# Prediction, detection, and correction of misunderstandings in interactive tasks

Dissertation
zur Erlangung des akademischen Grades
eines Doktors der Philosophie
der Philosophischen Fakultät
der Universität des Saarlandes

vorgelegt von
Martín Villalba
aus Posadas, Argentinische Republik

Saarbrücken, 2019

| | |
|---|---|
| *Der Dekan:* | Univ.-Prof. Dr. Heinrich Schlange-Schöningen |
| *Berichterstatter/innen:* | Prof. Dr. Alexander Koller |
| | Prof. Dr. Luciana Benotti |
| *Tag der letzten Prüfungsleistung:* | 16. August 2019 |

I'd like to thank Luciana Benotti for her help kickstarting my PhD,
and Alexander Koller for his patience guiding me throughout it.
I'd also like to thank my research group for making it so much fun.
Antoine, Christoph, Jonas, Meaghan, and Nikos,
this wouldn't have been possible without you.

I'd also like to dedicate this thesis to my family, who have supported
me at every step despite secretly hoping I'd just go back home.
In that respect, I'd like to thank my drawing group, Potsdam
researchers (specially PRIM and Manfred's students), my Iaido group,
and other Saarbrücken researchers for making it worth the stay.

Finally, special thanks to Manja for her support along the way,
without whom I wouldn't have made it this far.

**Abstract**

Technology has allowed all kinds of devices and software to come into our lives. Advances in GPS, Virtual Reality, and wearable computers with increased computing power and Internet connectivity open the doors for interactive systems that were considered science fiction less than a decade ago, and are capable of guiding us in a variety of environments.

This increased accessibility comes at the cost of increasing both the scale of problems that can be realistically tackled and the capabilities that we expect from such systems. Indoor navigation is an example of such a task: although guiding a car is a solved problem, guiding humans for instance inside a museum is much more challenging. Unlike cars, pedestrians use landmarks rather than absolute distances. They must discriminate from a larger number of distractors, and expect sentences of higher complexity than those appropriate for a car driver. A car driver prefers short, simple instructions that do not distract them from traffic. A tourist inside a museum on the contrary can afford the mental effort that a detailed grounding process would require.

Both car and indoor navigation are specific examples of a wider family of collaborative tasks known as "Instruction Following". In these tasks, agents with the two clearly defined roles of *Instruction Giver* and *Instruction Follower* must cooperate to achieve a joint objective. The former has access to all required information about the environment, including (but not limited to) a detailed map of the environment, a clear list of objectives, and a profound understanding of the effect that specific actions have in the environment. The latter is tasked with following the instructions, interacting with the environment and moving the undertaking forward. It is then the Instruction Giver's responsibility to assess a detailed plan of action, segment it into smaller subgoals, and present instructions to the Instruction Follower in a language that is clear and understandable.

No matter how carefully crafted the Instruction Giver's utterances are, it is expected that misunderstandings will take place. Although some of these misunderstandings are easy to detect and repair, others can be very difficult or even impossible to solve. It is therefore important for the Instruction Giver to generate instructions that are as clear as possible, to detect misunderstandings as early as possible, and to correct them in the most effective way. This thesis introduces several algorithms and strategies designed to tackle the aforementioned problems from end to end, presenting the individual aspects of a system that successfully predicts, detects, and corrects misunderstandings in interactive Instruction Following tasks.

We focus on one particular type of instruction: those involving *Referring Expressions*. A Referring Expression identifies a single object out of

many, such as "the red button" or "the tall plant". Generating Referring Expressions is a key component of Inst. Following tasks, since any kind of object manipulation is likely to require a description of the object. Due to its importance and complexity, this is one of the most widely studied areas of Natural Language Generation. In this thesis we use *Semantically Interpreted Grammars*, an approach that integrates both *Referring Expression Generation* (identifying which properties are required for a unique description) and *Surface realization* (combining those properties into a concrete Noun Phrase).

The complexity of performing, recording, and analyzing Instruction Following tasks in the real world is one of the major challenges of Instruction Following research. In order to simplify both the development of new algorithms and the access to those results by the research community, our work is evaluated in what we call a *Virtual Environment* — an environment that mimics the main aspects of the real world and abstracts distractions while preserving enough characteristics of the real world to be useful for research. Selecting the appropriate virtual environment for a research task ensures that results will be applicable in the real world. We have selected the Virtual Environment of the *GIVE Challenge*, an environment designed for an Instruction Following task in which a human Instruction Follower is paired with an automated Instruction Giver in a maze-like 3D world. Completing the task requires navigating the space, avoiding alarms, interacting with objects, generating instructions in Natural Language, and preventing mistakes that can bring the task to a premature end. Even under these simplified conditions, the task presents several computational challenges: performing these tasks in real time require fast algorithms, and ensuring the efficiency of our approaches remains a priority at every step.

Our first experimental study identifies the most challenging type of mistakes that our system is expected to find. Creating an Inst. Following system that leverages previously-recorded human data and follows instructions using a simple greedy algorithm, we clearly separate those situations for which no further study is warranted from those that are of interest for our research. We test our algorithm with similarity metrics of varying complexity, ranging from overlap measures such as *Jaccard* and edit distances to advanced machine learning algorithms such as *Support Vector Machines*. The best performing algorithms achieve not only good accuracy, but we show in fact that mistakes are highly correlated with situations that are also challenging for human annotators. Going a step further, we also study the type of improvement that can be expected from our system if we give it the chance of retrying after a mistake was made. This system has no prior beliefs on which actions are more likely to be selected next, and our results make a good case for this vision to be one of its weakest points. Moving away from a paradigm where all actions are considered equally likely, and moving towards a model in which the Inst. Follower's

own action is taken into account, our subsequent step is the development of a system that explicitly models listener's understanding.

Given an instruction containing a Referring Expression, we approach the Instruction Follower's understanding of it with a combination of two probabilistic models. The *Semantic model* uses features of the Referring Expression to identify which object is more likely to be selected: if the instruction mentions a red button, it is unlikely that the Inst. Follower will select a blue one. The *Observational model*, on the other hand, predicts which object will be selected by the Inst. Follower based on their behavior: if the user is walking straight towards a specific object, it is very likely that this object will be selected. These two *log-linear, probabilistic* models were trained with recorded human data from the GIVE Challenge, resulting in a model that can effectively predict that a misunderstanding is about to take place several seconds before it actually happens. Using our *Combined model*, we can easily detect and predict misunderstandings — if the Inst. Giver tells the Inst. Follower to "click the red button", and the Combined model detects that the Inst. Follower will select a blue one, we know that a misunderstanding took place, we know what the misunderstood object is, and we know both facts early enough to generate a correction that will stop the Inst. Follower from making the mistake in the first place.

A follow-up study extends the Observational model introducing features based on the gaze of the Inst. Follower. Gaze has been shown to correlate with human attention, and our study explores whether gaze-based features can improve the accuracy of the Observational model. Using previously-collected data from the GIVE Environment in which gaze was recorded using eye-tracking equipment, the resulting *Extended Observational model* improves the accuracy of predictions in challenging scenes where the number of distractors is high.

Having a reliable method for the detection of misunderstandings, we turn our attention towards corrections. A *corrective* Referring Expression is one designed not only for the identification of a single object out of many, but rather, for identifying a previously-wrongly-identified object. The simplest possible corrective Referring Expression is repetition: if the user misunderstood the expression "the red button" the first time, it is possible that they will understand it correctly the second time.

