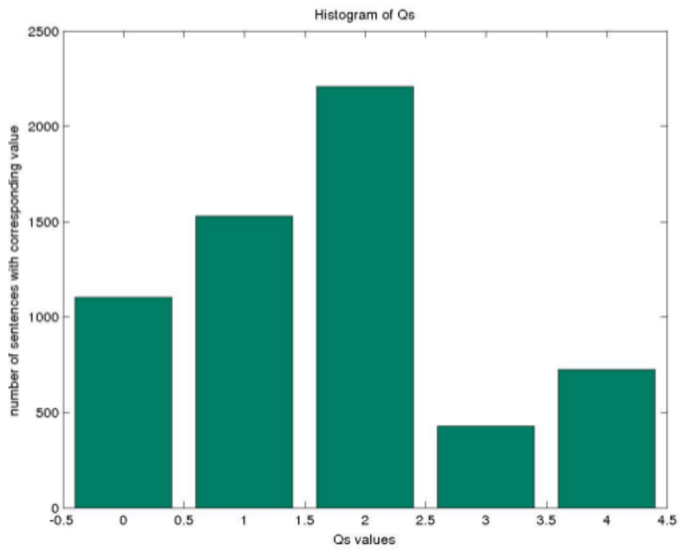


Figure B.15: Qs distribution for the “Fill in” task

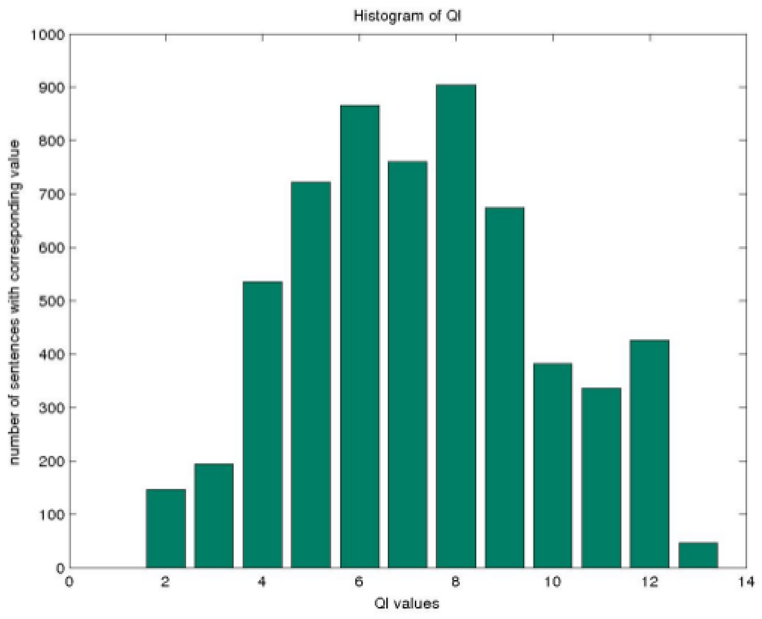


Figure B.16: Ql distribution for the “Fill in” task

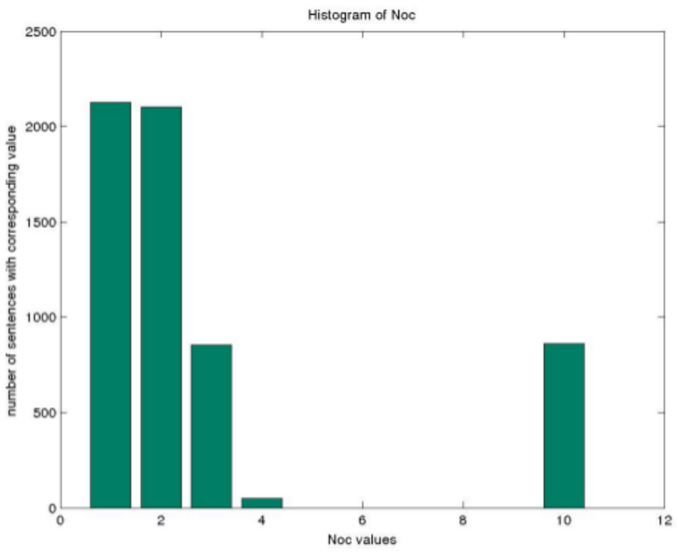


Figure B.17: Noc distribution for the “Fill in” task

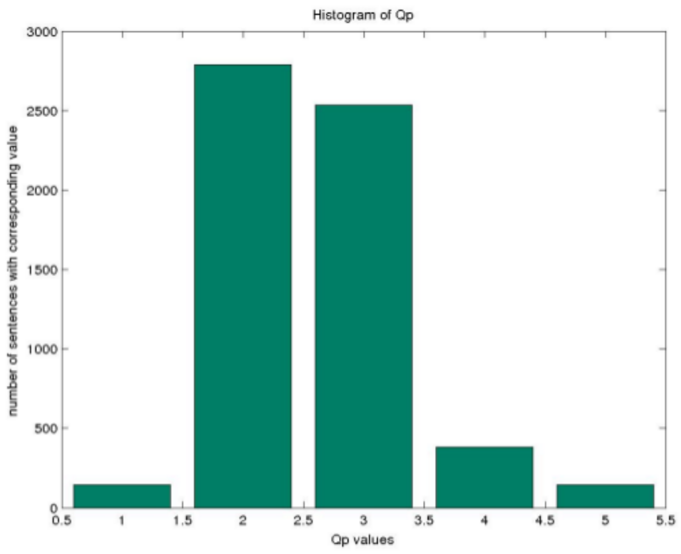


Figure B.18: Qp distribution for the “Fill in” task

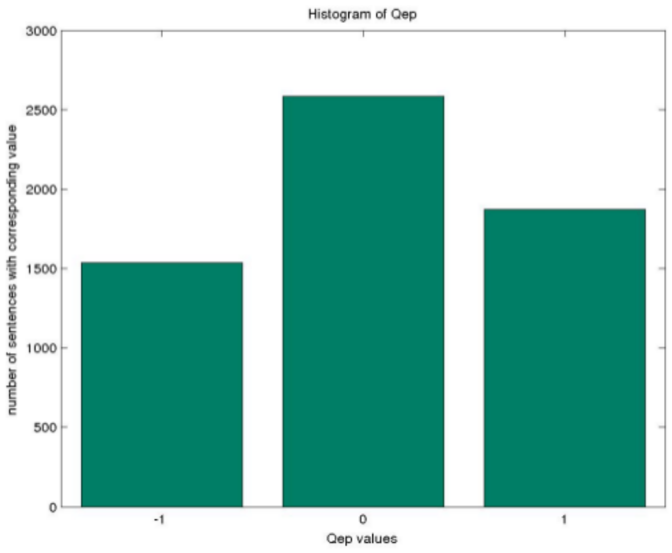


Figure B.19: Qep distribution for the “Fill in” task

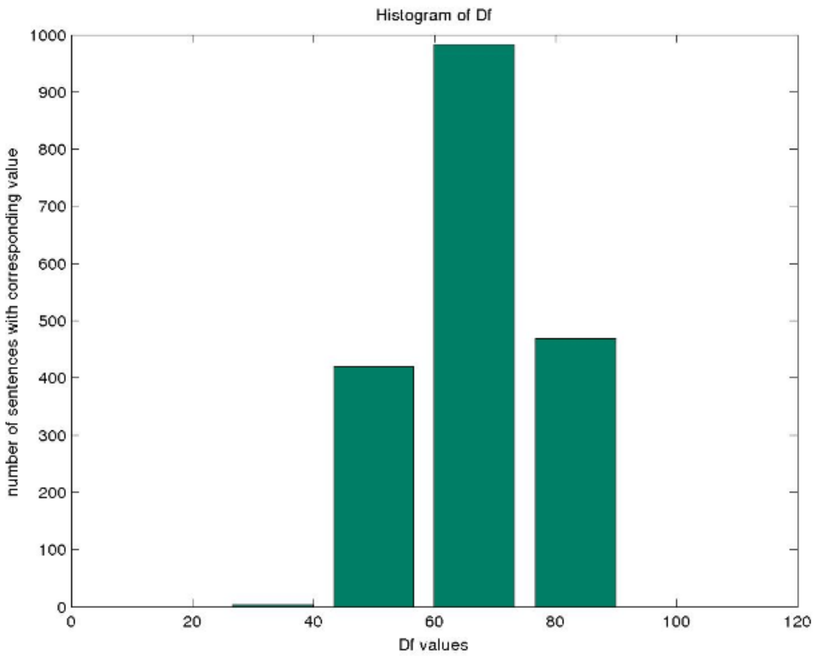


Figure B.20: Df distribution for the “Free dialogues” task

Appendix C

Crowdflower and UI

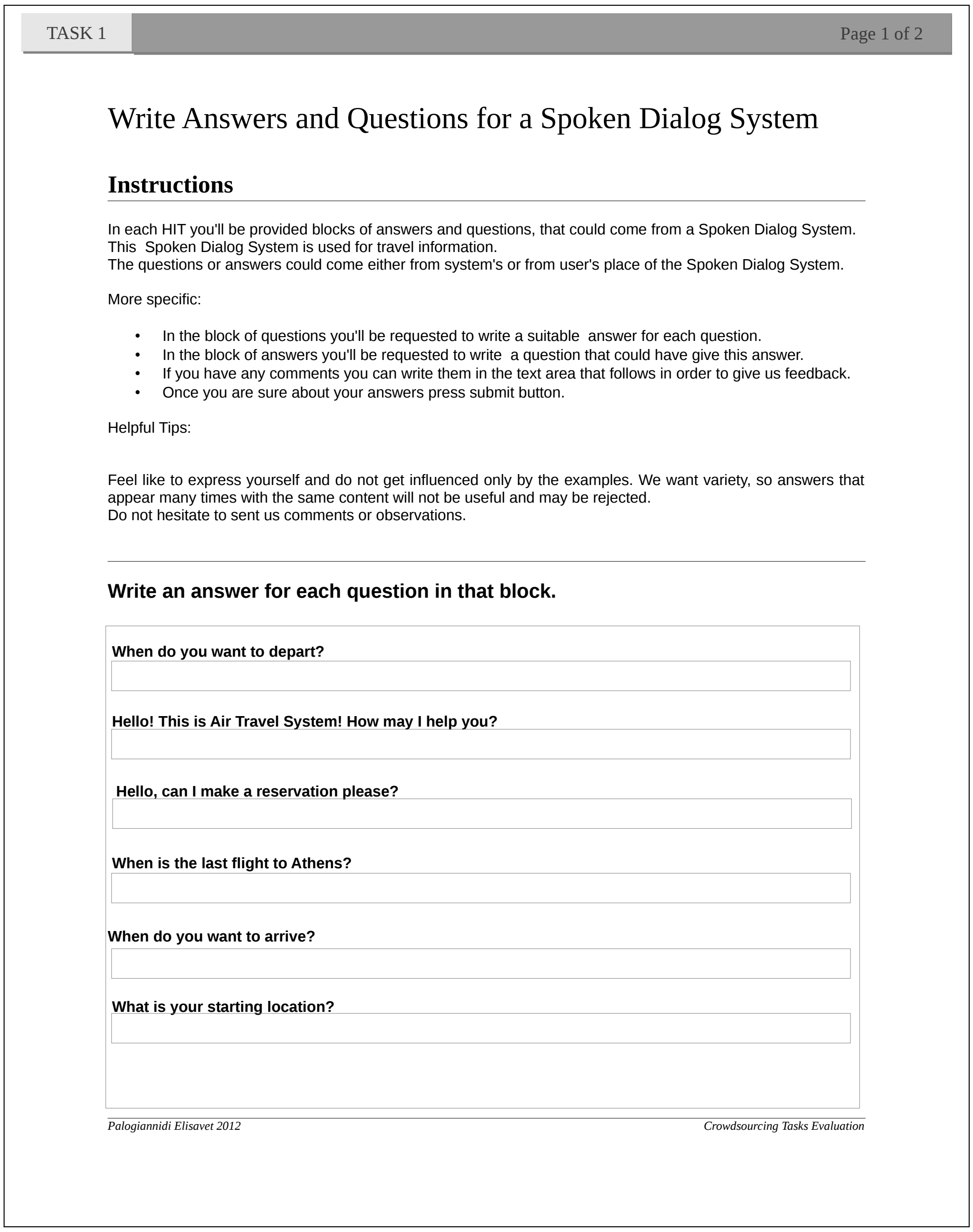
Goal of this appendix is to show how the UI of the tasks was created. First the UI was designed in LibreOffice Writer for the needs of the pilot process. Then, after the feedback we got from the users, we continued with the on-line design on Crowdflower. The design on Crowdflower was done with CML (*Crowdflower Markup Language*) and CSS when it is necessary. CML uses numerous tags, that make us handle the information that we upload on Crowdflower easier. In addition, a graphical tool is also available, but without allowing extra functionality except the graphical representation.

Next step, is the upload of the questions we designed, which is done through excel files. Each column of the excel, corresponds to a separate field of the UI. For example, in the "Answers" task, we have an excel with only one column, where we store the questions, but in the "Fill in" task we have two columns, one for the phrase left from the missing part, and one for the phrase right from the missing part. We continue similarly, with the rest tasks. Finally, the data we collect, were also returned to us stored in excel files, including extra information about the contributors, such us their city, the crowdsourcing channel, through which they completed the task, IP address e.t.c..

