AN EVALUATION FRAMEWORK FOR ADAPTIVE USER INTERFACES

by

Enrique Noriega Atala

A Thesis Submitted to the Faculty of the

DEPARTMENT OF COMPUTER SCIENCE

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2014

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

Enrique Noriega Atala

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_____ _____

Paul R. Cohen Date

ACKNOWLEDGEMENTS

Many people formed part of this work and without their valuable contributions, either directly or indirectly, this thesis wouldn't have come to exist.

First and foremost, I want to express my sincere gratitude to my thesis committee members: Paul Cohen, my thesis director and adviser, who gave me the opportunity to work with him and the group brilliant of people he leads, introducing me to the exciting, challenging and rewarding world of research. Clayton Morrison, for his mentorship, patience and excellent support throughout the writing of this document and for opening for me the door to of the amazing field of machine learning. And John Hartman, for his excellent academic advisement and for encouraging me to pursue my goals, regardless of any situation, during the course of my graduate studies.

The commitment and the time each of them spent sharing their insight, reviewing my work, and pointing my opportunities to improve was crucial to the completion of my thesis.

I want to thank Marco Valenzuela, Anh Tran and Thanima Pilantanakitti for helping me level up and tutoring me when I started contributing to the research project. All of them spent a considerable amount of their time teaching me how to work in the lab.

Also, my friends Yari Cabanillas, Edgar Molina and Jorge Gallegos, among many others, whose comradeship helped me bear the high demands and expectations of grad school.

CONACyT, COECyT Sonora and Instituto de Educación Sonora-Arizona for providing me with funding to successfully complete my Master of Science degree.

And last, but not least, to my family, my parents and my sisters, for their unconditional love and support.

DEDICATION

In memory of Olga González, my grandmother, who passed away unexpectedly during the course of the research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With the rise of powerful mobile devices and the broad availability of computing power, *Automatic Speech Recognition* is becoming ubiquitous. Despite the rapid advance and active research in the field, a flawless ASR system is still far from existence. Because of this, interactive applications that make use of ASR technology not always recognize speech perfectly and when not, the user must be engaged to repair the mistakes in the transcriptions.

When the user is engaged, he can repair these errors with different strategies. These strategies work better under different contexts. In this thesis, we explore a *rational user interface* that makes use of machine learning models to make its best effort in presenting the best repair strategy available to reduce the time in spent the interaction between the user and the system as much as possible. A study is conducted to determine how different candidate policies perform and conclusions are drawn from the analysis of the collected results.

Once the analysis is performed, the methodology that was used is generalized in terms of a decision theoretical framework that can be used to evaluate the performance of other rational user interfaces that try to optimize an expected cost or utility.

After the generalization, the framework is placed in the context of the relevant literature and some potential future work is outlined.

CHAPTER 1

INTRODUCTION

1.1 Overview

A computer-automated process may require the user's input in order to make certain decisions that will determine its information flow. Sometimes it's hard for software systems to make those decisions because, depending on the nature the problem, it may be infeasible to compute a correct solution. Consider spelling correction software: When the software detects a potential problem, the sentence's portion where the potential problem is located is highlighted and a list of candidate fixes is presented to the user to select from. A possibility is that the word is misspelled or a word that is not present on the dictionary. In either case, it is best for the user to select the proper way to proceed.

Poor design of users interfaces may lead to a large number of user decisions that, in general, may be unnecessary and just bother him and waste his valuable time, worsening the perception of the design, decreasing the efficiency and perhaps making him stop using the system at all. For instance, modern web browsers have autocompletion functionality for personal information web forms. Some users may think that a better job could be done without this feature, because it is common that the corresponding data gets filled into the wrong field, like filling the street address into the city field. This decreases the value of the feature because the user must correct the mistakes and he may spend similar or more time engaged before finishing the corrections than manually entering his information into an empty web form.

This relative high amount of interactions may be the result of giving the user an

excessive degree of control granted by the designers of the interface and/or making him responsible for the testing of certain properties that could be done by the computer for the sake of avoiding wrong decisions or incorrect outcomes. In some cases, user time can be saved by automatically making *some* decisions. Apple's iPhoto image handling software package has a face recognition feature which does a fairly good job at labeling photographs, but requires the user's approval before persisting the labeling. Sometimes, the picture library consists of thousands of images and it becomes tedious to approve individually each of the labeled pictures. For some users, this explicit approval considerably hinders the face recognition feature.

Sometimes, the computer can learn the decisions taken by a user with a high probability of success based on examples. For the sake of user-friendliness, user interactions can be automated using machine learning models that make the decision taken be very close to what the user would have chosen.
Going back to the spelling correction case, some implementations of the software automatically make the correction when the uncertainty about which potential fix is adequate is low. The latest version of Microsoft Word does this for accents in spanish words when omitted. Apple's iOS automatically introduces apostrophes on word contractions when they're detected.

The goal of this work is to describe a methodology and devise a framework to characterize and analyze quantitatively a decision process aided with machine learning methods to determine if it minimizes the user engagement with an interface and maximizes the probability of success by trying to make the best decisions and, as a result, saves time or some other valued resource.

1.2 Organization

The reminder of this thesis is organized as follows:

Chapter Two will describe an experiment from which this thesis was inspired and report its results. It serves as the intuition behind the formal methodology defined on the next chapter. Chapter Three gives a formal statement of the problem and provides a definition of the framework: The foundational concepts from which it is built on top of, the formal characterization of the integration of the decision process with the machine learning models and the definition of the criteria to measure the cost or utility of the user engagement. Chapter Four will provide a survey of related work present in the literature, highlighting the relationships with the work presented here. Chapter Five presents the conclusions and outlines potential future research and open questions that can be further studied.

CHAPTER 2

EXPERIMENT

2.1 The ASR Error Repair Problem

ASR stands for Automatic Speech Recognition, which is the action of transcribing speech in the form of an utterance, which is the acoustic representation of a message, into a character string that represents the sentence uttered by a person. ASR works by building statistical *acoustic* and *language* models that map audio segments to words. In the end, the output of an ASR system is the best *hypothesis* (hyp) it can provide given its statistical models.

The possible hypotheses are encoded in a data structure called a *confusion network* (Figure 2.1), which contains the possible detected words along with their probabilities associated with the edges in a graph-like organization. Each simple path from the head node towards the tail node represents a hypothesis and the product of the probability of each edge is the posterior probability of the hyp given the utterance.

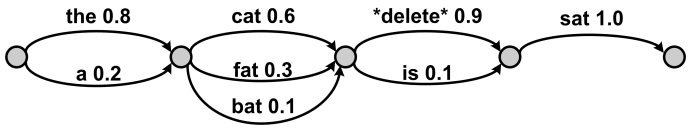


Figure 2.1: Confusion Network example. Each edge is labeled with a possible word plus its probability.

There are several kind of errors in the ASR process that cause the software not to yield a correct answer. Sometimes it may be close enough to be understood and

sometimes it may be completely wrong.

Some of this error kinds are:

- *Out-of-Vocabulary words* (OOV): The language model is built from a finite dictionary. It can't contain all the existing words in a language. Whenever a missing word appears in the utterance, the ASR system will try to match it to one of the existing words on its dictionary. This is very common with proper nouns.

- *Ambiguity*: There may be homophone words that appear on different contexts. This introduces noise in *n-gram* based language models and make the ASR introduce words that don't belong to the utterance. Some words or combination of words may be very similar on their sound but completely different on their meaning. For example: *Recognize speech* and *Wreck a nice beach* have similar sound but are totally unrelated. Depending on the training of the model one may be recognized instead of the other.