A smarter approach, however, is to reformulate the Referring Expression in a way that makes it easier for the Inst. Follower to understand. We designed and evaluated two different strategies for the generation of corrective feedback. The first of these strategies exploits the pragmatics concept of a *Context Set*, according to which human attention can be segmented into objects that are being attended to (that is, those inside the Context Set) and those that are ignored. According to our theory, we could virtually ignore all objects outside the Context Set and generate Referring Expressions that would not be uniquely identifying with respect to the entire context,

but would still be identifying enough for the Inst. Follower. As an example, if the user is undecided between a red button and a blue one, we could generate the Referring Expression "the red one" *even if there are other red buttons on the scene that the user is not paying attention to*. Using our probabilistic models as a measure for which elements to include in the Context Set, we modified our Referring Expression Generation algorithm to build sentences that explicitly account for this behavior. We performed experiments over the GIVE Challenge Virtual Environment, crowdsourcing the data collection process, with mixed results: even if our definition of a Context Set were correct (a point that our results can neither confirm nor deny), our strategy generates Referring Expressions that prevents *some* mistakes, but are in general *harder* to understand than the baseline approach. The results are presented along with an extensive error analysis of the algorithm. They imply that corrections can cause the Instruction Follower to re-evaluate the entire situation in a new light, making our previous definition of Context Set impractical. Our approach also fails at identifying previously grounded referents, compounding the number of pragmatic effects that conspire against this approach.

The second strategy for corrective feedback consists on adding *Contrastive focus* to a second, corrective Referring Expression In a scenario in which the user receives the Referring Expression "the red button" and yet mistakenly selects a blue one, an approach with contrastive focus would generate "no, the **RED** button" as a correction. Such a Referring Expression makes it clear to the Inst. Follower that on the one hand their selection of an object of type "button" was correct, and that on the other hand it is the property "color" that needs re-evaluation. In our approach, we model a misunderstanding as a noisy channel corruption: the Inst. Giver generates a correct Referring Expression for a given object, but it is corrupted in transit and reaches the Inst. Follower in the form of an altered, incorrect Referring Expression We correct this misconstrual by generating a new, corrective Referring Expression: starting from the original Referring Expression and the misunderstood object, we identify the constituents of the Referring Expression that were corrupted and place contrastive focus on them. Our hypothesis states that the minimum edit sequence between the original and misunderstood Referring Expression correctly identifies the constituents requiring contrastive focus, a claim that we verify experimentally.

We perform crowdsourced preference tests over several variations of this idea, evaluating Referring Expressions that either present contrast side by side (as in "no, not the **BLUE** button, the **RED** button") or attempt to remove redundant information (as in "no, the **RED** one"). We evaluate our approaches using both simple scenes from the GIVE Challenge and more complicated ones showing pictures from the more challenging TUNA people corpus. Our results show that human users significantly prefer our

most straightforward contrastive algorithm.

In addition to detailing models and strategies for misunderstanding detection and correction, this thesis also includes practical considerations that must be taken into account when dealing with similar tasks to those discussed here. We pay special attention to *Crowdsourcing*, a practice in which data about tasks can be collected from participants all over the world at a lower cost than traditional alternatives. Researchers interested in using crowdsourced data must often deal both with unmotivated players and with players whose main motivation is to complete as many tasks as possible in the least amount of time. Designing a crowdsourced experiment requires a multifaceted approach: the task must be designed in such a way as to motivate honest players, discourage other players from cheating, implementing technical measures to detect bad data, and prevent undesired behavior looking at the entire pipeline with a *Security mindset*. We dedicate a Chapter to this issue, presenting a full example that will undoubtedly be of help for future research.

We also include sections dedicated to the theory behind our implementations. Background literature includes the pragmatics of dialogue, misunderstandings, and focus, the link between gaze and visual attention, the evolution of approaches towards Referring Expression Generation, and reports on the motivations of crowdsourced workers that borrow from fields such as psychology and economics. This background contextualizes our methods and results with respect to wider fields of study, enabling us to explain not only that our methods *work* but also *why* they work.

We finish our work with a brief overview of future areas of study. Research on the prediction, detection, and correction of misunderstandings for a multitude of environments is already underway. With the introduction of more advanced virtual environments, modern spoken, dialogue-based tools revolutionizing the market of home devices, and computing power and data being easily available, we expect that the results presented here will prove useful for researchers in several areas of Natural Language Processing for many years to come.

**Zusammenfassung**

Die Technologie hat alle möglichen Arten von unterstützenden Geräten und Softwares in unsere Leben geführt. Fortschritte in GPS, Virtueller Realität, und tragbaren Computern mit wachsender Rechenkraft und Internetverbindung öffnen die Türen für interaktive Systeme, die vor weniger als einem Jahrzehnt als *Science Fiction* galten, und die in der Lage sind, uns in einer Vielfalt von Umgebungen anzuleiten. Diese gesteigerte Zugänglichkeit kommt zulasten sowohl des Umfangs der Probleme, die realistisch gelöst werden können, als auch der Leistungsfähigkeit, die wir von solchen Systemen erwarten. Innennavigation ist ein Beispiel einer solcher Aufgaben: obwohl Autonavigation ein gelöstes Problem ist, ist das Anleiten von Meschen zum Beispiel in einem Museum eine größere Herausforderung. Anders als Autos, nutzen Fußgänger eher Orientierungspunkte als absolute Distanzen. Sie müssen von einer größeren Anzahl von Ablenkungen unterscheiden können und Sätze höherer Komplexität erwarten, als die, die für Autofahrer angebracht sind. Ein Autofahrer bevorzugt kurze, einfache Instruktionen, die ihn nicht vom Verkehr ablenken. Ein Tourist in einem Museum dagegen kann die metale Leistung erbringen, die ein detaillierter Fundierungsprozess benötigt. Sowohl Auto- als auch Innennavigation sind spezifische Beispiele einer größeren Familie von kollaborativen Aufgaben bekannt als *Instruction Following*. In diesen Aufgaben müssen die zwei klar definierten Akteure des *Instruction Givers* und des *Instruction Followers* zusammen arbeiten, um ein gemeinsames Ziel zu erreichen. Der erstere hat Zugang zu allen benötigten Informationen über die Umgebung, inklusive (aber nicht begrenzt auf) einer detallierten Karte der Umgebung, einer klaren Liste von Zielen und einem genauen Verständnis von Effekten, die spezifische Handlungen in dieser Umgebung haben. Der letztere ist beauftragt, den Instruktionen zu folgen, mit der Umgebung zu interagieren und die Aufgabe voranzubringen. Es ist dann die Verantwortung des *Instruction Giver*, einen detaillierten Handlungsplan auszuarbeiten, ihn in kleinere Unterziele zu unterteilen und die Instruktionen dem *Instruction Follower* in einer klaren, verständlichen Sprache darzulegen. Egal wie sorgfältig die Äußerungen des *Instruction Givers* erarbeitet sind, ist es zu erwarten, dass Missverständnisse stattfinden. Obwohl einige dieser Missverständnisse einfach festzustellen und zu beheben sind, können anderen sehr schwierig oder gar unmöglich zu lösen sein. Daher ist es wichtig, dass der Instruction Giver die Anweisungen so klar wie möglich formuliert, um Missverständnisse so früh wie möglich aufzudecken, und sie in der effektivsten Weise zu berichtigen.