Next step was the selection of the titles of the tasks and the UI of pilot process as well as Crowdsourcing experiment. Note that in the beginning "Answers" and "Prompts" were one task.

Task	**Title of Crowdsourcing task**
Answers	Answer The Questions
Prompts	Provide Questions For The Given Responses
Paraphrasing	Provide Phrases With The Same Meaning
Free Dialogs	Complete The Dialogs
Fill in	Fill In The Empty Fields With The Most Appropriate Word(s)

Table C.1: Crowdsourcing tasks's titles



TASK 1 Page 1 of 2

Write Answers and Questions for a Spoken Dialog System

Instructions

In each HIT you'll be provided blocks of answers and questions, that could come from a Spoken Dialog System. This Spoken Dialog System is used for travel information.
The questions or answers could come either from system's or from user's place of the Spoken Dialog System.

More specific:

- In the block of questions you'll be requested to write a suitable answer for each question.
- In the block of answers you'll be requested to write a question that could have give this answer.
- If you have any comments you can write them in the text area that follows in order to give us feedback.
- Once you are sure about your answers press submit button.

Helpful Tips:

Feel like to express yourself and do not get influenced only by the examples. We want variety, so answers that appear many times with the same content will not be useful and may be rejected.
Do not hesitate to sent us comments or observations.

Write an answer for each question in that block.

When do you want to depart?

Hello! This is Air Travel System! How may I help you?

Hello, can I make a reservation please?