- *Noise in the environment*: Poor audio quality and speaker accents may introduce noise on the acoustic model and propagate the error in the ASR pipeline.

There are different strategies to try to obtain the true sentence intended by the user who spoke to the ASR system. Some of those strategies are:

- Highlight a potentially problematic error segment in the hyp and provide an alternative word or combination of words coming from the confusion network. The user can select an alternative segment if he wants to.

- Present the user a list of the top ranked hypotheses from the confusion network and let him pick whichever he deems appropriate, if any.

- Let the user type the uttered sentence himself.

- Utter a rephrased version of the message with less ambiguity and/or noise.

Each of these strategies requires some expense to the user on their own way and involve different expense to the user depending of the problem.
The following experiment analyzes a family of candidate configurations of machine learning models, which we should call policies, to detect when there may be a problem with the hyp and select the best repair strategy. In this context, the best strategy is the one that minimizes the effort of the user. To approximate the effort, we measure the time spent on the interface as a proxy and in our context, the policy that minimizes time also minimizes the effort. With the results of the study, it can be determined if it is worth to use one of the policies or not and how much value it is gained by using it.

2.2 Interface Description

We propose a user interface that tries to present the most appropriate repair strategy. To decide which strategy is this, the computer will perform a series of tests on the hypothesis and use these results to decide how to interact with the user. This estimations are be based on empirical data that was gathered during a study.

The study was performed during October 2013 to measure the time spent by people using an interface to repair ASR transcriptions using some of the strategies outlined above. The study took place in the context of a DARPA-funded research project for the development of a real time machine translation device in which the output of the ASR system is translated to a foreign language and then speech is synthesized based on the translation. To avoid noise and loss of information, the system tries to provide the most accurate transcribed sentence, which may be the result of fixing the ASR's hyp.

The dynamics of the user interface are the following: The user is presented with the top *hypothesis* generated by the ASR. After analyzing the hyp, he will determine whether it's correct or wrong (See Fig. 2.2).

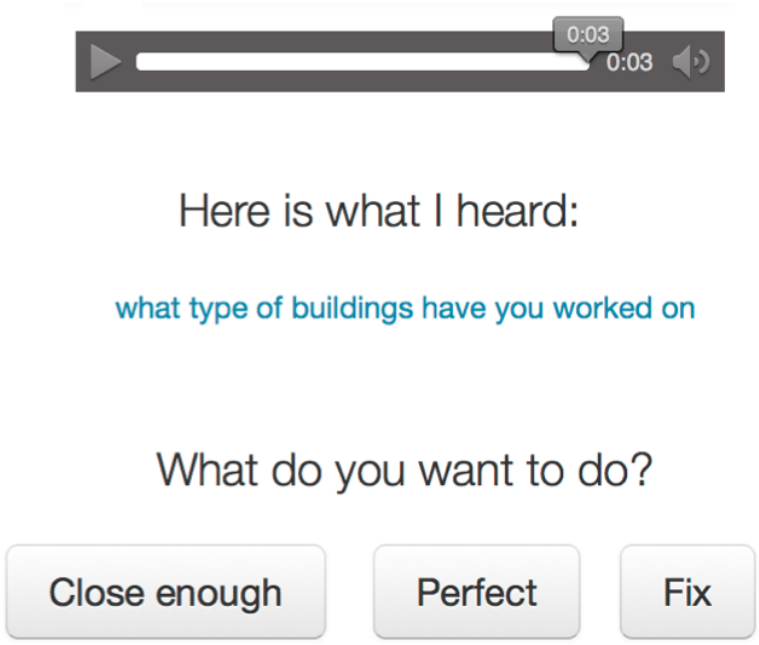


Figure 2.2: UI's main screen

The user has the option to determine whether the transcription is a perfect match with the utterance, a close enough match or none. In the former two cases, the transcription continues through the translation pipeline; the latter case allows him to determine a fix strategy to improve the fidelity of the hyp.

We consider two ways to fix the hyp. First, the user can be presented with a list of the N-top ranked hypotheses generated from the confusion network, where N is a parameter adjusted by the developer of the system. Hypotheses are ranked by the joint probability of their words in the confusion network. The joint probability of a hyp that results from a confusion network is calculated by taking the product of their individual likelihood. Even though hypotheses that come from from the same confusion network may have a different number of words, they will have the same number of tokens in the confusion network. The missing word is represented by a *DELETE* token, with its respective probability associated. Because of this,

the probabilities of the two hypotheses are comparable.
The second way is to type the sentence completely from scratch. (See Figs. 2.3, 2.4). Note that some candidate hypotheses in Figure 2.3 may appear twice. This is because words like *huh* and *ah* are omitted on the interface.

Our intuition tells us that if the best hyp preserves enough meaning to carry the same message as the uttered sentence, it is very likely that the correct sentence will appear among the top ranked hyps. If the presented hyp doesn't preserve the meaning, it will be easier to type it rather than trying to inspect the possible outputs from the ASR and very likely not finding it.

One possible way of designing the interface is to give the user the choice to pick either way to fix the sentence. If he finds the correct sentence among the N best ranked hyps, he will select it and send it down through the translation pipeline. If a perfect hype is not present, but there is a hyp that preserves enough meaning to be considered *close enough*, he can pick it and send it through. If none of the alternative hyps are acceptable for the user, he can still fix it by going to a new screen to type the message. Refer to Figure 2.3.

Observe that if the user knows that he was not going to find a correct or satisfactory answer, he would have chosen the retype screen from the beginning instead of wasting time scanning the N alternative hypotheses.

2.3 Methodology

Forty five human users participated by evaluating the ASR-repair interface. Each of them used the system for thirty minutes. The system was set up to randomly select hypotheses from the *TRANSTAC corpus* [Bach et al., 2007]. The TRANSTAC corpus was built for the development and testing of two-way speech-to-speech translation systems from the recordings of speech from deployed military personnel,