Diese Thesis führt mehrere Algorithmen und Strategien ein, die dazu entworfen wurden, die oben genannten Probleme in einem *End-to-End* Pro-

zess zu lösen. Dabei werden die individuellen Aspekte eines Systems präsentiert, dass erfolgreich Missverständnisse in interaktiven *Instruction Following* Aufgaben vorhersagen, feststellen und korrigieren kann. Wir richten unsere Aufmerksamkeit auf eine bestimmte Art von Instruktion: die sogennanten *Referring Expressions*. Eine *Referring Expression* idenfiziert ein einzelnes Objekt aus vielen, wie zum Beispiel „der rote Knopf" oder „die große Pflanze". Das Generieren von *Referring Expressions* ist eine Schlüsselkomponente von *Instruction Following* Aufgaben, da jegliche Art von Manipulation sehr wahrscheinlich eine Beschreibung des Objektes erfordert. Wegen der Wichtigkeit und Komplexität ist dies eine der am meisten untersuchten Gebiete der Textgenerierung. In dieser Thesis verwenden wir *Semantisch Interpretierte Grammatik*, eine Methode, die sowohl die Generierung von *Referring Expressions* (Identifizierung von Eigenschaften für eine eindeutige Beschreibung) als auch *Surface Realization* (Kombinieren dieser Eigenschaften in eine konkrete Substantivgruppe) integriert. Die Komplexität der Durchführung, Aufzeichnung und Analyse von *Instruction Following* Aufgaben in der realen Welt ist eine der großen Herausforderungen der *Instruction Following* Forschung. Um sowohl die Entwicklung neuer Algorithmen und den Zugang zu diesen Ergebnissen durch die Wissenschaftsgemeinde zu vereinfachen, wird unsere Arbeit in einer *Virtuellen Umgebung* bewertet. Eine virtuelle Umgebung ahmt die Hauptaspekte der realen Welt nach und nimmt Ablenkungen weg, während genug Eigenschaften der realen Welt erhalten bleiben, um verwendbar für die Untersuchung zu sein. Die Auswahl der angebrachten virtuellen Umgebung für eine Forschungsaufgabe gewährleistet, dass die Ergebnisse auch in der realen Welt anwendbar sind. Wir haben eine virtuelle Umgebung der *GIVE Challenge* ausgesucht âĂŞ eine Umgebung, die für eine *Instruction Following Aufgabe entworfen wurde, in der ein menschlicher Instruction Follower* mit einem automatischen *Instruction Giver* in einer Labyrinth-artigen 3D Welt verbunden wird. Die Aufgabe zu beenden erfordert Navigation im Raum, Vermeidung von Alarmen, Interagieren mit Objekten, Textgenerierung und Verhindern von Fehlern, die zu einer vorzeitigen Beendung der Aufgabe führen. Sogar unter diesen vereinfachten Bedingungen stellt die Aufgabe mehrere rechentechnische Herausforderungen dar: die Aufgabe in Echtzeit durchzuführen erfordert schnelle Algorithmen, und die Effizienz unserer Methode zu gewährleisten bleibt Priorotät in jedem Schritt.

Unser erstes Experiment identifiziert die herausfordernste Art von Fehlern, die unser System erwartungsgemäß finden soll. Durch den Entwurf eines *Instruction Following* Systems, das sich zuvor aufgezeichnete menschliche Daten zu Nutze macht und durch die Nutzung eines einfachen *gierigen Algorithmus* Intruktionen folgt, grenzen wir klar die Situationen ab, die keine weitere Studie rechtfertigen, von denen, die interessant für unsere Forschung sind. Wir testen unseren Algorithmus mit Ähnlichkeitsmaßen verschiedener Komplexität, die sich von Überlappungsmaßnahmen

wie Jaccard und Editierdistanzen, bis zu fortgeschrittenen Algorithmen des Maschinellen Lernens erstrecken. Die am besten ausführenden Algorithmen erreichen nicht nur gute Genauigkeit sondern tatsächlich zeigen wir, dass Fehler hoch korreliert sind mit Situationen, die auch herausfordernd für menschliche Kommentatoren sind. In einem weiteren Schritt untersuchen wir die Art von Verbesserung, die von unserem System erwartet werden kann wenn wir ihm die Chance geben, es wieder zu versuchen nachdem ein Fehler gemacht wurde. Dieses System macht keine vorherigen Annahmen darüber, welche Aktionen am wahrscheinlichsten als nächstes ausgewählt werden und unsere Ergebnisse liefern gute Argumente dafür, dass dieser Ansatz einer der schwächsten Aspekte ist. Um sich von einem Paradigma wegzubewegen, in dem alle Aktionen gleich wahrscheinlich betrachtet werden, zu einem Model, in dem das Handeln des *Instruction Followers* in Betracht gezogen wird, ist unser folgender Schritt die Entwicklung eines Systems, dass explizit das Verständnis des Anwenders modelliert. Voraussetzend, dass die Instruktion eine *Referring Expression* beinhaltet, gehen wir das Verstehen des *Instruction Followers* mit einer Kombination aus zwei probabilistischen Modellen an. Das semantische Modell verwendet Eigenschaften der *Referring Expression* um zu identifizieren, welches Objekt wahrscheinlicher ausgewählt wird: wenn die Instruktion einen roten Knopf benennt, ist es unwahrscheinlich, dass der *Instruction Follower* den blauen wählt. Das Beobachtungsmodell dagegen sagt vorher, welches Objekt von dem *Instruction Follower* basierend auf seinem Verhalten gewählt wird: wenn der Anwender direkt auf ein spezifisches Objekt zuläuft, ist es sehr wahrscheinlich, dass er dieses auswählt. Diese zwei log-linearen, probabilistischen Modelle wurden mit aufgezeichneten menschlichen Daten von der *GIVE Challenge* trainiert. Daraus resultiert ein Modell, welches effektiv vorhersagen kann, dass ein Missverständnis geschehen wird, mehrere Sekunden bevor es tatsächlich passiert. Durch die Nutzung unseres Kombinierten Modells können wir leicht Missverständnisse feststellen und vorhersagen: wenn der *Instruction Giver* zu dem *Instruction Follower* sagt „betätige den roten Knopf", und das Kombinierte Modell feststellt, dass der *Instruction Follower* den blauen wählen will, wissen wir, dass ein Missverständnis stattgefunden hat. Zusätzlich wissen wir beide Fakten früh genug, um eine Korrektur vorzunehmen, die den *Instruction Follower* davon abhält, den Fehler in erster Linie zu begehen. Eine Folgestudie erweitert das Beobachtungsmodell, indem Eigenschaften des *Instruction Followers* basierend auf dessen Blickrichtung eingeführt werden. Die Blickrichtung korreliert mit der Aufmerksamtkeit von Menschen, und unsere Studie untersucht, ob Blick-basierte Eigenschaften die Genauigkeit des Beobachtungsmodells verbessern können. Durch die Nutzung vorherig gesammelter Daten aus der *GIVE* Umgebung, in welcher die Blickrichtung mithilfe einer Augenverfolgungsausstattung aufgezeichnet wurde, verbessert das resultierende Erweiterte Beobachtungsmodell die Genauigkeit der Vorhersagen

in herausfordernden Vorgängen, in der die Anzahl der Ablenkungen hoch ist.