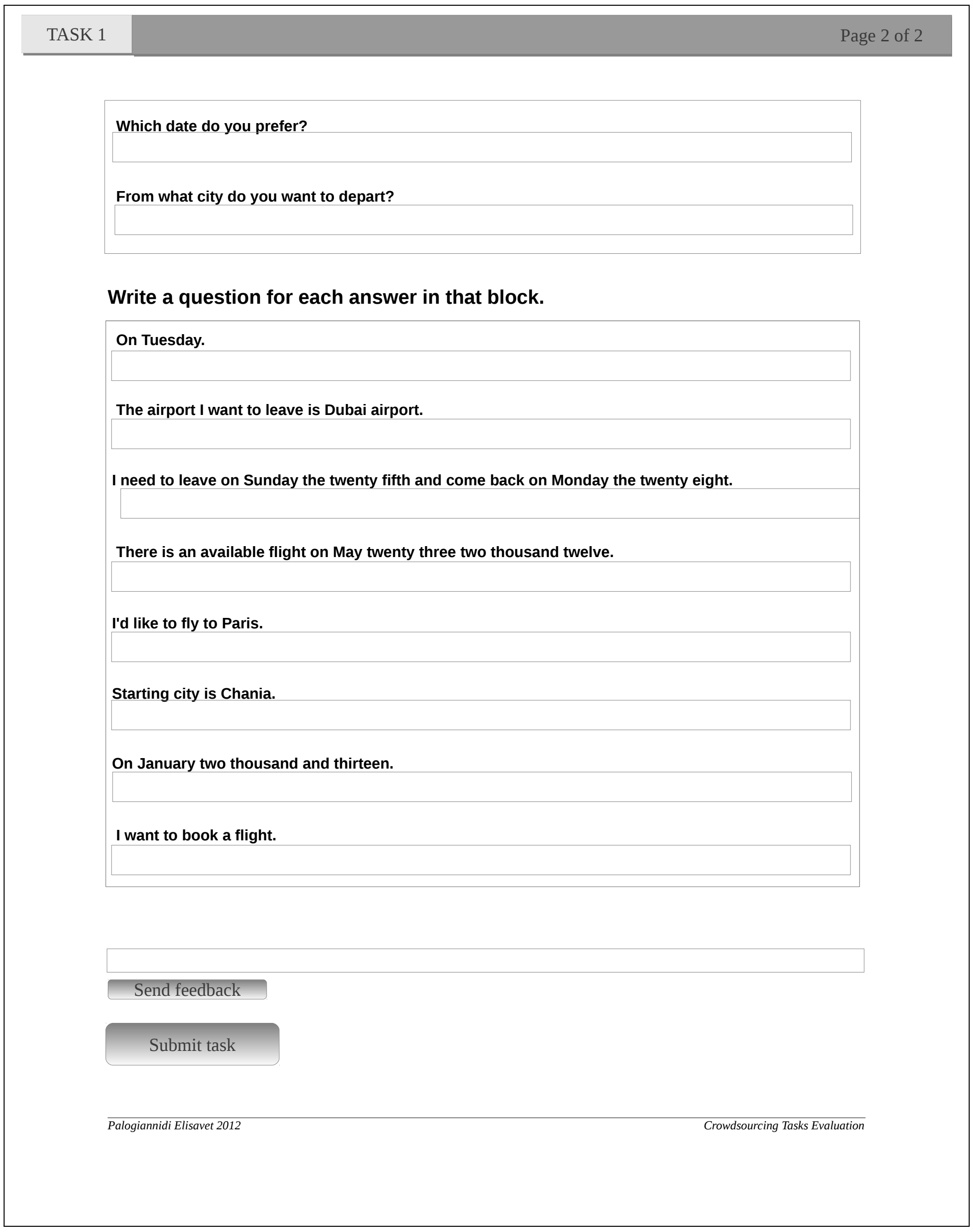
When is the last flight to Athens?

When do you want to arrive?

What is your starting location?

Palogiannidi Elisavet 2012 *Crowdsourcing Tasks Evaluation*

Figure C.1: “Answers” and “Prompts” task on pilot (a)



Which date do you prefer?

From what city do you want to depart?

Write a question for each answer in that block.

On Tuesday.

The airport I want to leave is Dubai airport.

I need to leave on Sunday the twenty fifth and come back on Monday the twenty eight.

There is an available flight on May twenty three two thousand twelve.

I'd like to fly to Paris.

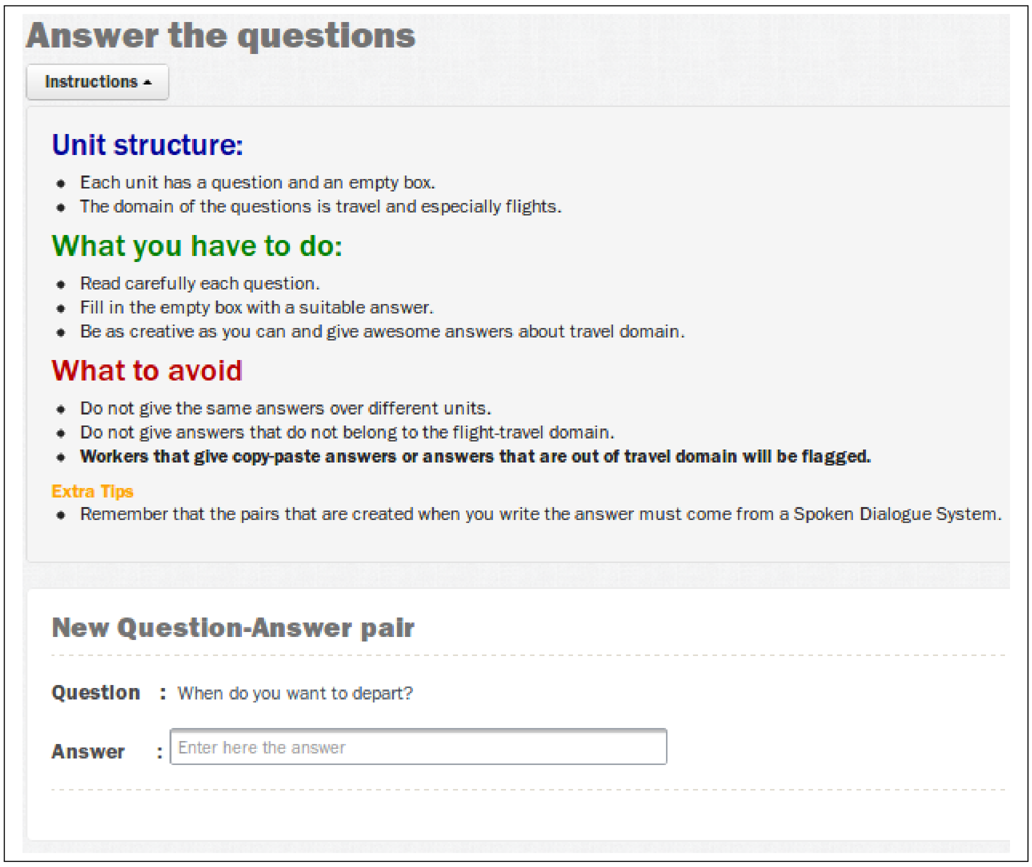
Starting city is Chania.

On January two thousand and thirteen.

I want to book a flight.

Palogiannidi Elisavet 2012 Crowdsourcing Tasks Evaluation

Figure C.2: “Answers” and “Prompts” task on pilot (b)



Answer the questions

Unit structure:

- Each unit has a question and an empty box.
- The domain of the questions is travel and especially flights.

What you have to do:

- Read carefully each question.
- Fill in the empty box with a suitable answer.
- Be as creative as you can and give awesome answers about travel domain.

What to avoid

- Do not give the same answers over different units.
- Do not give answers that do not belong to the flight-travel domain.
- **Workers that give copy-paste answers or answers that are out of travel domain will be flagged.**