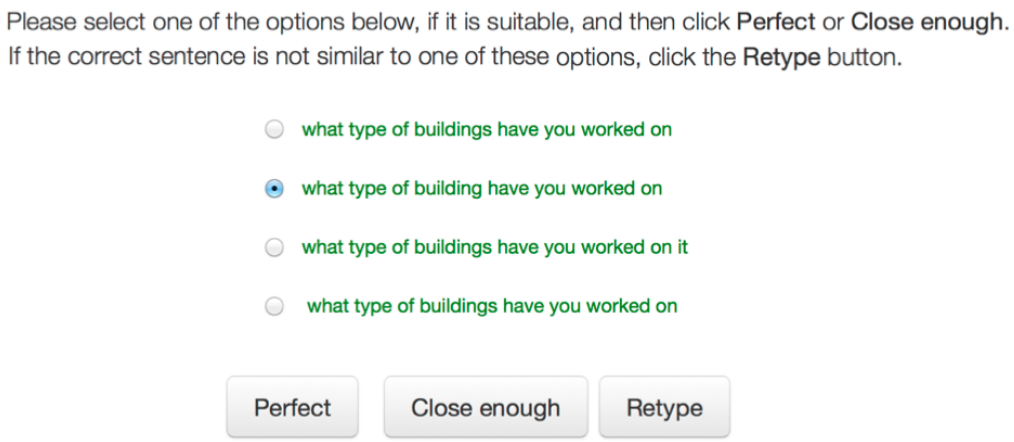


Figure 2.3: N best hyps screen

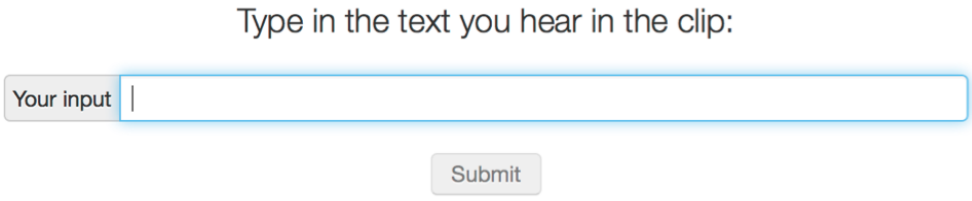


Figure 2.4: Retype screen

and was manually transcribed to provide ground truth transcriptions for training and testing. For each shown hypothesis, the sentence was presented through the user interface and it was up to the evaluator to fix it using one of the available strategies or decline to fix it. Each instance of the information flow was collected for postprocessing.

At the end of the study, a total of 2,404 interface use instances were collected. Each instance consists of the original sound clip, the ground truth *reference transcription of the uttered sentence* (ref), the top hyp, the sequence of screens the user followed along with the time he spent on each, the particular outcome of the process and the repaired sentence, if a repair was made.

2.4 Analysis

In this section, the study will be modeled more methodically to serve as an initial intuition of the generalization of the methodology presented in chapter 3.

Every time a user is presented with a hyp, he can decide which strategy to follow to repair or not to repair at all. Depending on which strategy was used, a sequence of screens will be presented and the original hyp will be transformed into something that (hopefully) captures the meaning of the message better.

The first decision the user will take is whether the output of the ASR is correct or not. After this decision is made, if it is determined that the hyp is not correct, a second screen will be shown where a list of the top 4 ranked hypotheses is displayed. The rank of a hyp is determined by the joint probability of its words given the confusion network (Figure 2.1).

If it is determined that none of the alternative hypotheses preserves the meaning of the message, the user will proceed to a *retype* screen in which he will manually

enter the sentence (Figure 2.4).

Given the randomly selected hyp from the dataset, the user follows one of three possible use patterns: *No Correction, Better choice* and *Retype*. This is visualized in Figure 2.5, where the squares represent the screen shown and the ovals the state of the message after finishing.

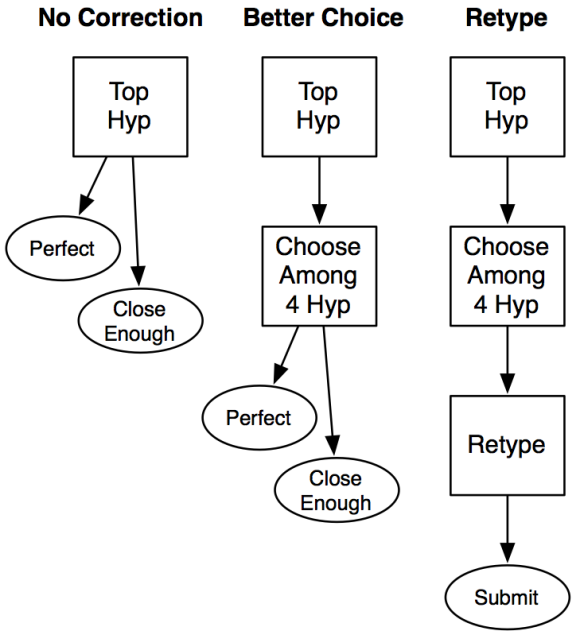


Figure 2.5: Possible outcomes of a sampled *hyp* from the dataset.

Pattern:	No Correction	Better Choice	Retype
Frecuencies:	1850	197	357

Table 2.1: Count of pattern instances

Table 2.1 summarizes the frequency of the patterns. From the $2,404$ instances, $1,850$ were not fixed, corresponding to the *No correction* pattern from Figure 2.5, 197 were fixed by selecting an alternative hyp from the list, corresponding to

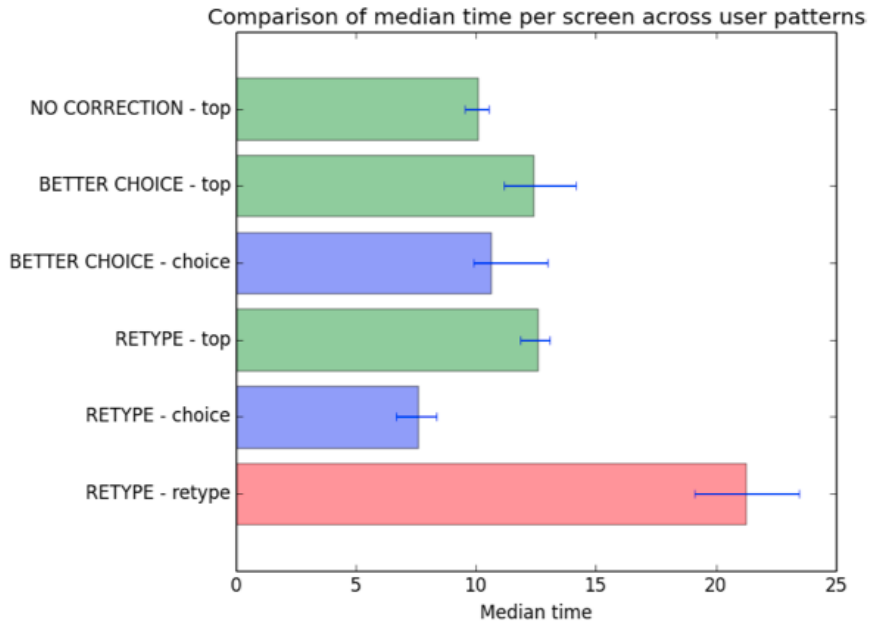
the *Better choice* pattern and 357 were retyped, which corresponds to the *Retype* pattern.

With the recorded times, some statistics of the time spent on each screen can be computed. Figure 2.6 shows the median and mean times for each screen where *top* is the main screen with the *top hyp* (Figure 2.2), *choice* is the screen that displays the list of the four best ranked hyps (Figure 2.3) and *retype* is the screen where the user types the message (Figure 2.4). On Figure 2.6 the time on each screen is analyzed for each of the user patterns described above.