Durch den Besitz einer verlässlichen Methode für die Feststellung von Missverständnissen wenden wir unsere Aufmerksamkeit in Richtung Korrekturen. Eine korrigierende *Referring Expression* ist nicht nur für die Identifikation eines einzelnes Objekts aus vielen entworfen, sondern für die Feststellung der zuvor falschen Identifikation eines Objekts. Die einfachst mögliche korrigierende *Referring Expression* ist die Wiederholung: wenn der Anwender die Aussage „der rote Knopf" das erste Mal falsch verstanden hat, ist es möglich, dass er es beim zweiten Mal richtig versteht. Eine klügere Methode ist, die *Referring Expression* so umzuformulieren, dass es für den *Instruction Follower* leichter ist, sie zu verstehen. Wir entwarfen und evaluierten zwei verschiedene Strategien für die Generierung von korrigierender Rückmeldung. Die erste dieser Strategien nutzt das pragmatische Konzept eines *Context Sets*, nachdem menschliche Aufmerksamkeit in Objekte unterteilt wird, denen Aufmerksamkeit zukommt (innerhalb des *Context Sets*) und denen, die ignoriert werden. Nach unserer Theorie könnten wir nahezu alle Objekte außerhalb des *Context Sets* ignorieren und *Referring Expressions* generieren, die nicht eindeutig mit Bezug auf den gesamten Kontext identifizieren, aber genug für den *Instruction Follower*. Zum Beispiel, wenn der Anwender unentschieden ist zwischen einem roten und einem blauen Knopf, könnten wir die *Referring Expression* „der Rote" generieren, sogar wenn es noch andere rote Knöpfe in der Szene gibt, auf die der Anwender nicht achtet. Durch die Nutzung unseres probabilistischen Modells als Maß dafür, welches Element in das *Context Set* einbezogen wird, modifizieren wir den Generierungsalgorithmus der *Referring Expression*, um Sätze zu bilden, die explizit dieses Verhalten einbeziehen. Wir haben Experimente in der *GIVE Challenge* virtuellen Umgebung, mit einem *Crowdsource* Datensammlungsprozess durchgeführt, mit gemischten Ergebnissen: sogar wenn unsere Definition von Context Set richtig gewesen wäre (ein Punkt, den unsere Ergebnisse weder bestätigen noch widerlegen), generiert unsere Strategie *Referring Expressions*, die einige Fehler verhindern aber generell schwieriger zu verstehen sind als der Basisansatz. Die Ergebnisse gehen einher mit einer umfangreichen Fehleranalyse des Algorithmus'; sie legen nah, dass Korrekturen den *Instruction Follower* dazu veranlassen, die gesamte Situation in einem neuen Licht zu bewerten, wodurch unsere vorherige Definition von *Context Set* unbrauchbar wird. Unsere Methode ist weiterhin nicht in der Lage, bereits zuvor verstandene Objekte zu identifizieren, wodurch die Anzahl der pragmatischen Effekte steigt, die sich gegen diese Methode stellen. Die zweite Strategie für eine korrigierende Rückmeldung besteht darin, einen Gegensatzfokus zu einer zweiten korrigierenden *Referring Expression* hinzuzufügen. In einem Szenario, in dem der Anwender die *Referring Expression* „der rote Knopf" erhält und trotzdem fälschlicherweise den blauen wählt, würde ein Ansatz mit Gegensatzfokus

„nein, der **ROTE** Knopf" als Korrektur generieren. Solch eine *Referring Expression* macht dem *Instruction Follower* klar, dass einerseits die Auswahl eines Objekts des Typ „Knopf" richtig war, und andererseits die Eigenschaft „Farbe" eine Neubewertung benötigt.

In unserer Methode modellieren wir ein Missverständnis mit einer noisy channel Verfälschung: der *Instruction Giver* generiert eine korrekte *Referring Expression* für ein bestimmtes Objekt aber es wird bei der Übertragung verfälscht und erreicht den *Instruction Follower* in der Form einer veränderten, inkorrekten *Referring Expression*. Wir korrigieren die Fehlinterpretation durch die Generierung einer neuen, korrigierenden Referring Expression: ausgehend von der originalen *Referring Expression* und dem missverstandenem Objekt, identifizieren wir die Bestandteile der *Referring Expression*, die verfälscht wurden und legen einen gegensätzlichen Fokus auf sie. Unsere Hypothese besagt, dass die minimale Überarbeitungssequenz zwischen der originalen und der missverstandenen *Referring Expression* die Bestandteile korrekt identifiziert, die einen gegesätzlichen Fokus erfordern - eine Behauptung, die wir experimentell bestätigen. Wir führen *Crowdsource* Tests mit verschiedenen Varianten dieser Idee durch, wobei wir *Referring Expressions* evaluieren, die entweder einen Kontrast darstellen (z.B. „nein, nicht der **BLAUE** Knopf, der **ROTE** Knopf") oder versuchen, die überflüssige Information zu entfernen (z.B. „nein, der **ROTE**"). Wir evaluieren unsere Ansätze durch die Nutzung von sowohl einfachen Szenen der *GIVE Challenge* als auch komplizierteren Bildern aus dem mehr herausfordernden *TUNA people corpus*. Unsere Ergebnisse zeigen, dass menschliche Anwender unseren unkompliziertesten gegensätzlichen Algorithmus signifikant bevorzugen.

Zusätzlich zu den detaillierten Modellen und Strategien zur Missverständnisfestellung und -korrektur, beinhaltet diese Thesis praktische Hinweise, die in Betracht gezogen werden müssen wenn man mit Aufgaben zu tun hat, die ähnlich zu den hier diskutierten sind. Wir richten besonderes Augenmerk auf *Crowdsourcing*, eine Praxis, bei der durch Aufgaben Daten von Teilnehmern aus aller Welt zu niedrigeren Kosten als durch traditionelle Alternativen gesammelt werden können. Wissenschaftler, die daran interessiert sind, *Crowdsource* Daten zu verwenden, müssen oft sowohl mit unmotivierten Spielern umgehen, als auch mit Spielern, deren Hauptmotivation es ist, so viele Aufgaben in so kurzer Zeit wie möglich zu lösen. Ein *Crowdsource* Experiment zu entwerfen erfordert eine vielseitige Methodik: die Aufgabe muss so entwickelt sein, dass sie einen ehrlichen Spieler motiviert, andere Spieler vom Betrügen abhält, technische MaÎše zum Feststellen schlechter Daten implementiert, und unerwünschtes Verhalten verhindert mit Blick auf die gesamte Pipeline und einem „Sicherheit zuerst"-Ansatz. Wir widmen ein ganzes Kapitel dieser Problematik und präsentieren ein komplettes Beispiel, das zweifelsfrei hilfreich für künftige Forschung sein wird.

Weiterhin fügen wir Sektionen ein, die sich der Theorie hinter unseren Anwendungen widmen. Hintergrundliteratur beinhaltet die Pragmatik von Dialogen, Missverständnissen und Fokus, die Verbindung zwischen Blickrichtung und visueller Aufmerksamkeit und der Evolution von Methoden für die Generierung von *Referring Expressions*. Zusätzlich präsentieren wir Berichte, die die Motivation von *Crowdsource* Arbeitern untersuchen, sowohl aus einer theoretischen als auch aus einer experimentellen Perspektive âĂŞ die erstere durch Aufgreifen von Konzepten aus der Psychologie und Wirtschaft, und die zweitere durch Experimente, die zum Testen der Ehrlichkeit des Arbeiters entworfen wurden. Dieser Hintergrund bringt unsere Methoden und Ergebnisse in Verbindung mit Bezug zu breitgefächerten Fachgebieten, die uns ermöglichen nicht nur zu erklären, dass unsere Methoden funktionieren sondern auch warum sie funktionieren. Wir beenden unsere Arbeit mit einem kurzen Überblick über künftige Forschungsbereiche. Forschung zur Vorhersage, Feststellung und Korrektur von Missverständnissen für eine Vielzahl von Umwelten ist bereits unterwegs. Mit der Einführung von fortgeschrittenen virtuellen Umgebungen, modern ausgedrückt, Dialog-basierte Werkzeuge, die den Markt von Heimgeräten, Rechenkraft und vereinfachtem Datenzugriff revolutionieren, erwarten wir, dass die hier präsentierten Ergebnisse sich für Forscher in mehreren Bereichen der natürlichen Sprachverarbeitung auf Jahre hinaus als nützlich erweisen werden.