Extra Tips

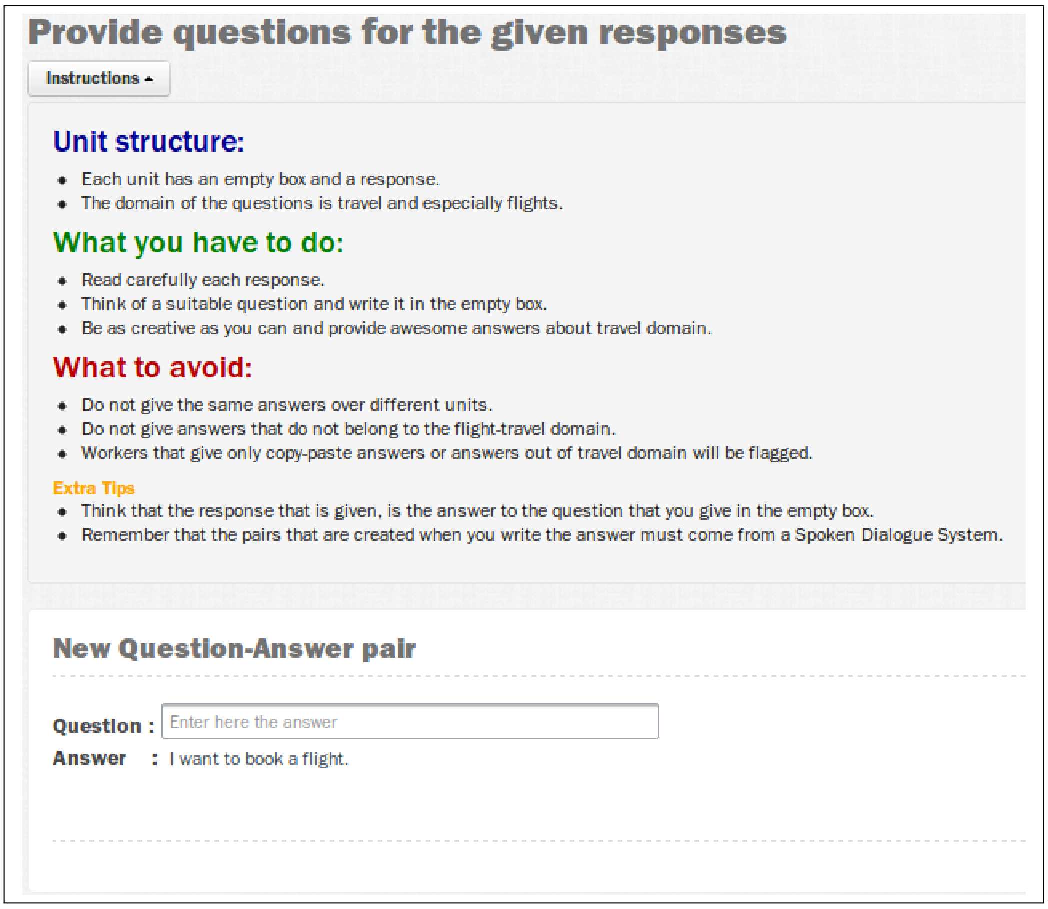
- Remember that the pairs that are created when you write the answer must come from a Spoken Dialogue System.

New Question-Answer pair

Question : When do you want to depart?

Answer : Enter here the answer

Figure C.3: "Answers" task on Crowdflower



Provide questions for the given responses

Unit structure:

- Each unit has an empty box and a response.
- The domain of the questions is travel and especially flights.

What you have to do:

- Read carefully each response.
- Think of a suitable question and write it in the empty box.
- Be as creative as you can and provide awesome answers about travel domain.

What to avoid:

- Do not give the same answers over different units.
- Do not give answers that do not belong to the flight-travel domain.
- Workers that give only copy-paste answers or answers out of travel domain will be flagged.

Extra Tips

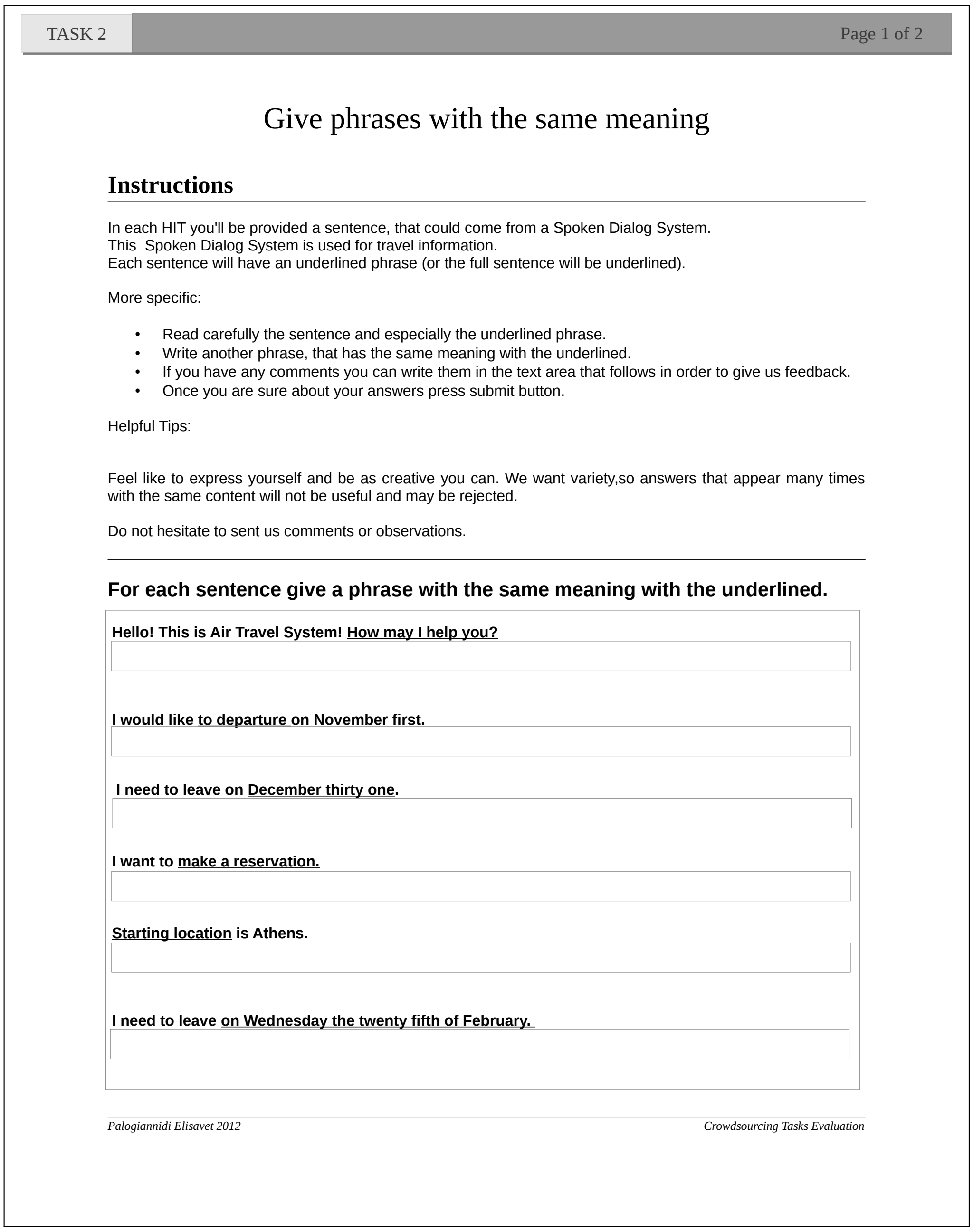
- Think that the response that is given, is the answer to the question that you give in the empty box.
- Remember that the pairs that are created when you write the answer must come from a Spoken Dialogue System.

New Question-Answer pair

Question : Enter here the answer

Answer : I want to book a flight.

Figure C.4: “Prompts” task on Crowdflower



TASK 2 Page 1 of 2

Give phrases with the same meaning

Instructions

In each HIT you'll be provided a sentence, that could come from a Spoken Dialog System.
This Spoken Dialog System is used for travel information.
Each sentence will have an underlined phrase (or the full sentence will be underlined).

More specific:

- Read carefully the sentence and especially the underlined phrase.
- Write another phrase, that has the same meaning with the underlined.
- If you have any comments you can write them in the text area that follows in order to give us feedback.
- Once you are sure about your answers press submit button.

Helpful Tips:

Feel like to express yourself and be as creative you can. We want variety,so answers that appear many times with the same content will not be useful and may be rejected.

Do not hesitate to sent us comments or observations.

For each sentence give a phrase with the same meaning with the underlined.

Hello! This is Air Travel System! How may I help you?

I would like to departure on November first.

I need to leave on December thirty one.