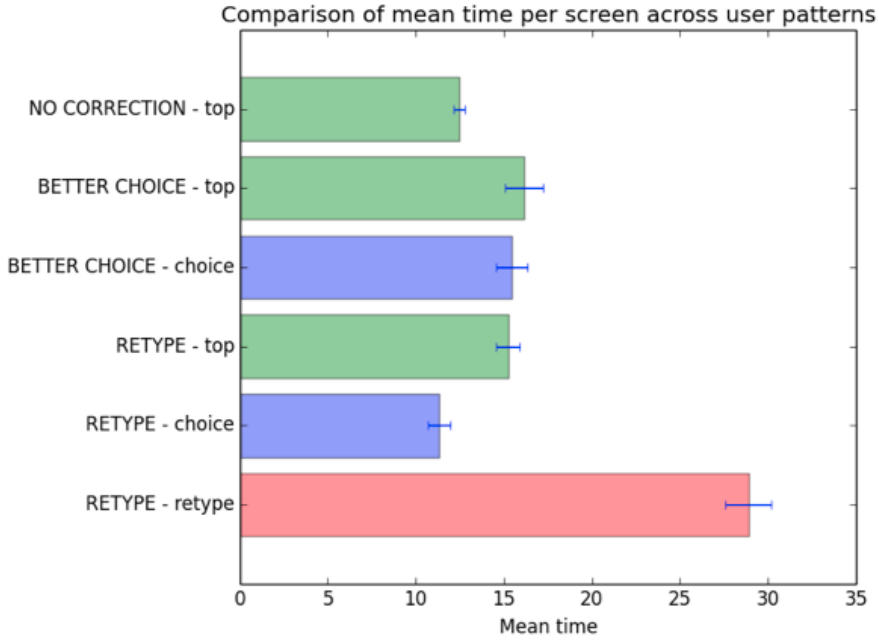
The whiskers on Figure 2.6a represent a bootstrap estimate of 95% confidence interval around the median. Similarly, the whiskers in Figure 2.6b represent the 95% confidence interval around the mean, defined by: $\hat{\mu} \pm 1.96\frac{s}{\sqrt{n}}$.

For example, In Figure 2.6b the green bars labeled as *NO CORRECTION - top* represents the average time spent on main screen when no correction was done by the users and the red bar labeled as *RETYPE - retype* is the average time spent on the retype screen when user chose to type again the message.

This measurements support the intuition that retyping the message is the most time consuming task overall, regardless of the statistic chosen to represent it.



(a) Median times in seconds per screen.



(b) Mean times in seconds per screen.

Figure 2.6: Statistics of empirical times per screen decomposed by usage pattern

2.4.1 Candidate Policies

We try to approximate the decisions that the users make by defining a *policy*. A policy can be characterized as a decision process that will determine which repair strategy to use after a series of tests performed on the hyp. These tests can be performed by machine learning models that will try to pick the most appropriate choices depending on the properties of the input hyp. A decision process is depicted in Figure 2.7. The two diamond shaped nodes represent the machine learning models, and depending on their output, the designated repair strategy will be used.

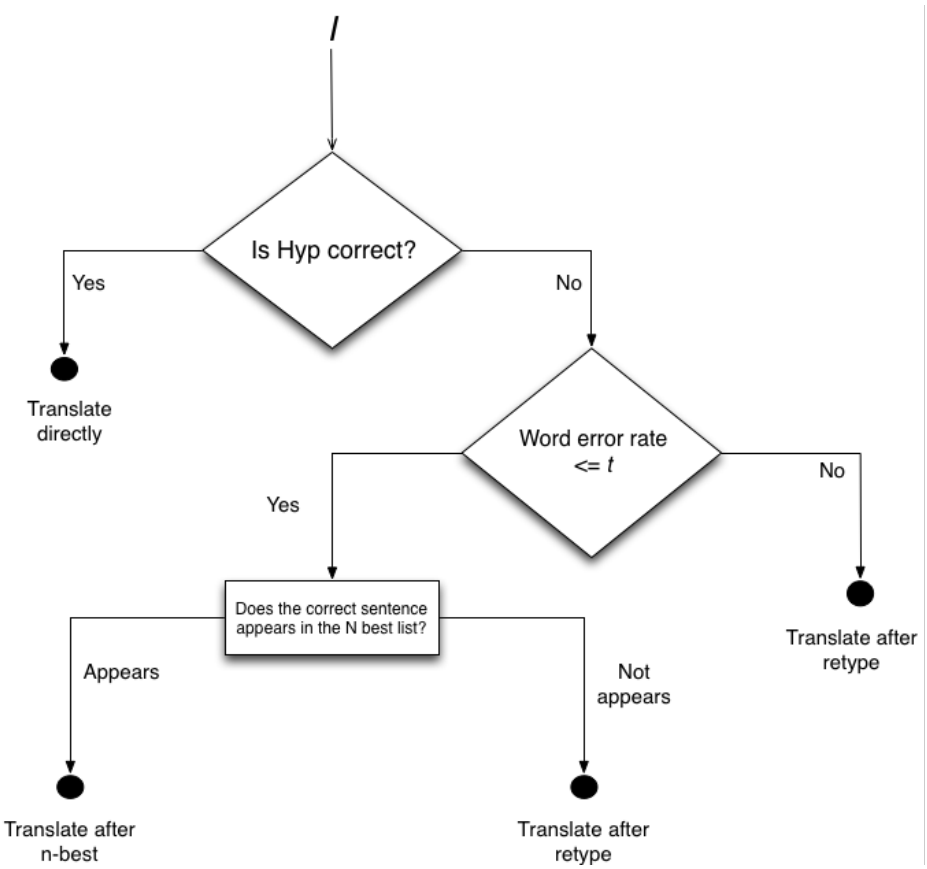


Figure 2.7: Template of candidate policies family.

The upper diamond represents a classifier that predicts whether the hyp is right or wrong. Depending on the outcome, the second model may be used, represented

by the lower diamond, which is a linear regression classifier to predict the *word error rate* with a threshold to discretize its outcome. Parameter t is the threshold that can be adjusted to define different policies, thus, a concrete policy is defined by assigning a value for t and a family of policies is composed of the concrete policies with different values for t.

The linear models that make the decision on the diamond shaped nodes from Figure 2.7 are the following:

1. The first classifier is a *Logistic regression classifier*, which classifies the hyp as correct or incorrect. This is a binary classifier.
   The features used by the logistic regression are:
   - The maximum entropy of the slots of the confusion network.
   - The minimum entropy of the slots of the confusion network.
   - The ratio between the number of tokens in the hyp with the number of tokens on the hyp after being aligned with the confusion network.
   - The number of delete tokens included into the aligned hyp.
   - The probability of the hyp given the confusion network.
   - The hyp's score generated by the alignment algorithm.
   - The hyp's acoustic model score from the ASR.
   - The hyp's language model score from the ASR.
2. The second classifier is a *Linear regression model*, which predicts the *word error rate*, defined as the number of incorrect tokens over the total number of tokens in the hyp. This is a regression model, which outputs a continuous value, because of this we need the threshold t discretize the output and make it a binary classifier.
   The features used by the logistic regression are:
   - The hyp's language model score from the ASR.

- The hyp's score generated by the alignment algorithm.
- The number of delete tokens included into the aligned hyp.
- The number of tokens in the hyp.
- The inverse of the number of tokens in the hyp.

Observe that the square node will require engagement from the user by forcing him to scan the *N-top ranked hyps*. In the context of this study, the value of N is 4.

Policy names will be referred to in bold for clearness inside the equations and to distinguish them of usage patterns, which are written in italics. Each member of the family of candidate policies described above is referred to as **Choice** $\leq t <$ **Retype**, where the value of t varies depending on the threshold for the particular classifier. Conversely, a similarly defined set of candidate policies referred to as **Retype** $\leq t <$ **Choice** Where the outcomes of the linear regression classifier are reversed.

The policies **Always Retype** and **Always Choice** only make use of the first classifier to predict whether the output of the ASR has at least one error or more, and if it has, the Better choice screen is presented in the former policy and the retype screen on the latter policy.

2.4.2 Expected Costs of Policies

We need a way to quantify in general the expense of the users when using a candidate policy. For this experiment we can use the *expected time* on the interface given a policy. In order to calculate the expected time, we need some date to base our estimation in.