# Contents



**Abstract** **v**

**Zusammenfassung** **xi**

**Table of contents** **xix**

**List of Figures** **xxii**

**List of Abbreviations** **xxiii**

**1 Introduction** **1**
1.1 Challenges . . . 3
1.1.1 Challenges in developing for the real world . . . 4
1.1.2 Challenges in detection of misunderstandings . . . 5
1.1.3 Challenges in correcting misunderstandings . . . 6
1.1.4 Published research . . . 7
1.2 Summary . . . 8

**2 Instruction following in Virtual Environments** **11**
2.1 Early work on Instruction Following . . . 12
2.2 Automated planning . . . 14
2.3 Natural Language Generation . . . 15
2.3.1 Referring Expression Generation . . . 18
2.3.2 Interpreted Regular Tree Grammars . . . 20
2.4 Virtual Environments . . . 26
2.5 The GIVE Challenge . . . 30
2.6 The pragmatics of dialogue, misunderstandings, and focus . 32
2.6.1 Misunderstandings . . . 33
2.6.2 Focus . . . 34
2.7 Conclusion . . . 34
2.8 Further reading . . . 35

**3 Following instructions** **37**
3.1 A simple strategy for instruction following . . . 38
3.2 Implementing our IF . . . 39
3.2.1 Data collection and segmentation . . . 39
3.2.2 Interpretation . . . 41
3.2.3 Group selection by word similarity . . . 42
3.2.4 Group selection with machine translation methods . 43
3.2.5 Group selection with machine learning . . . 43
3.2.6 Corrections . . . 44
3.3 Experiments and Results . . . 45
3.4 Lessons learned . . . 47

3.5 Conclusion . . . . . 47

**4 Detecting misunderstandings 49**
4.1 Definitions . . . . . 50
4.2 A model of listener's understanding . . . . . 51
4.2.1 The Principle of Maximum Entropy and log-linear models . . . . . 52
4.2.2 Feature functions for $P_{Sem}$ . . . . . 53
4.2.3 Feature functions for $P_{Obs}$ . . . . . 55
4.3 Experimental setup and evaluation . . . . . 58
4.3.1 Prediction accuracy . . . . . 58
4.3.2 Feedback appropriateness . . . . . 59
4.4 Conclusion . . . . . 60

**5 Tracking attention 63**
5.1 Attention and Visual Attention . . . . . 64
5.1.1 Visual Saliency . . . . . 66
5.1.2 Eye-tracking . . . . . 67
5.2 Eye-tracking and Extended Probabilistic model . . . . . 68
5.2.1 New features . . . . . 69
5.2.2 Time to interaction . . . . . 70
5.3 Experimental setup and evaluation . . . . . 70
5.3.1 Results . . . . . 71
5.4 Conclusion . . . . . 72

**6 Crowdsourcing and cheating detection 75**
6.1 Cheating . . . . . 76
6.1.1 The security mindset . . . . . 77
6.1.2 Related work . . . . . 78
6.2 Pipeline . . . . . 79
6.2.1 The GIVE Matchmaker . . . . . 80
6.2.2 The GIVE Unity client . . . . . 80
6.2.3 The GIVE Automated IG . . . . . 81
6.2.4 The Crowdflower interface . . . . . 84
6.3 Quality control . . . . . 85
6.3.1 Payment scale . . . . . 86
6.3.2 Secret words . . . . . 86
6.3.3 Results . . . . . 87
6.4 Conclusion . . . . . 88

**7 Correcting misunderstandings: reformulation** **91**
7.1 Attention and Context Set . . . . . 92
7.2 Generating with a Context Set . . . . . 93
7.3 Experimental setup . . . . . 96
7.3.1 Strategies for feedback generation . . . . . 96
7.3.2 Results . . . . . 97
7.3.3 Error analysis . . . . . 98
7.4 Conclusion . . . . . 102

**8 Correcting misunderstandings: Contrastive Referring Expressions** **103**
8.1 Contrastive focus . . . . . 104
8.2 A minimum-distance approach to contrast . . . . . 106
8.2.1 Finding the missing RE . . . . . 107
8.3 Generation of contrastive feedback . . . . . 110
8.4 Experimental setup . . . . . 111
8.4.1 Experiment results . . . . . 113
8.4.2 Experiment 2 . . . . . 113
8.5 Discussion . . . . . 114
8.6 Conclusion . . . . . 115

**9 Conclusion** **117**
9.1 Summary . . . . . 118
9.2 Future work . . . . . 119

**Bibliography** **133**

# List of Figures




2.1 Example of all steps involved in a weather report system. a) Content determination. b) Text structuring. c) Sentence aggregation. d) Lexicalization. e) Referring Expression Generation. f) Linguistic realization. . . . . 17
2.2 Annotated example of a SIG . . . . . 22
2.3 World model for the example . . . . . 23
2.4 Example derivation tree (center) with the string interpretations of every subtree on the left and the relational interpretations of every subtree on the right. . . . . 23
2.5 Chart containing derivations for the object $b_1$. This chart is based on a simplified grammar that does not include [above/below]-of relations between objects. . . . . 24
2.6 Visual domain of the GRE3D corpus and its corresponding attributes. *bef*="before", *beh*="behind", *nt*="next to". . . . . 28
2.7 Images and attributes from the TUNA corpus domains . . . 28
2.8 SCARE Corpus - First-Person view and upstairs floor map . 30
2.9 First-person and Map view of a GIVE world from the 2011 evaluation worlds (Striegnitz et al., 2011). . . . . 31

3.1 Accuracy values as a function of the number of corrections . 46

4.1 Anatomy of an episode . . . . . 51
4.2 Prediction accuracy as a function of time . . . . . 59
4.3 Feedback F1 measure as a function of time . . . . . 60

5.1 Input images and their corresponding saliency maps . . . . 67
5.2 Recording user data with an eye-tracker. The image shows the eye-tracker equipment on the bottom right, the player's point of view, and circles (not shown to the player) indicating the recorded user's gaze on the current scene. . . . . 68
5.3 Gaze cursor and fixated targets in the GIVE Challenge . . . . 69
5.4 Accuracy as a function of training and testing time . . . . . 72

6.1 Crowdsourcing pipeline . . . . . 79
6.2 Network connectivity example. Computers inside a private network can connect with each other, but communication outside must go through a router. Public computers cannot contact computers inside the private network directly. . . . . 82
6.3 Network connectivity for our pipeline. The GIVE Matchmaker is on the public internet, while the GIVE Automated IG resides in a private network. Port forwarding is requires to allow communication between them. . . . . 83
6.4 Web interface of the cf-thesis project . . . . . 86

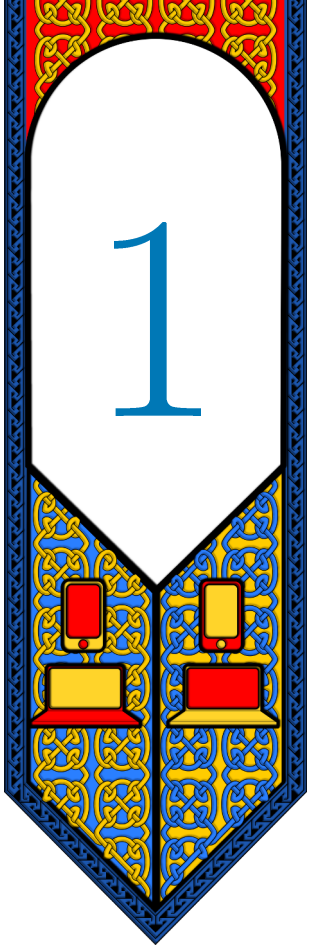
7.1 Example of scenes with color buttons . . . . . 94
7.2 SIG rules that were altered to incorporate the CS into the RE generation process . . . . . 95
7.3 Button *bhall9* and surrounding environment . . . . . 98
7.4 Buttons in an L-shaped configuration . . . . . 99
7.5 Example of failed Episodes from the IF's point of view . . . 100