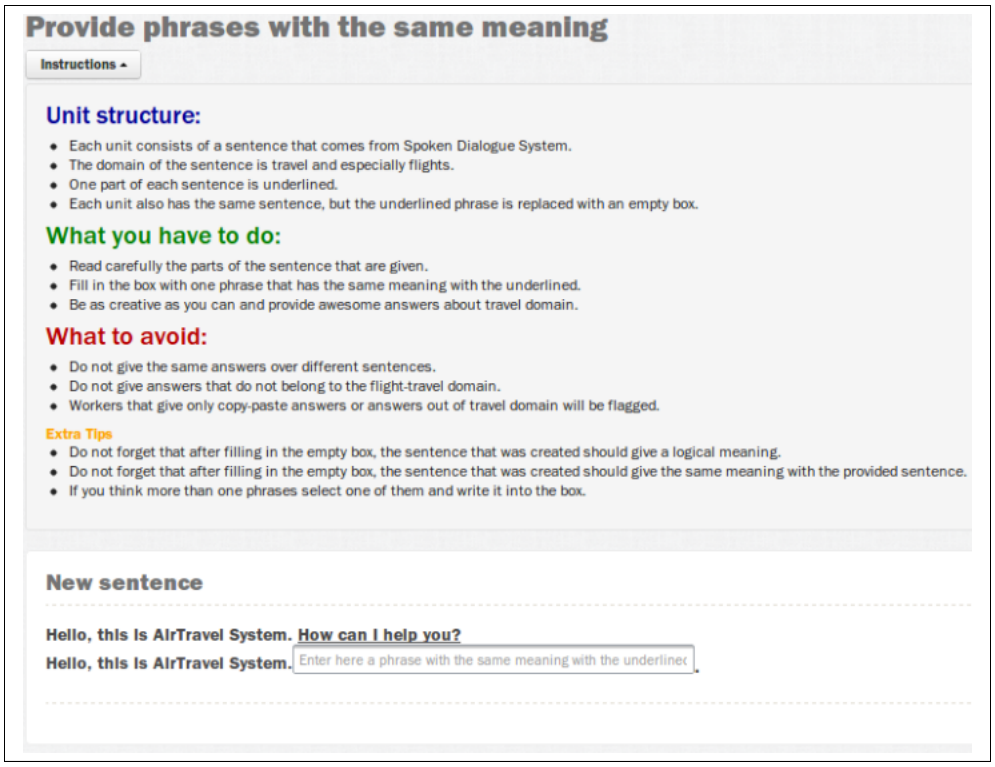
I want to make a reservation.

Starting location is Athens.

I need to leave on Wednesday the twenty fifth of February.

Palogiannidi Elisavet 2012 *Crowdsourcing Tasks Evaluation*

Figure C.5: “Paraphrasing” task on pilot



Provide phrases with the same meaning

Unit structure:

- Each unit consists of a sentence that comes from Spoken Dialogue System.
- The domain of the sentence is travel and especially flights.
- One part of each sentence is underlined.
- Each unit also has the same sentence, but the underlined phrase is replaced with an empty box.

What you have to do:

- Read carefully the parts of the sentence that are given.
- Fill in the box with one phrase that has the same meaning with the underlined.
- Be as creative as you can and provide awesome answers about travel domain.

What to avoid:

- Do not give the same answers over different sentences.
- Do not give answers that do not belong to the flight-travel domain.
- Workers that give only copy-paste answers or answers out of travel domain will be flagged.

Extra Tips

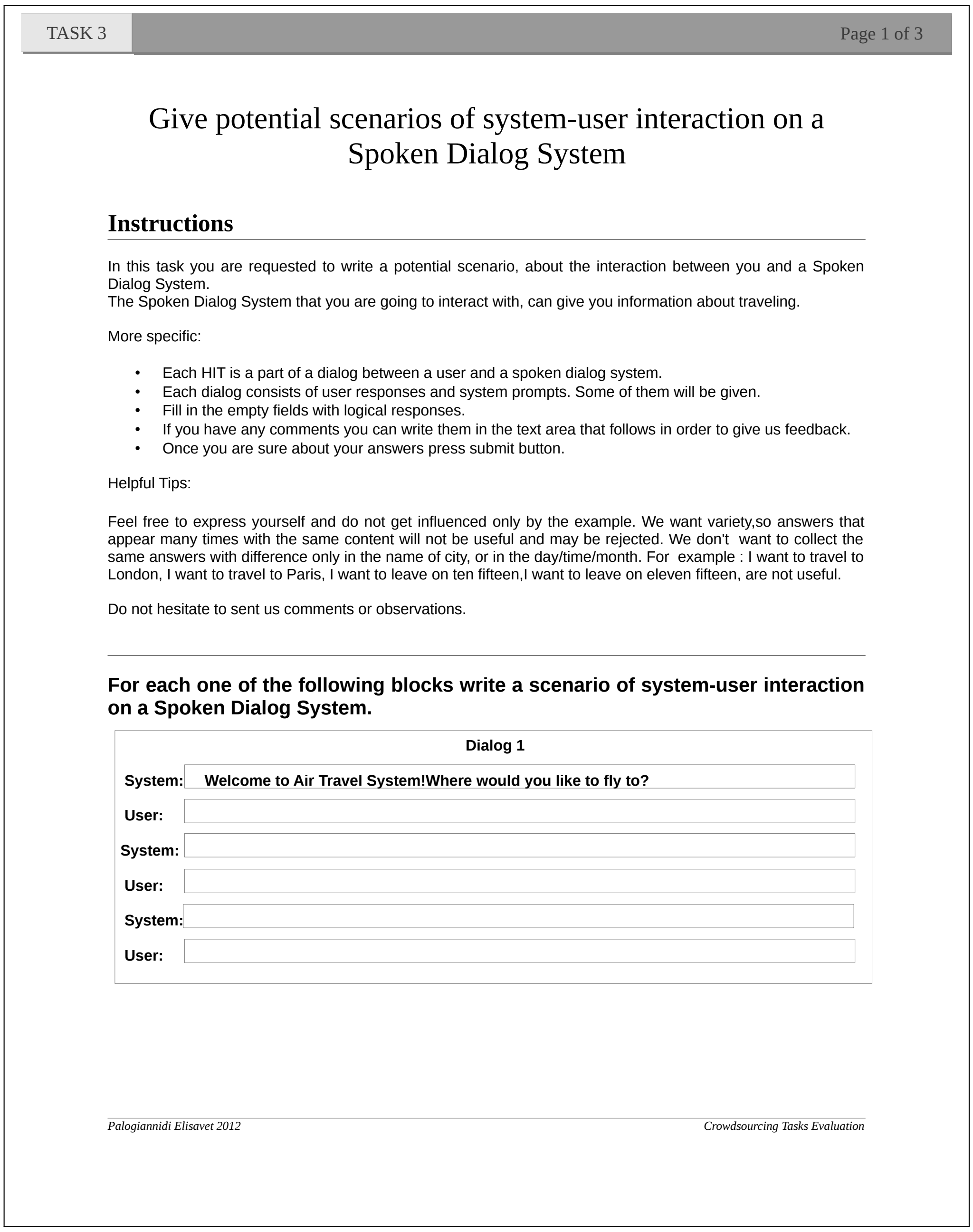
- Do not forget that after filling in the empty box, the sentence that was created should give a logical meaning.
- Do not forget that after filling in the empty box, the sentence that was created should give the same meaning with the provided sentence.
- If you think more than one phrases select one of them and write it into the box.

New sentence

Hello, this is AirTravel System. How can I help you?

Hello, this is AirTravel System. Enter here a phrase with the same meaning with the underlinec.

Figure C.6: "Paraphrasing" task on Crowdflower



TASK 3 Page 1 of 3

Give potential scenarios of system-user interaction on a Spoken Dialog System

Instructions

In this task you are requested to write a potential scenario, about the interaction between you and a Spoken Dialog System.
The Spoken Dialog System that you are going to interact with, can give you information about traveling.

More specific:

- Each HIT is a part of a dialog between a user and a spoken dialog system.
- Each dialog consists of user responses and system prompts. Some of them will be given.
- Fill in the empty fields with logical responses.
- If you have any comments you can write them in the text area that follows in order to give us feedback.
- Once you are sure about your answers press submit button.

Helpful Tips:

Feel free to express yourself and do not get influenced only by the example. We want variety,so answers that appear many times with the same content will not be useful and may be rejected. We don't want to collect the same answers with difference only in the name of city, or in the day/time/month. For example : I want to travel to London, I want to travel to Paris, I want to leave on ten fifteen,I want to leave on eleven fifteen, are not useful.

Do not hesitate to sent us comments or observations.