First, we the user take their own decisions and measure the times spent on each screen. Then, we use this data to estimate how long an enforced policy will take on

average by post-processing each of the usage patterns to calculate the expected time spent on each screen. The times will be aggregated *only* when the collected data that is consistent with the prediction of the policy. This means that only the usage patterns that follow the same path as the prediction of the policy will be aggregated and accounted for analysis. The expected cost of a particular policy is estimated using the times observed in the study. For example: to calculate the expected time of the *Better choice* pattern (Fig. 2.5), only the subset of measurements in which a user decided to use one of the alternative hyps in the list will be used to calculate the expectation.

The top screen will be ignored because it will always appear regardless of which policy is followed. Only in the other screens the expected times may vary. As shown on Figure 2.6, the **Always Retype** policy we only consider the times spent in the retype screen, which appears in 357 instances (Table 2.1). Because this policy always uses the same strategy to fix the hyp, hence the same screen, the expected cost is simply the average time of all the usage patterns.

The expected time of the **Always Choice** policy is calculated similarly, aggregating the times by calculating the average of corresponding to 197 of the instances, but in some instances there may not be an alternative hyp present on the list that satisfies the user and in those cases he proceeds to the retype screen. To deal with this it is necessary to combine the expected value of two proportions: The ones in which a satisfying hyp appears on the N top ranked list and the ones in which not. This expected time is characterized in the following equation:

$$\begin{aligned} \mathbb{E}(\textbf{Always Choice}) = \quad & p(\text{hyp in list}) \times T_\mu(\text{Better choice made}) \\ + \quad & p(\text{hyp not in list}) \times T_\mu(\text{Retype necessary}) \end{aligned} \tag{2.1}$$

Lets analyze the components of (2.1): T_μ stands for the mean time over a

sample of usage patterns. When a better choice was made, only the time spent in the better choice screen is considered, but if a retype was necessary, the user spent time scanning the list and then typing the message; in this case it is necessary to consider the time spend on both screens by adding their values.

p(hyp in list) is the probability of the event of the hyp being in the top 4 ranked list. To calculate the times the hyp appears must be counted over all the data set and normalized by dividing the count over the size of the data set. It has been decided to include the counts over the complete data set because presumably, when no repair is needed, the top hyp, which is the first element of the list and appears on it, matches the uttered message. These probabilities have the following values:

$$p(\text{hyp on list}) = \frac{|\text{No Correction}|+|\text{Better Choice}|}{|\text{No Correction}|+|\text{Better Choice}|+|\text{Retype}|} = \frac{764+197}{764+197+357} = 0.729 \tag{2.2}$$

$$p(\text{hyp not on list}) = \frac{|\text{Retype}|}{|\text{No Correction}|+|\text{Better Choice}|+|\text{Retype}|} = \frac{357}{764+197+357} = 0.271 \tag{2.3}$$

Where the vertical bars represent the cardinality of the specified subset of the dataset. p(hyp not in list) is the complement.

The family of policies **Choice** $\leq t <$ **Retype** are slightly more sophisticated. There is now an extra model in the decision process, the linear regression, that decides which strategy is going to be used to fix the hyp. We can understand these policies as a combination of **Always Retype** and **Always Choice** in which the complete data set will be divided in two pieces by threshold t and each piece will behave as one of the previous policies. The expected time for each policy is characterized as:

$$\begin{aligned}\mathbb{E}(\textbf{Choice} \leq t < \textbf{Retype}) = p(\leq t) \times \Big[& p(\text{hyp on list}) \times T_\mu(\text{Better choice made} \mid \leq t) \\ & + p(\text{hyp not on list}) \times T_\mu(\text{Retype necessary} \mid \leq t) \Big] \\ & + p(> t) \times \Big[T_\mu(\text{Always Retype} \mid > t) \Big]\end{aligned} \tag{2.4}$$

The terms $p(> t)$ and $p(\leq t)$ represent the probability of the top hyp being above or below threshold t. Observe how (2.4) has an equivalent formulation in terms of expectations:

$$\begin{aligned}\mathbb{E}(\textbf{Choice} \leq t < \textbf{Retype}) = & \; p(\leq t) \times \mathbb{E}(\text{Always Choice} \mid \leq t) \\ & + p(> t) \times \mathbb{E}(\text{Always Retype} \mid > t)\end{aligned} \tag{2.5}$$

Each of the expectations on the right side of (2.5) works like the previously described policies but only count the instances in which the linear regression model predicted that the *word error rate* will be above or below the specified threshold on conditioning inequality. The probabilities of being above or below the threshold are also calculated by counting and normalizing over the size of the complete dataset. Table 2.2 summarizes the frequencies of usage instances by the users for the different thresholds and membership on the alternative hypotheses list to compute the probabilities.

Similarly, the expected time for the members of the family of policies **Retype** $\leq t <$ **Choice** is defined as:

$$\begin{aligned}\mathbb{E}(\textbf{Retype} \leq t < \textbf{Choice}) = & \; p(\leq t) \times \mathbb{E}(\text{Always Retype} \mid \leq t) \\ & + p(> t) \times \mathbb{E}(\text{Always Choice} \mid > t)\end{aligned} \tag{2.6}$$

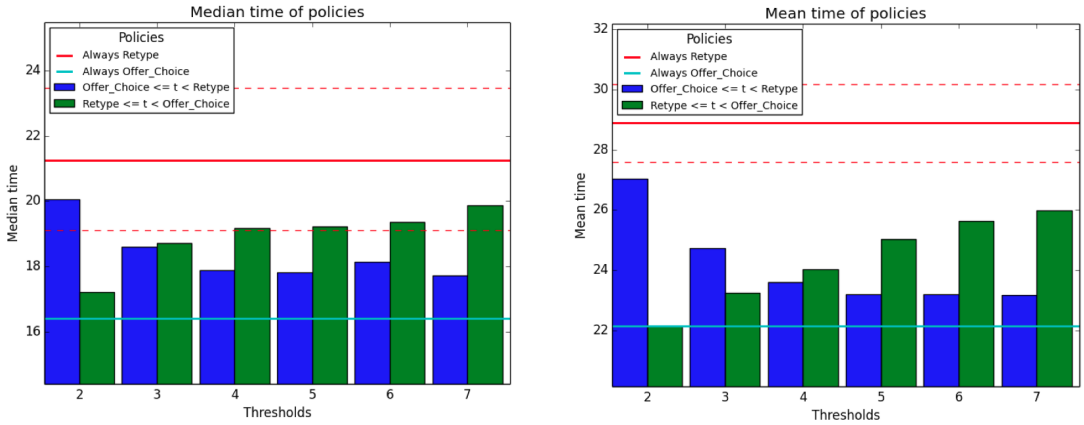
	t	2	3	4	5	6	6
$\leq t$	4-best list *has* solution	214	447	625	729	793	852
	4-best list *doesn't have* solution	31	98	150	206	249	286
	Total	245	575	775	935	1042	1138
$> t$	4-best list *has* solution	747	484	336	232	168	109
	4-best list *doesn't have* solution	326	259	207	151	108	71
	Total	1073	743	543	383	276	180

Table 2.2: Counting summary for the different values of threshold t and membership to the 4-best list

With the definition of the expected times as estimated based on the screen times observed in the study, we can compare how well the policies might perform if actually run. Figure 2.8a shows the median times for all the policies. Because the **Always Retype** policy doesn't vary with the threshold, its median time is drawn as a solid red line; the dashed red lines represent the boundaries of the 95% quantile. The **Always Choice** policy appears as a solid cyan line. Given the high probability that the solution is present in the 4-best list, the wide gap between both policies follows the intuition that, in general, retyping is considerably more expensive than the simple act of picking one of the choices, if present on the top ranked list.