8.1 Corruption model . . . . . 106
8.2 Sample scenes from our experiments in the GIVE and Tuna domains. . . . . 112

# List of Abbreviations



| | |
|---|---|
| **CFG** | Context-Free Grammar |
| **CS** | Context set |
| **FSA** | Finite-State Automaton |
| **GPS** | Global Positioning System |
| **IF** | Instruction Follower |
| **IG** | Instruction Giver |
| **IRTG** | Interpreted Regular Tree Grammars |
| **ML** | Machine Learning |
| **NLG** | Natural Language Generation |
| **NLP** | Natural Language Processing |
| **NP** | Noun Phrase |
| **PP** | Prepositional Phrase |
| **RE** | Referring Expression |
| **REG** | Referring Expression Generation |
| **SIG** | Semantically Interpreted Grammars |
| **VE** | Virtual Environment |
| **VS** | Visual Salience |
| **VSTM** | Visual Short-Term Memory |



# 1

# Introduction

It was not a dark and stormy night. It was dark, true, but in a German winter that barely rules out a quarter of the day. It was not stormy either, but it was certainly wet, foggy, and cold.

So it was a dark, wet, foggy, and cold day in southwest Germany.

Max was not happy. As he walked once more down the city's main square, with rain drops on his hair and wet grass leaves on his shoes, he tried again to find his way to the museum. This, and not the weather, is what made Max unhappy: that he didn't know his way, and his phone kept giving him directions that made no sense. "Go south", it told him, assuming that no young adult in the 21st Century would leave his house without a compass to point out where the South is. "Turn left", it would also say, followed by lengthy instructions for coming back to the starting point. *Apparently*, Max thought, *the phone meant my other left*.

There are several reasons why Max won't be happy anytime soon. Some of them we cannot fix, such as the lack of empty lockers for his backpack once he reaches his destination. But there are others that we *can* improve, and we'll focus on those a lot in the next hundred pages.

Max's phone is a miracle of technology: it can pinpoint his location anywhere in the world, combine hundreds of route directions, and pick the best one in a matter of seconds. It definitely knows where Max is, where he is going, and how are streets laid out. But as much as it knows, there is a lot still that it doesn't know: what the buildings look like, where the doors lead to, whether there are any sidewalks, or even whether Max is a car or not. It also makes no attempt to understand Max and, even though it realizes (eventually) that Max is not where he should be, it doesn't make

any attempts to figure out *why* Max went the way he did. Max is a person, living in a 3D world full of visual cues. His phone is a computer, evaluating everything in terms of graph distances and cost functions. They live in different worlds, and although Max has made his best to turn his concerns into terms his phone can understand, his phone did not level back with him.

What could have Max's phone done? For starters, it could have checked whether Max understood its instructions correctly. Why waiting until Max walks into a corner, when a simple "Sorry, my bad, go back and let's try again" would have sufficed? And even better: once Max walks back to where he started, his phone should *not* say the exact same thing that led to the dead-end in the first place.

If Max's phone had realized that Max is a pedestrian, the prospects would have been better: car drivers have to focus their attention on the road and other cars, and therefore their GPSs cannot go beyond instructions more complex than "drive 200m. and turn right". But given that Max *can* stop and look around, he could follow instructions such as "go through the arch between those two buildings", "walk to the building with yellow metallic doors", or "the yellow building – you should see it from here". These kind of instructions are more effective because they refer to attributes in the environment, making it easier for Max and his phone to coordinate with each other: if the phone refers to a yellow building that Max cannot see, he could immediately realize that something is wrong and take steps to correct it. Making reference to physical properties of the surrounding environment plays an important role in what we call an *effective referring expression*, and the process by which Max and his phone try to sort out what went wrong is called *grounding*. Both topics are central to this thesis and will be presented in great detail.

We also need to talk about what Max's phone needs. Our first problem: if we want Max and his phone to successfully reach their destination, Max's phone needs to know where they are and what things look like. These are hard problems, and discussing them would take one or two extra theses. Luckily, we can simplify: instead of asking a human to walk around and expect their phone to understand their surroundings, we can bring the human down to the phone's level. This is what we call a *virtual environment*, an artificial world where all distractions are taken away leaving behind only those aspects we care about: for a navigational task such as our example, we would have buildings and doors, but no weather nor other people. This simplified approach makes it easy enough to run all kinds of experiments with collaborators all over the world, while still being challenging enough to show what our results are solid.

The second and last problem is detecting misunderstandings. If Max went in a different direction than his phone intended, what's the best way for his phone to correct him? And how early can we detect this misunder-

standing? Both issues can be approached with a little help from statistics: if we have a good approximation of how the average person would react in a certain situation, and we observe that Max's behavior does not agree with that, we can assume that Max has misunderstood what his phone said. Even better: by exploring why would an average person do what Max just did, we can infer what Max intended to do and say "I think you misunderstood me: you seem to be walking towards the glass door, but I meant the metallic one".

This thesis focuses on methods for detecting, identifying and correcting misunderstandings in interactive, collaborative tasks. More specific, the detection and correction of misunderstood referring expressions in 3D virtual environments. We will explore probabilistic models of listener's understanding to detect *when* a misunderstanding took place, *what* the misunderstanding was, and *how* to best correct it. With these capabilities in place, Max' phone should be able to guide Max with less misunderstandings, in a shorter time, and in a more pleasant way.

## 1.1 Challenges

Generating good referring expressions, detecting, identifying, and correcting misunderstandings are very complex tasks that can be applied to a multitude of different problems and combined in a myriad of ways. This section details why each one of these tasks is relevant to our problem, the challenges we face when integrating them all in a unified framework, and a few ideas about how to overcome them. Turning these sketches into full–fledged strategies is what this thesis is about.

Max' problems are rooted in a simple confusion: his phone believes that Max is a special, slow type of car that can "drive" in the wrong direction and is not allowed on highways. This is the result of some compromises made by GPS manufacturers in the name of efficiency: displaying maps, calculating routes, and giving directions in any point of the planet represents an incredible amount of work done in a small device with limited memory and processing power. Some design choices had to be made and, as a result, your device knows where you are, where (most) roads are... and that's it. If you want to go from A to B, all your device cares about is where point A is, where point B is, and what's the shortest road that connects them both. Car drivers have limited attention to spare, and GPS devices have limited memory and information; all a car driver wants to hear is where to turn, and all a GPS device wants to do is describe the road one intersection at the time. This approach to navigation would not work inside a museum, but works very well on the highway.

Extending the functionality of navigation devices to work inside a museum is a complicated task. One key problem is rethinking the way our

device "sees" the world. Cars move in what we call *network space*, which is characterized "by clearly identifiable decision points (intersections) connected by paths (streets)" Mast and Wolter (2013); Rüetschi (2007). Traffic rules impose other constraints: a car does not typically make an unexpected 180Âř turn, drive in the wrong way, nor stops in the middle of the highway for no reason. But pedestrians move in *scene space* with no clear network structure Rüetschi (2007), get inside buildings, pay attention to their surroundings and collide with each other all the time. A good pedestrian navigation system needs to take all of this into account and, as a result, the number of ways in which instructions can be given grow exponentially. Guiding a person to the exit could include traditional instructions such as "take the door to your right", but good route instructions are useful only if they relate to environmental features Mast and Wolter (2013), such as:

(1) go through the Egyptian exhibit.

(2) the door surrounded by two Greek statues.

(3) walk through the yellow arch behind you and keep going.