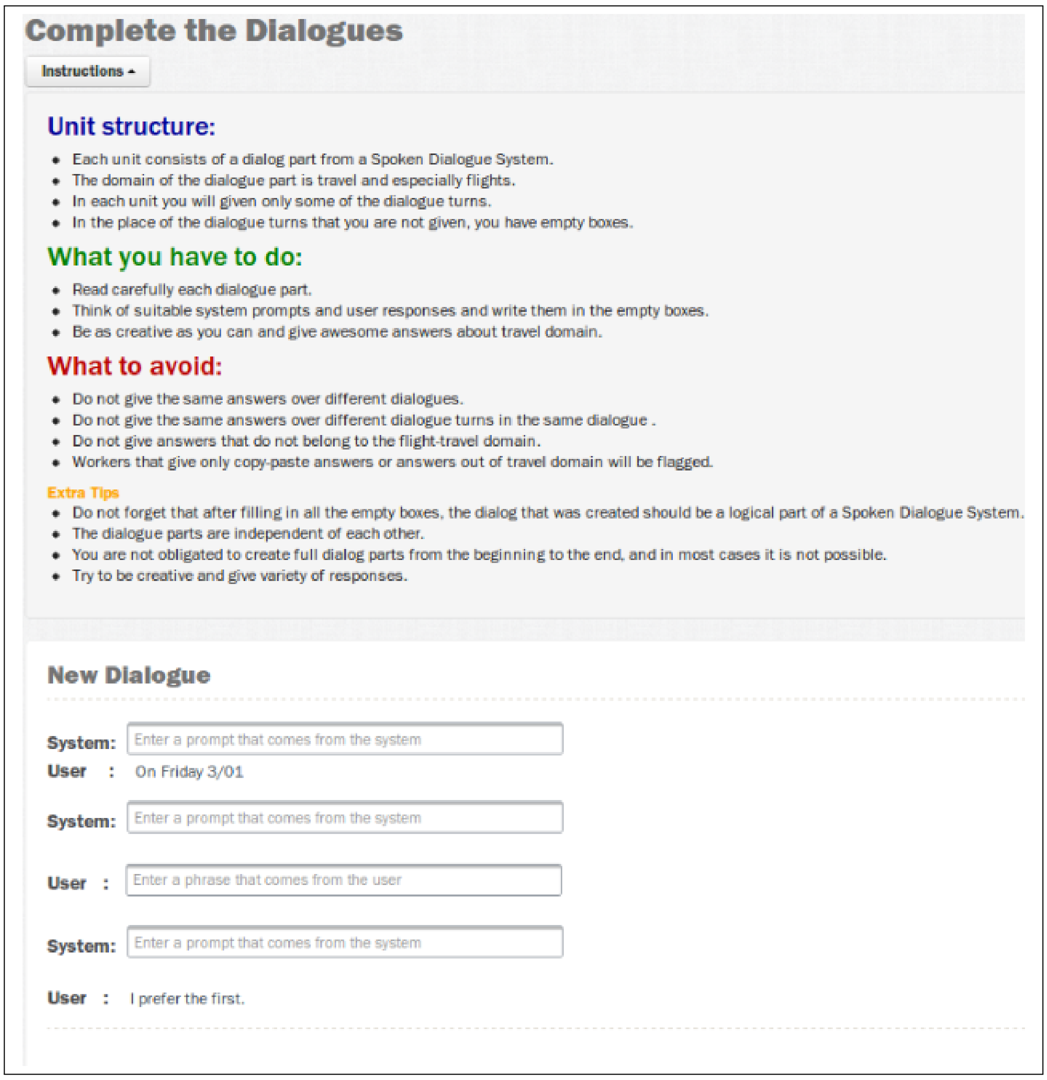
For each one of the following blocks write a scenario of system-user interaction on a Spoken Dialog System.

Dialog 1

System:	**Welcome to Air Travel System!Where would you like to fly to?**
User:	
System:	
User:	
System:	
User:	

Palogiannidi Elisavet 2012 *Crowdsourcing Tasks Evaluation*

Figure C.7: “Free dialogue” task on pilot


Complete the Dialogues

Instructions ▴

Unit structure:

- Each unit consists of a dialog part from a Spoken Dialogue System.
- The domain of the dialogue part is travel and especially flights.
- In each unit you will given only some of the dialogue turns.
- In the place of the dialogue turns that you are not given, you have empty boxes.

What you have to do:

- Read carefully each dialogue part.
- Think of suitable system prompts and user responses and write them in the empty boxes.
- Be as creative as you can and give awesome answers about travel domain.

What to avoid:

- Do not give the same answers over different dialogues.
- Do not give the same answers over different dialogue turns in the same dialogue .
- Do not give answers that do not belong to the flight-travel domain.
- Workers that give only copy-paste answers or answers out of travel domain will be flagged.

Extra Tips

- Do not forget that after filling in all the empty boxes, the dialog that was created should be a logical part of a Spoken Dialogue System.
- The dialogue parts are independent of each other.
- You are not obligated to create full dialog parts from the beginning to the end, and in most cases it is not possible.
- Try to be creative and give variety of responses.

New Dialogue

System: Enter a prompt that comes from the system

User : On Friday 3/01

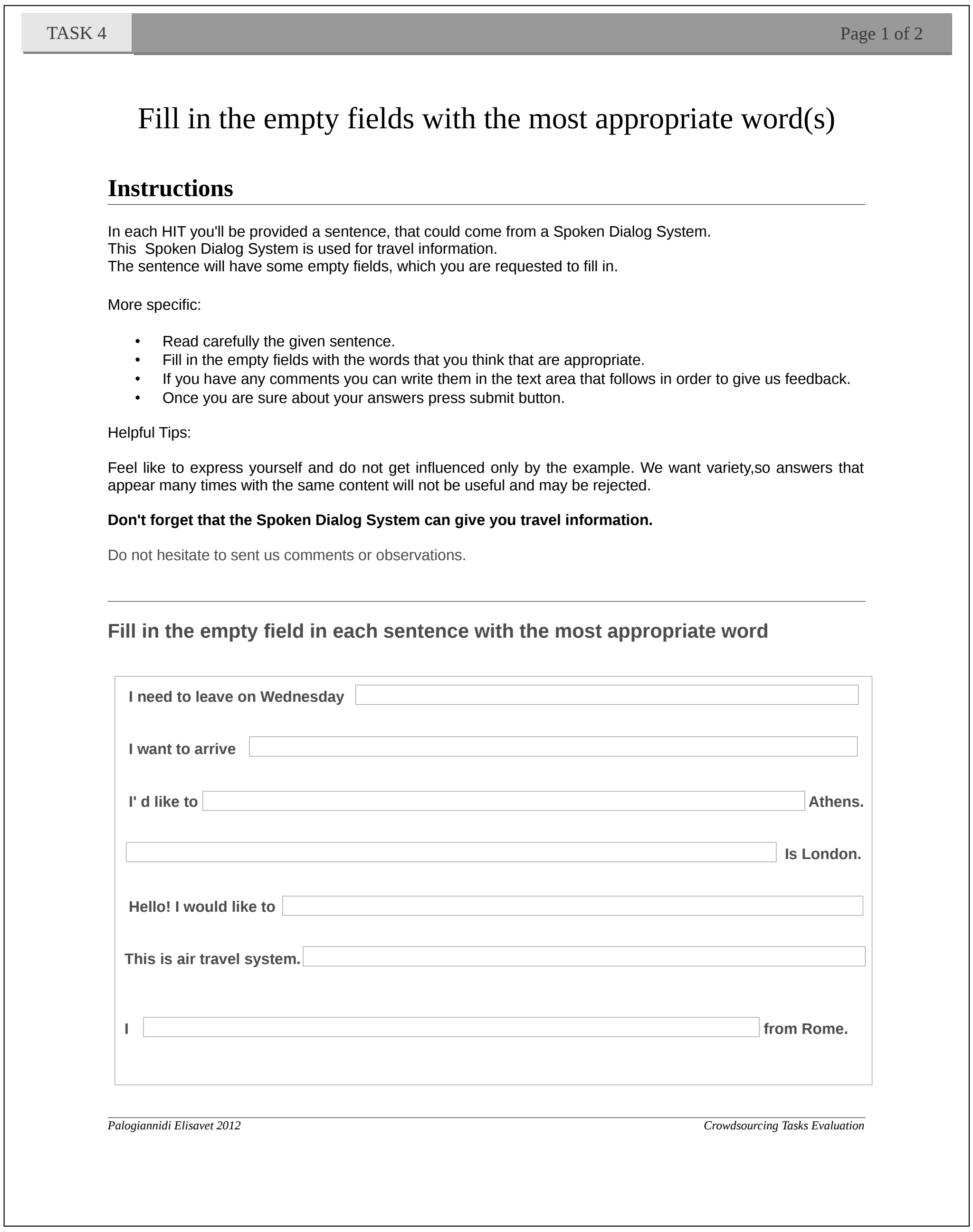
System: Enter a prompt that comes from the system

User : Enter a phrase that comes from the user

System: Enter a prompt that comes from the system

User : I prefer the first.