The bars represent the varying policies, where the blue ones are **Choice** $\leq t <$ **Retype** policies and the green ones are **Retype** $\leq t <$ **Choice**. Each pair of bars correspond to the threshold value of the regression model predicting the word error rate in the hyp. The threshold increases left to right along the horizontal axis.

Figure 2.8b presents the expected times where the chosen statistic is the mean; the dashed lines represent the 95% confidence interval.



(a) Median times (b) Mean times

Figure 2.8: Time statistics for all the policies

2.5 Study Conclusions

The initial hypothesis is that we could improve the overall expected time of the error correction on the ASR system by introducing a policy that determines tries to predict when the user is more likely to retype the hypothesis, this way he avoids wasting time with other repair strategies or, on the other hand, predict when the real message is present in the top ranked list and let the user pick it himself quickly. Following the intuition that the predicted word error rate may be a good criteria to select the best strategy given the input data, we found that the error-predicting regression classifier offered no advantage over simply always giving the user the choice.

This result doesn't mean that there is no way to successfully predict which repair strategy to use. It only states that using the classifiers for the study there is no information that let the polices perform better than looking at the 4 top ranked hypotheses to make a choice.

Nonetheless, the lessons learned from the study will work as a good ground to generalize the methodology and apply it to similar problems that involve rational user interfaces.

CHAPTER 3

ANALYSIS FRAMEWORK

3.1 Characterization of a decision process

In the previous chapter, a rational user interface was analyzed to determine if it brings value to its users. While it was just a single example, the analysis strategy can be generalized in order to be used on any rational UI that makes use of machine learning methods to optimize its efficiency in the same fashion as the *ASR repair interface*.

To generalize the analysis, its components and steps will be characterized in decision theoretic terms. Recall the structure of the candidate policies from Figure 2.7. A policy received a hypothesis and its confusion network as an input and based on the classifiers, it determined which repair repair strategy to use based on *two sequential tests*.

A more general description of this problem can be formulated by representing a policy as a *Decision tree* (Figure 3.1). Under the context of decision theory, decision tree is the representation of a compound *lottery*.

A lottery combines distribution over possible outcomes with associated utilities. Each of these outcomes has an associated *value* and a probability of happening. For example, a contestant in a T.V. show has already in his pocket a $100 reward. He has the choice of spinning a wheel, in which he could duplicate his reward with a chance of 10% or lose it with a chance of 90% or going home with what he already has.

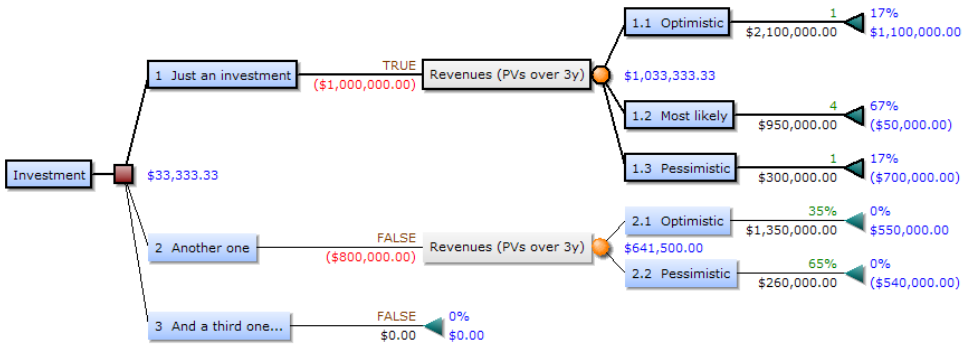


Figure 3.1: Example of a decision tree

The expected value of the gamble is:

$$\frac{10}{100} \times \$200 + \frac{90}{100} \times \$0 = \$20$$

The expected value of the gamble is considerably lower than his original hundred dollars. If the contestant is *rational*, which means that he will follow the rationality axioms of decision theory in which the decision maker tries to maximize his earnings, he will decide not to take his chances and go home because he is expected to earn more by not playing than if he played the lottery.

The policies analyzed in the experiment follow a well structured decision process, hence have a decision tree structure; the outcomes of the process are the repair strategies. We can define the set of outcomes as:

$$\mathcal{O} = \{\text{No Correction, Better Choice, Retype }\} \tag{3.1}$$

Of course, in order to pick one of the outputs, there must exist the information to make the best decision, or at least try to make it. This information may take any arbitrary shape and size. In the concrete case of the experiment, the input information takes the following form:

$$I = (\text{Hypothesis, Confusion Network}) \tag{3.2}$$

We can abstractly define the set of all possible inputs I as $\mathcal{I}$. Even when in some concrete problems $\mathcal{I}$ won't be feasible to compute, or even possible at all, it is an important symbol for the definition of the framework.

If the policy is treated as a black box, it will define a mapping from the input to one of the outcomes; one hyp and confusion network pair to a repair strategy. This can be viewed as a mapping, where π is the policy:

$$\pi : \mathcal{I} \mapsto \mathcal{O} \tag{3.3}$$

Given an input, reaching an outcome will have some utility, the utility for each outcome may be different. The utility follows the intuition of how much value does the user gets if he reaches an outcome when presented with a specific input. In general, we define the utility of an outcome given an input as a real valued function:

$$u(o \mid I) \in R \tag{3.4}$$

Alternatively the same idea can be expressed on terms of *costs*, where the cost of reaching an outcome intuitively represents the expense of a resource by the user when reaching the outcome if he is presented with a certain input. The cost is also a real valued function:

$$c(o \mid I) \in R \tag{3.5}$$

The experiment is more naturally expressed in terms of costs, where the cost of using a repair strategy is the time spent by the user engaged on the screens that implement it. Since the interpretation of the results privileges lower time spent on the system, the best policy is the one that minimize it.

3.2 The problem with complex decision processes

A decision process π may include several tests in a sequence that will steer the flow of information and action to lead to a particular outcome. In the experiment,

whenever a hyp was classified as incorrect, another test was performed to predict it's word error rate using the linear regression model. This nested structure of the models is what give the process the tree shape and what makes it a compound lottery.

In a compound lottery, one or more of the outcomes is another lottery itself, which has its own outcomes with their own distributions and costs.

To deal with a nested lottery (subtree on the tree representation), the value considered as the cost or utility of that lottery is the *expectation* over the induced distribution. We can see in equations (2.4) and (2.5) how the structure of expectations arises from the recursive nature of the policy.

Having a probability distribution over the outcomes of each decision, introduces uncertainty, which means that the selected outcome may not be the best. So, in order to do the best effort in selecting it, we need to calculate the expected costs taking into account the possible outcomes and their probabilities.

Because of this, it is necessary to establish a baseline, gold standard, as a reference frame to contrast the candidate policies. The gold standard and the metrics of the policies must be obtained using the same methodology to be comparable and be able to answer the following questions: *Is the policy bringing value to the users?, How well does a candidate policy performs with respect to other candidate policies?* and *Is it worth to use a policy at all?*.

3.3 Computing the expected costs of the decision process

The decision process representing a candidate policy π can be defined as a decision tree T, where each of the tests is represented as a node n.