If we intend to generate instructions like those, we need a *model* of the world that a computer can understand; we also need to generate *referring expressions* (RE) for all objects, and a mechanism to decide which of several possible referring expressions is the "best" one – and all of it has to be done efficiently, too.

The challenge of finding a good representation of the world is discussed in Section 1.1.1, where Virtual Environments are introduced as a mechanism for testing new research ideas much more efficiently than testing in the real world would be. Section 1.1.2 is focused on the challenges involved in detecting that a misunderstanding took place, arguing that a probabilistic model of listener's understanding is the best approach for the task. And finally, having made our case in favor of detecting misunderstandings, 1.1.3 details the challenges that must be overcome to generate corrections that are both natural and effective.

### 1.1.1 Challenges in developing for the real world

The instructions we've seen in the previous paragraph are definitely good, but *how* good are they, really? Each one of them raises further questions. What if in (1) I don't know the difference between the Egyptian and Sumerian exhibits? What if I know the two Greek statues in (2) are actually Syrian, but belonging to the Hellenistic period - is that the correct door? What if I turned while instruction (3) was being spoken? And what if I'm colorblind? Luckily, rather than guess, we can experiment: we can test several different strategies, measure which ones give the best results, and implement only the ones that make sense for our use case.

The last step cannot be overstated: even if a strategy is shown to be the one guiding users the fastest and/or with the least number of instructions, it might still not be the best strategy for *us*. Maybe it's too expensive, requiring us to manually enter too many attributes per object; maybe it's too inconvenient, requiring people to walk with their phones in front of their faces at all times. Or maybe it's too unfriendly, too unnatural, or too slow. These are called non-functional requirements, and they have the annoying habit of popping up only after we've done all the work, which we now have to throw away.

We would like to find a spot in the middle in which we can experiment with as many complicated strategies as we can without spending too many resources on them. This is the reason why we use *Virtual Environments* (VE) rather than experiments in the real world. If we wanted to navigate a museum we would have to catalog all artifacts, develop computer vision algorithms capable of recognizing them, install expensive, fine-grained, position-detection hardware, and heavily optimize for the kind of underpowered portable devices visitors are expected to carry around. In contrast, we could test our algorithms in a 3D world with made-up artifacts with comparable results at a fraction of the development and test cost. All that remains afterwards is to map the real world onto our VE. More information on VEs is presented in Chapter 2.

### 1.1.2 Challenges in detection of misunderstandings

Detecting misunderstandings is an example of the type of scaffolding that, done well, can greatly enhance a user's experience. It is a well known defect of computers that they do what we tell them to do, rather than what we *meant* to tell them. Luckily, for some specific, limited tasks computers are wisening up: cameras put our faces in focus automatically, search engines ask us whether we meant something else, social networks guess who among several strangers with the same name is the one we are looking for, and so on. The key word here is *limited*: the smaller the task, the easier it gets to infer what a user is trying to do and to add some extra scaffolding that helps them with their work.

In the context of a navigational task, detecting a misunderstanding is straightforward: we expected the user to move in a certain direction, but they went somewhere else. Detecting that a RE was misunderstood is not as easy, and potentially costly: if a user activates a fire alarm due to a misunderstood RE, no correction will takes us back to a state in which the user is not in trouble. It is important not only to detect that a misunderstanding took place, but also to do it as early as possible.

To detect a misunderstanding, we need to model the *listener's understanding* of a RE. This is a limited enough task that, while challenging, it's within our reach. In this thesis we'll explore how to keep track of the user's

*focus* in all objects around them, generate REs that take advantage of the user's mental model, and detect early if the user's response to a RE does not correspond with the expected response to said RE. This topic is the focus of Chapters 4 and 5. Chapter 4 shows how a log-linear model trained on user data can accurately predict the resolution of a Referring Expression, while Chapter 5 shows an extension to this model based on eye-tracking data.

### 1.1.3 Challenges in correcting misunderstandings

Even the best navigation system has to deal with the topic of misunderstandings. *Misconstruals*, as discussed in Section 2.6, are influenced by a multitude of factors that are external to the text of the conversation itself. Misunderstandings can take place at any point during a dialogue.

In the case of pedestrian navigational tasks, there's one specific type of mistake that we care about: unsuccessfully-resolved REs.

It should be clear that the most effective way of correcting misunderstandings is not to have misunderstandings to begin with. In order to generate referring expressions that are best suited for a specific domain, multiple families of algorithms have been developed based on their main objective. Some approaches optimize for human-likeness (Altamirano et al., 2012), brevity (Dale, 1989), usefulness (Garoufi and Koller, 2011a), over- or under-specification (Koolen et al., 2009), specific use cases such as medical decision support (Portet et al., 2009), etc. Interactivity is also an important factor: while some tasks are limited to *one-shot* strategies (Gorniak and Roy, 2004), other tasks allow for reference in installments and corrections.

The scenarios that we have described during this chapter are always interactive, and our REG must be suited for it. Interactive settings allows us to monitor the user's behavior and generate corrections at any time but, as a downside, require fast algorithms capable of dealing with an environment in constant change. No matter which strategy we choose, it must be fast enough to be used in real time.

The type of corrective RE that we choose plays a major role too. For some specific situations and domains, even the simplest approach may be enough to keep a user in track. If we expected a user to take the door on the left, but the user is about to take the door on the right, an utterance as simple as "no, not that way" could be enough. Our typical scenarios are not like that: in pedestrian navigation, the range of possibilities is so large that merely pointing out the presence of a misunderstanding does not suffice — rather, we need to make it clear that a misunderstanding took place, what the misunderstanding was, and we need to point out again what the original intention was.

We intend to keep misunderstandings as low as possible by using the REG algorithm presented by Engonopoulos and Koller (2014). This algo-

rithm can codify a (possibly infinite) set of REs for a target object and select the most effective one, where *the most effective* is defined as "the RE that maximizes the probability of being understood as the target referent".

Should a misunderstanding still take place, we also present two approaches for the generation of corrective Referring Expressions built as extensions of this REG framework. For the instruction presented in (2), "the door surrounded by two Greek statues", we could imagine a user that chooses a door surrounded by Egyptian statues instead. The first approach we introduce to correct this misunderstanding explores whether a theory of mind based on *Context Sets* can generate good corrections. This approach could generate corrections like "no, Greek statues", where "door" is no longer mentioned because it is assumed that the user has already understood that part, and is only confused about a specific characteristic. Chapter 7 presents a different (and more successful) approach, in which the correction would introduce *contrastive feedback* and read "no, the door surrounded by **GREEK** statues".

### 1.1.4 Published research

The Challenges presented here and the solutions we propose have been thoroughly discussed and tested. A significant part of the work presented here has been published in peer-reviewed conferences and journals, and several Chapters of this thesis correspond roughly with one of them.

The simple Instruction Follower system described in Chapter 3 was published in Benotti et al. (2012). This paper details how to interpret an instruction in a VE without a manually annotated corpora: Given a corpus of instructions and the user actions that followed them, this approach associates the user-performed actions and the state of the VE to the given instruction. Benotti et al. (2014) expands on these results, fine-tuning the details of the overall algorithm.

The system for predicting misunderstandings detailed in Chapter 4 was first published in Engonopoulos et al. (2013). This paper details two probabilistic models, $P_{sem}$ and $P_{obs}$, that predict a user's reaction to an instruction based respectively in the semantics of the instruction and the observed behavior of the user after receiving said instruction. Both models can be combined in a probabilistic model $P_{comb}$, improving on the accuracy of either individual model. These models can reliably predict to which target object will a RE be resolved, which we will use to detect misunderstandings. The improved results with eye–tracking detailed in Chapter 5 were presented in Koleva et al. (2015). At the time of this writing, it is expected that both results will also be detailed in the PhD Theses of Nikos Engonopoulos and Nikolina Koleva, respectively.