Figure C.8: “Free Dialogs” task on Crowdflower



TASK 4 Page 1 of 2

Fill in the empty fields with the most appropriate word(s)

Instructions

In each HIT you'll be provided a sentence, that could come from a Spoken Dialog System.
This Spoken Dialog System is used for travel information.
The sentence will have some empty fields, which you are requested to fill in.

More specific:

- Read carefully the given sentence.
- Fill in the empty fields with the words that you think that are appropriate.
- If you have any comments you can write them in the text area that follows in order to give us feedback.
- Once you are sure about your answers press submit button.

Helpful Tips:

Feel like to express yourself and do not get influenced only by the example. We want variety,so answers that appear many times with the same content will not be useful and may be rejected.

Don't forget that the Spoken Dialog System can give you travel information.

Do not hesitate to sent us comments or observations.

Fill in the empty field in each sentence with the most appropriate word

I need to leave on Wednesday ______

I want to arrive ______

I' d like to ______ **Athens.**

______ **Is London.**

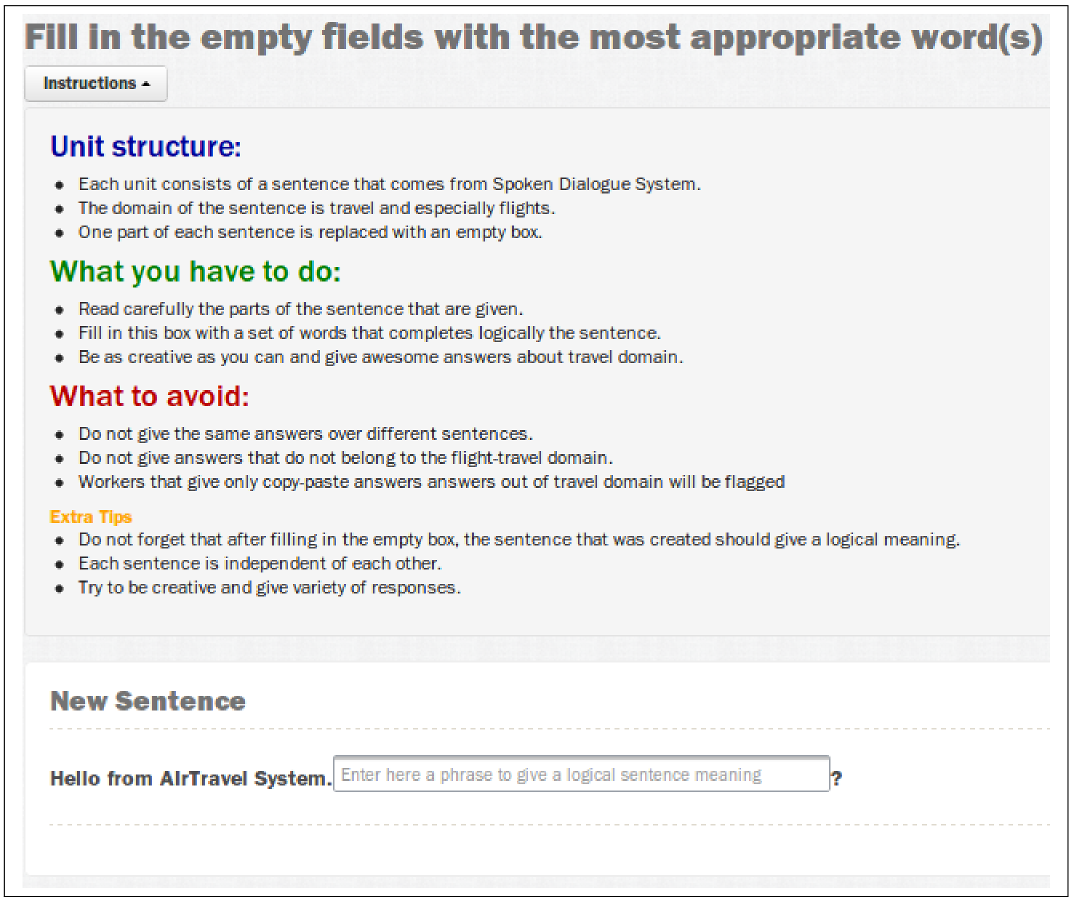
Hello! I would like to ______

This is air travel system. ______

I ______ **from Rome.**

Palogiannidi Elisavet 2012 *Crowdsourcing Tasks Evaluation*

Figure C.9: “Fill in” task on pilot



Fill in the empty fields with the most appropriate word(s)

Unit structure:

- Each unit consists of a sentence that comes from Spoken Dialogue System.
- The domain of the sentence is travel and especially flights.
- One part of each sentence is replaced with an empty box.

What you have to do:

- Read carefully the parts of the sentence that are given.
- Fill in this box with a set of words that completes logically the sentence.
- Be as creative as you can and give awesome answers about travel domain.

What to avoid:

- Do not give the same answers over different sentences.
- Do not give answers that do not belong to the flight-travel domain.
- Workers that give only copy-paste answers answers out of travel domain will be flagged

Extra Tips

- Do not forget that after filling in the empty box, the sentence that was created should give a logical meaning.
- Each sentence is independent of each other.
- Try to be creative and give variety of responses.

New Sentence

Hello from AirTravel System. Enter here a phrase to give a logical sentence meaning ?

Figure C.10: “Fill in” task on Crowdflower

Bibliography

[1] Ambati, V., and Vogel, S. Can crowds build parallel corpora for machine translation systems? In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 62–65.

[2] Audhkhasi, K., Georgiou, P. G., and Narayanan, S. S. Reliability-weighted acoustic model adaptation using crowd sourced transcriptions. In *INTERSPEECH* (2011), pp. 3045–3048.

[3] Bloodgood, M., and Callison-Burch, C. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (2010), Association for Computational Linguistics, pp. 854–864.

[4] Bloodgood, M., and Callison-Burch, C. Using mechanical turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 208–211.

[5] Buchholz, S., and Latorre, J. Crowdsourcing preference tests, and how to detect cheating. In *INTERSPEECH* (2011), pp. 3053–3056.

[6] Buzek, O., Resnik, P., and Bederson, B. B. Error driven paraphrase annotation using mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 217–221.

[7] Callison-Burch, C., and Dredze, M. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 1–12.

[8] Card, S. K., Moran, T. P., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM 23*, 7 (1980), 396–410.

[9] Card, S. K., Moran, T. P., and Newell, A. *The psychology of human computer interaction.* Routledge, 1983.

[10] Carroll, J. M. Human computer interaction (hci). *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* (2013).

[11] Chernova, S., Orkin, J., and Breazeal, C. Crowdsourcing hri through online multiplayer games. In *Proc. Dialog with Robots: AAAI fall symposium* (2010).

[12] Chomsky, N. Three models for the description of language. *Information Theory, IRE Transactions on 2*, 3 (1956), 113–124.

[13] Cooke, M., Barker, J., Lecumberri, M. L. G., and Wasilewski, K. Crowdsourcing for word recognition in noise. In *INTERSPEECH* (2011), pp. 3049–3052.

[14] Crowdflower. `http://www.crowdflower.com`.