Each of the children of n is either another test dependent of the result of n, a *random node* which will follow one of its descending paths according to a *probability distribution* or an outcome in $\mathcal{O}$ (3.1). If n has no parent, it is the root of the tree and it is the *first test of the process*.

If a node in the tree is a decision node or a random node, it must have *at least* two children. If it is a leaf node, it *must be* an outcome.

Let $ch(n)$ be the direct descendants of node n. For any given n that is not a leaf in T, that is, $ch(n) \neq \emptyset$, the probability of happening for each of its descendants $n' \in ch(n)$ is denoted by $p(n')$. The sum of their probabilities is a probability distribution:

$$\sum_{n' \in ch(n)} p(n') = 1$$

We define the *expected cost* or *expected utility* of a node n of T equivalently as follows; we will stick to costs for consistency with the experiment:

1. If n is an outcome o, the expected utility is given by:

$$\mathbb{E}\left[c(o)\right] = \sum_{I \in \mathcal{I}} c(o \mid I) p(I \mid o) \tag{3.6}$$

   This is, the average of the possible costs for o weighted over the conditional probability distribution of the inputs given o.

2. If n is a decision node or a random node, its expected utility is given by:

$$\mathbb{E}\left[c(n)\right] = \sum_{n' \in\, ch(n)} \mathbb{E}[c(n')] p(n' \mid \pi(I)) \tag{3.7}$$

   That is, the weighted average of the posible results of the test or estimation performed by n.

   We can observe that this definition is recursive, where the base case of the recursion happens when $n \in \mathcal{O}$.

Overall, the expected cost for T is given by the expected cost of its root node r:

$$\mathbb{E}\left[\pi^*(T)\right] = \mathbb{E}_p\left[c(r)\right] = \sum_{n' \in\, ch(r)} \mathbb{E}_p[c(n')]p(n' \mid \pi(I)) \tag{3.8}$$

Observe that there is now a criteria to optimize any *candidate policy* π: The expected cost of π. The goal of the framework is evaluating the quality of π using this criteria.

In order to qualitatively evaluate the performance of the automated decision process, we need to have a baseline to compare it with. There are different criteria that will answer different kind of questions depending on what do we want to quantify.

Recall that some of the questions to answer are:

- Is it better to use π or to always pick a fixed action?
- How well does the candidate policy π performs against an "empirical" policy defined by the users in average?
- Do the users have a decrease on the cost on average by using the candidate policy π at all?

First of all, it is useful to know whether the candidate policy is useful at all. It may be the case that choosing a fixed outcome instead of giving a choice is less costly on average than using π. To answer this question we need to build a baseline by proposing a *trivial model* for the decision process' outcomes. The trivial model is the set of expected costs for all the outcomes in $\mathcal{O}$. Since regardless of any property of the inputs, all of them will result on the same outcome, we say that the elements of the trivial model are *trivial outcomes*. This model represents an approximation to the cost of each of the outcomes in π.

The expected cost of the trivial outcomes are defined as follows:

$$\left\{ \sum_{I \in \mathcal{I}} p(o \mid I) c(o) \mid \forall o, I \in \mathcal{O} \times \mathcal{I} \right\} \tag{3.9}$$

This means that for any outcome, its expected *cost* is the average cost over all the possible inputs of the problem. The expected cost for any trivial outcome policy would reduce to this expression because all the inputs of the data set finished on it.

Policies **Always Choice** and **Always Retype** are the trivial policies for the experiment. In Figure 2.8, the solid red and cyan lines represent the expected cost of the trivial models for the *ASR error correction interface* user different choices of statistic.

3.4 Computing the expected costs of the candidate policy

Our candidate policy π (3.3) will have a set of models for the different attributes we want to test before making a decision, like the word error rate or the correctness of the hyp. If we iterate over all the elements in $\mathcal{I}$, which are the possible input values, evaluating each with π, each instance will eventually reach an outcome by traversing the tree. The differences on the paths that I_1 and I_2 may take, will depend on their properties, which determine the result of each of the tests performed by the models.

The probability of each local outcome in the decision process is the associated probability on the compound lottery. This probability is determined by counting the frequencies of the outcome, as in the case of the experiment or by modeling it using a Bayesian approach if we bring a prior over the outcomes to the analysis.

With this definition, we can calculate the expected cost of the policy by using (3.8), the expected cost equation. This will recursively calculate the expected costs

of the outcomes at each level of the tree until reaching the leaves.

Now, if the expected cost of the policy is less desirable than any of the trivial models, then the performance of policy π is worse than trivially picking at least one of those outcomes in $\mathcal{O}$.
This what we observed in the study, in which policy **Always Choice** consistently beats the policies in families **Choice** $\leq t <$ **Retype** and **Retype** $\leq t <$ **Choice**.

By calculating the expected cost of all candidate policies, they can be directly compared against each other. The best candidate policy is the one with the minimum cost or maximum utility. Using this criteria, interpreting Figure 2.8 concludes that threshold 7 yields the most successful policy from family **Choice** $\leq t <$ **Retype** when considering mean and median times and $t = 5$ when considering the third quartile.

3.5 Empirical Policy

An empirical policy is the result of letting the users perform the tests that the classifiers of policy π would have predicted and computing the expected cost using the distributions that the users followed. Lets address the question *"How well does the candidate policy π performs against an "empirical" policy defined by the behavior of the users?"*. This question is of a different nature, because now we are not questioning the usefulness of π but contrasting it to the observed behavior of the users without any aid.

Let T^* be another decision tree that shares the structure of T but instead of using learning models, the users will select how to proceed themselves, effectively making all the decisions.

A study can be conducted with representative population of users that tests the interface. If we use the same formulation of the problem and apply (3.8), we can

compute the expected cost of the empirical policy.

Evidently, the distribution of inputs over the outcomes may change, and by consequence the expected costs. Also, the cost function may yield different results, presumably because the users will take more time time to perform the tests than the models. These are fundamental differences that may add value to the use of machine learning models instead of relying on the users to select the outcomes manually, but are still important because they allow us to measure the benefit of using a machine learning enhaced decision process.

By considering the difference of the expected cost of the empirical policy and that of a candidate policy of our choice, it can be determined how much benefit is obtained, thus answering also the question.

3.6 Summary

The analysis of a policy depends in general on the choice of cost metric and the distributions of the outcomes induced by the classifiers in the policy.
The distributions of the lottery and the cost or utility function will dictate how well π performs either globally or locally for each outcome with respect to another policy or to the empirical policy.

This gives the designers of the UI the required tools to refine the models to skew the distributions based on the metric of their choice to perform better in general or in particular to an specific subset of outcomes. It also provides them with a guideline to make an informed comparison of the machine learning models they could use or which specific ones tune in order to achieve the desired effect. If the designers want to calibrate the cost in general, they can train the models to jointly minimize the costs for all of the outcomes using a training algorithm like SEARN [Iii et al., 2009]

or DAGGER [Ross et al., 2010].

CHAPTER 4

RELATED WORK

There is plenty of work in the literature related to user interfaces and the criteria to qualify their usability and effectiveness.

Some authors, like Nielsen [Nielsen, 1994] and Jeffries [Jeffries et al., 1991], classify and make a taxonomy of usability metrics and evaluation methodologies. They provide an overview of different inspection methods to evaluating user interfaces, like Heuristic evaluation, cognitive walkthroughs, usability inspections, among others. The methods involve subjective judgements and are rather informal because thay make use of a domain expert (i.e. the opinion of a human) and concludes that the best results are whenever the domain expert that conducts the evaluation is an specialist in the field.