[15] Denkowski, M., Al-Haj, H., and Lavie, A. Turker-assisted paraphrasing for english-arabic machine translation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 66–70.

[16] Dix, A. *Human computer interaction.* Pearson Education, 2004.

[17] Evanini, K., and Zechner, K. Using crowdsourcing to provide prosodic annotations for non-native speech. In *INTERSPEECH* (2011), pp. 3069–3072.

[18] Gao, Q., and Vogel, S. Consensus versus expertise: A case study of word alignment with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 30–34.

[19] Gelas, H., Abate, S. T., Besacier, L., and Pellegrino, F. Quality assessment of crowdsourcing transcriptions for african languages. In *INTERSPEECH* (2011), pp. 3065–3068.

[20] Glass, J. R., Polifroni, J., Seneff, S., and Zue, V. Data collection and performance evaluation of spoken dialogue systems: the mit experience. In *INTERSPEECH* (2000), pp. 1–4.

[21] Glass, J. R., and Weinstein, E. Speechbuilder: facilitating spoken dialogue system development. In *INTERSPEECH* (2001), Citeseer, pp. 1335–1338.

[22] Goto, M., and Ogata, J. Podcastle: Recent advances of a spoken document retrieval service improved by anonymous user contributions. In *INTERSPEECH* (2011), pp. 3073–3076.

[23] Gustafson, J., Lindberg, N., and Lundeberg, M. The august spoken dialogue system. In *Eurospeech* (1999).

[24] Higgins, C., McGrath, E., and Moretto, L. Mturk crowdsourcing: a viable method for rapid discovery of arabic nicknames? In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 89–92.

[25] Hirth, M., Hossfeld, T., and Tran-Gia, P. Human cloud as emerging internet application-anatomy of the microworkers crowdsourcing platform. *University of Wurzburg, Institute of Computer Science, Am Hubland, D-97074 Wurzburg, Germany, Research Report*, 478 (2011), 45–46.

[26] Iosif, E. Unsupervised induction of semantic metrics using similarity metrics, 2007.

[27] Irvine, A., and Klementiev, A. Using mechanical turk to annotate lexicons for less commonly used languages. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 108–113.

[28] ISLE. Spoken dialogue system design. `http://www.isle.illinois.edu/synthesis/research_dialog.html`.

[29] Jurcicek, F., Keizer, S., Gašic, M., Mairesse, F., Thomson, B., Yu, K., and Young, S. Real user evaluation of spoken dialogue systems using amazon mechanical turk. In *Proceedings of INTERSPEECH* (2011), vol. 11.

[30] Kittur, A., Chi, E. H., and Suh, B. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2008), ACM, pp. 453–456.

[31] Klasinas, I., Potamianos, A., Iosif, E., Georgiladakis, S., and Mameli, G. Web data harvesting for speech understanding grammar induction. In *Proceedings Interspeec, Lyon* (2013).

[32] Lee, C.-y., and Glass, J. R. A transcription task for crowdsourcing with automatic quality control. In *Interspeech* (2011), pp. 3041–3044.

[33] Levin, E., Pieraccini, R., and Eckert, W. Using markov decision process for learning dialogue strategies. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on* (1998), vol. 1, IEEE, pp. 201–204.

[34] Litman, D. J., and Silliman, S. Itspoke: An intelligent tutoring spoken dialogue system. In *Demonstration Papers at HLT-NAACL 2004* (2004), Association for Computational Linguistics, pp. 5–8.

[35] Liu, S., Seneff, S., and Glass, J. A collective data generation method for speech language models. In *Spoken Language Technology Workshop (SLT), 2010 IEEE* (2010), IEEE, pp. 223–228.

[36] López-Cózar, R., García, P., Díaz-Verdejo, J. E., and Rubio, A. J. A voice activated dialogue system for fast-food restaurant applications. In *Eurospeech* (1997).

[37] Martin, J. H., and Jurafsky, D. Speech and language processing, 2000.

[38] McGraw, I., Glass, J. R., and Seneff, S. Growing a spoken language interface on amazon mechanical turk. In *INTERSPEECH* (2011), pp. 3057–3060.

[39] Pangos, A., Iosif, E., Potamianos, A., and Fosler-Lussier, E. Combining statistical similarity measures for automatic induction of semantic classes. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on* (2005), IEEE, pp. 278–283.

[40] Parent, G., and Eskenazi, M. Speaking to the crowd: Looking at past achievements in using crowdsourcing for speech and predicting future challenges. In *INTERSPEECH* (2011), pp. 3037–3040.

[41] Pargellis, A., Fosler-Lussier, E., Lee, C.-H., Potamianos, A., and Tsai, A. Auto-induced semantic classes. *Speech communication 43*, 3 (2004), 183–203.

[42] Pargellis, A., Fosler-Lussier, E., Potamianos, A., and Lee, C.-H. A comparison of four metrics for auto-inducing semantic classes. In *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on* (2001), IEEE, pp. 218–221.

[43] Pieraccini, R., and Huerta, J. Where do we go from here? research and commercial spoken dialog systems. In *6th SIGdial Workshop on Discourse and Dialogue* (2005).

[44] Pietquin, O., and Dutoit, T. A probabilistic framework for dialog simulation and optimal strategy learning. *Audio, Speech, and Language Processing, IEEE Transactions on 14*, 2 (2006), 589–599.

[45] Raux, A., Langner, B., Bohus, D., Black, A. W., and Eskenazi, M. Let's go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005* (2005), Citeseer.

[46] Ross, J., Irani, L., Silberman, M., Zaldivar, A., and Tomlinson, B. Who are the crowdworkers?: shifting demographics in mechanical turk. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems* (2010), ACM, pp. 2863–2872.

[47] Scheffler, K., and Young, S. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research* (2002), Morgan Kaufmann Publishers Inc., pp. 12–19.

[48] Siu, K.-C., and Meng, H. M. Semi-automatic acquisition of domain-specific semantic structures. In *EuroSpeech* (1999).

[49] Sutton, S., Novick, D. G., Cole, R., Vermeulen, P., de Villiers, J., Schalkwyk, J., and Fanty, M. Building 10,000 spoken dialogue systems. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on* (1996), vol. 2, IEEE, pp. 709–712.

[50] toolkit, C. `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview/`.

[51] toolkit, S. `http://www.speech.sri.com/`.

[52] Traum, D. R., and Allen, J. F. Discourse obligations in dialogue processing. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* (1994), Association for Computational Linguistics, pp. 1–8.

[53] Turing, A. M. Computing machinery and intelligence. *Mind 59*, 236 (1950), 433–460.

[54] Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics* (1997), Association for Computational Linguistics, pp. 271–280.

[55] Wang, W. Y., Bohus, D., Kamar, E., and Horvitz, E. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *Spoken Language Technology Workshop (SLT), 2012 IEEE* (2012), IEEE, pp. 73–78.

[56] Yang, Z., Li, B., Zhu, Y., King, I., Levow, G., and Meng, H. Collection of user judgments on spoken dialog system with crowdsourcing. In *Spoken Language Technology Workshop (SLT), 2010 IEEE* (2010), IEEE, pp. 277–282.

[57] Zaidan, O. F., and Ganitkevitch, J. An enriched mt grammar for under $100. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), Association for Computational Linguistics, pp. 93–98.

[58] Zhu, Y., Yang, Z., Meng, H., Li, B., Levow, G., and King, I. Using finite state machines for evaluating spoken dialog systems. In *Spoken Language Technology Workshop (SLT), 2010 IEEE* (2010), IEEE, pp. 478–483.

[59] Zue, V., Seneff, S., Glass, J. R., Polifroni, J., Pao, C., Hazen, T. J., and Hetherington, L. Juplter: a telephone-based conversational interface for weather information. *Speech and Audio Processing, IEEE Transactions on 8*, 1 (2000), 85–96.

[60] Zue, V., Seneff, S., Polifroni, J., Phillips, M., Pao, C., Goodine, D., Goddeau, D., and Glass, J. Pegasus: A spoken dialogue interface for on-line air travel planning. *Speech Communication 15*, 3 (1994), 331–340.