Some of the papers propose and discuss an evaluation methodology designed with a particular domain in mind, similarly to this thesis, but using a different approach, for example, Biswas [Biswas et al., 2008] designed a methodology for the evaluation of user interfaces in health-related user interfaces, particularly Tele-Physiotherapy that makes use of an extensive checklist of requirements to achieve an acceptable user interface . It still relies on the opinion of an expert, although the criteria set is well defined and unambiguous as on the previously cited papers.

Neto [Neto and da Gracca Pimentel, 2013] extends Nielsen's work by defining a set of heuristics specific for mobile device interfaces, which is presumably where the largest share of ASR technology is going to be applied. His approach is the same, conduct audits by a human expert whose criteria will provide an assessment of the

mobile user interface. He presents a case in which he applies Nielsen's work and his to contrast the difference and show that his work is more effective when assessing mobile interfaces.

Hilbert [Hilbert and Redmiles, 2000] focuses on explaining how to automate the extraction of information about usability by providing an extensive overview of the commonly used computer-aided methods to gather usability metrics, like in our case, the time of interaction on the screens. He also provides a framework to compare the nature of these different information-gathering methods by defining a taxonomy of the different kind of Human-Computer interaction events.

Moving forward in the literature, some authors apply a more formal methodology to asses the usability of the UI. Savioja [Savioja et al., 2008] talks about GUI usability in terms of qualification of the information content and presentation.
They define their own formal framework and criteria to quantify usability the interface and analyze the user activity.

It is worth mentioning one of the foundational theories that this thesis makes use of: The Von Neumann Morgenstern utility theorem [Neumann and Morgenstern, 1947] which lays the axioms of rationality with which the expected utility decision theory is built. This work is extensively used on the thesis to define the notion of policies.

There is evidence that there is work done that uses decision theoretic and probabilistic methods to improve user interface design:

Gajos [Gajos and Weld, 2004] created a framework that leverages the same mathematical tools as this work. The perspective and the scope are completely different. They concerns about the generation of GUIs as an optimization problem to organize the display of widgets optimally, not as an outcome optimization or

interface selection.

Stumpf [Stumpf et al., 2005] implemented a system that tries to achieve a goal similar to the case studied in this thesis by using probabilistic methods to reduce the engagement of users with the UI of a system. They discuss the implementation developed for Microsoft Windows environments and speak on high level about the approach they used, but do not present a formal evaluation of their work neither a strong argument about its validity.

Smyth [Smyth and Cotter, 2002] provide an example of a system with a UI which could be modeled under the framework derived here. It uses probabilistic methods to minimize a cost function ("click distance") in order to enhance utility ("money saved").

Finally, Paek [Paek and Pieraccini, 2008] presents a survey of dialog driven interface design research. Dialog driven interfaces naturally fit into the analysis framework presented in this thesis because under their context, the designers must figure out correctly the best questions to ask given at any moment during the interaction .

The work of this thesis is complementary to other work in the literature. It doesn't try to replace any of above's usability assessments nor to define a new framework that automatically generates better user interfaces. It is a generalization of a methodology to evaluate the application of machine learning models to rational user interfaces. To the knowledge of the author, there is no previous work that tackles this problem explicitly and directly as a decision problem for optimizing a expected utility or cost in a general form.

CHAPTER 5

CONCLUSIONS AND FURTHER DEVELOPMENT

The intention of the thesis is to provide an objective methodology to evaluate the usefulness of adaptive user interfaces. It was developed in such a way that it is abstract enough to be applied to user interfaces in general, not necessarily graphical user interfaces, but simple enough to be easily understood and grounded in real implementations.

With the proposed analysis structure, we can determine if an improvement is achieved by a candidate policy with respect of an empirical policy defined by the usual users' behavior and the degree of this improvement, if any, under a decision theoretic criterion.

The measurements can be used as guidelines by the interaction designers and data analysts to determine the best way to train their models in such a way that suits their needs.

An example of the application was shown in which the framework was born and matched to define a set of candidate policies in a real life ASR application and the results where presented to draw conclusions.

Hopefully, this work can be used on other analysis and prove to be useful. A possible niche in which it could be applied are dialog driven interfaces which could even serve to further refine and extend it, given their dynamic nature.

Given its abstract derivation, the framework could also be applied to analyze the application of machine learning models to any arbitrary decision process, although

no claims about its usefulness are given in this document, it may be an area worth exploring.

REFERENCES

Bach, N., M. Eck, P. Charoenpornsawat, T. Khler, S. Stker, T. Nguyen, R. Hsiao, A. Waibel, S. Vogel, and T. Schultz (2007). The CMU TransTac 2007 eyes-free and hands-free two-way speech-to-speech translation system. In *Proceedings of the IWSLT*, volume 7.

Biswas, J., V. Foo, M. A. Feki, L. S. Yee, and P. Yap (2008). Usability Evaluation of an Innovative Platform for Tele-physiotherapy. In *Proceedings of the 2Nd International Convention on Rehabilitation Engineering & Assistive Technology*, iCREATe '08, pp. 197–200. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, Kaki Bukit TechPark II,, Singapore.

Gajos, K. and D. S. Weld (2004). SUPPLE: Automatically Generating User Interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, pp. 93–100. ACM, New York, NY, USA. ISBN 1-58113-815-6.

Hilbert, D. M. and D. F. Redmiles (2000). Extracting Usability Information from User Interface Events. *ACM Comput.Surv.*, **32**(4), pp. 384–421.

Iii, H. D., J. Langford, and D. Marcu (2009). Search-based structured prediction. *Machine Learning*, **75**(3), pp. 297–325.

Jeffries, R., J. R. Miller, C. Wharton, and K. Uyeda (1991). User Interface Evaluation in the Real World: A Comparison of Four Techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pp. 119–124. ACM, New York, NY, USA. ISBN 0-89791-383-3.

Neto, O. M. and M. da Gracca Pimentel (2013). Heuristics for the Assessment of Interfaces of Mobile Devices. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*, WebMedia '13, pp. 93–96. ACM, New York, NY, USA. ISBN 978-1-4503-2559-2.

Neumann, L. J. and O. Morgenstern (1947). *Theory of games and economic behavior*, volume 60. Princeton university press Princeton, NJ.

Nielsen, J. (1994). Usability Inspection Methods. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, pp. 413–414. ACM, New York, NY, USA. ISBN 0-89791-651-4.

Paek, T. and R. Pieraccini (2008). Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communication*, **50**(89), pp. 716–729.

Ross, S., G. J. Gordon, and J. A. Bagnell (2010). A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*.

Savioja, P., L. Norros, and L. Salo (2008). Evaluation of Systems Usability. In *Proceedings of the 15th European Conference on Cognitive Ergonomics: The Ergonomics of Cool Interaction*, ECCE '08, pp. 26:1–26:8. ACM, New York, NY, USA. ISBN 978-1-60558-399-0.

Smyth, B. and P. Cotter (2002). Personalized adaptive navigation for mobile portals. In *ECAI*, pp. 608–612.

Stumpf, S., X. Bao, A. Dragunov, T. G. Dietterich, J. Herlocker, K. Johnsrude, L. Li, and J. Shen (2005). Predicting user tasks: I know what you're doing. In *20th National Conference on Artificial Intelligence (AAAI-05), Workshop on Human Comprehensible Machine Learning*.