The algorithm for generating contrastive referring expressions, related experiments, and results described in Chapter 8 were published in Villalba

et al. (2017). This paper details a method for generating contrastive REs, such as "No, the **BLUE** button". This method can be used to correct misunderstandings, as the explicit contrast marking is shown to be preferred over corrective REs without contrast.

## 1.2 Summary

The research presented in this thesis improves several aspects of an instruction giving system. A system built with this approach should be capable of detecting and correcting misunderstandings thanks to its model of listener's understanding. With this model we can infer where has the user focused their attention, giving us a chance to accurately detect when a misunderstanding takes place and how to better correct it. To my knowledge, this is the first time that so much care is paid to user's attention in an end-to-end system.

Each stage of this research improves over the previous one, ensuring that the entire *communicative act* (instruction, misunderstanding, and correction) can be handled from beginning to end in one place. We expect this process to feel much more natural to an IF than previous IG systems.

Chapter 2 presents a detailed introduction to the task of Instruction Following, laying the technical foundation for this thesis. This Chapter introduces key concepts such as *Virtual Environments*, *Referring Expression Generation*, the *GIVE Challenge*, and an overview of relevant related work. The Chapter closes with an analysis of the pragmatics involved throughout this process, introducing well-researched concepts like *Common ground* and *Context Set* that underlie our algorithms.

Chapter 3 presents an instruction following system, capable of following instructions in simple virtual environments with a high degree of accuracy. Extending this system to more complex environments requires enhancing the IF system with several attributes, and these attributes will lay the foundations for what we can expect from an ideal human IF in the following chapters.

Chapter 4 uses these attributes to explain how and why misunderstandings occur, even when an instruction is technically correct. The Chapter introduces a probabilistic model to detect and predict misunderstandings, divided into two models of listener's understanding, showing that these models are more effective at this task when combined. These models will tackle the problem of detecting misunderstandings based on two complementary approaches: a semantic model that predicts misunderstandings based on the semantics of an instruction, and an observational model that predicts misunderstandings based on the observed behavior of the IF.

Chapter 5 introduces a detailed analysis of user attention, rooted in the psycholinguistics literature. This analysis introduces *eye–tracking* as a use-

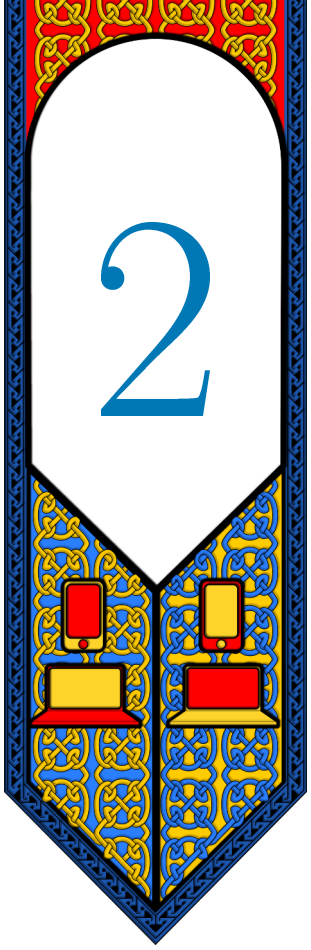
ful technique, and it is shown that features based on eye–tracking can effectively improve the accuracy of the observational model presented in Chapter 4.

Chapter 6 takes a small detour, detailing our efforts in the collection of data for our crowdsourced experiments. This is a challenging and underreported aspect of modern online research, and it's included here with enough detail to be of use for future researchers. Our methodology of data collection is rooted on the concept of a *Security Mindset*, a concept we borrow from the Computer Security community and apply with great success.

Once a misunderstanding has been detected, it is necessary to act on it. Chapter 7 discusses our first strategy: the correction of misunderstandings using a simple reformulation of the misunderstood referring expression and a model of user attention. This strategy does not perform as well as we expected and, as a result, we dedicate the second half of the Chapter to error analysis.

A step forward in feedback strategies is the implementation of contrastive feedback, a type of feedback in which both the misunderstanding and the correction are presented in clear contrast to each other. Chapter 8 explores the use visual contrast between what we believe the listener understood and what was actually intended, and we show that this strategy is both effective and natural to guide users in the intended direction.

Finally, Chapter 9 presents how this system can work as a pipeline giving instructions, monitoring their effect, and presenting feedback whenever misunderstandings are detected. The Chapter, and this thesis, close with thoughts on future work that builds on the concepts introduced here.



# 2

# Instruction following in Virtual Environments

The term "Instruction following" refers to a family of tasks in which two participants with clearly defined roles must cooperate to reach a specific goal, one as the "Instruction Follower" and the other as the "Instruction Giver". The Instruction follower has limited information about the state of the environment, and must be guided by the better-informed Instruction Giver in order to complete the task. They are typically allowed to communicate with each other, although specific tasks call for restrictions on which type of communication between them is allowed.

A large number of everyday situations fall under this family of tasks. The car driver that follows the instructions of his GPS, the tourist asking how to reach Baker Street, the daughter that explains to his father over the phone how to use the printer, and the man in the supermarket going through a list written by their wife, they are all involved in Instruction Following tasks. These tasks are very different from each other, and it is the purpose of this Chapter to provide a unified framework to reason about all of them.

This thesis focuses on one specific sub-category of Instruction Following tasks, *Navigational* Instruction Following. These tasks require the Instruction Follower to navigate an environment (real or virtual), and demand that the Instruction Giver provide not only a description of *where* to go, but also *how to go there*. Following a GPS is a Navigational task, but collecting items from a grocery list is not. Throughout this thesis, and unless otherwise specified, we use the term "Instruction Following" to refer

to "*Navigational* Instruction Following".

At it's core, Instruction Following can be divided into three interrelated tasks:

1. Making a plan about how to achieve the next required step of the overall goal,
2. Guiding the Instruction Follower to the place where the next action must take place, and
3. Explaining what has to be done in there.

When following a GPS, for instance, the system may evaluate multiple possible routes, and reach the conclusion that the best route involves taking the highway (1). The system may then guide the driver to the highway saying "drive 500m and turn left" (2) and, once there, instruct the driver to "turn left and get on the highway" (3). It is clear that the system does not intend the trip to end there, but rather, than both the Instruction Follower and Giver have collaborated and, as a result, are both now closer to their intended destination.

This Chapter introduces the foundations required to understand our Instruction Following framework. Automated planning, introduced in Section 2.2, explains how a system decides which action should be performed next; Section 2.3 presents an overview of the most influential and/or relevant approaches to Natural Language Generation, the part of the task in which computer instructions and databases are turned into human language. This Section puts a strong focus on Referring Expression Generation and Semantically Interpreted Grammars, as these concepts are particularly important in subsequent chapters.

Not all instructions are followed in the real world - indeed, research in this thesis is performed in what we call a *Virtual Environment*. Section 2.4 examines what a Virtual Environment is, what are its characteristics, and what are the benefits of using them. All of these research areas are combined in the GIVE Challenge, a Virtual Environment that will be used throughout this thesis as a test bed for our experiments. Finally, Section 2.6 will introduce theoretical concepts from the Pragmatics and Linguistics literature that will be necessary to contextualize our experimental results and algorithms, giving us the background to understand not only that our algorithms *work*, but also providing a framework for explaining *why* they work.

## 2.1 Early work on Instruction Following

Early work on (Navigational) Instruction Following was performed by linguists such as Klein (1982) who studied the role of verbal and situational

context in language behavior. In the context of a larger project, this study brings together Linguistics and Psychology to understand and explain how space structures language in ways that neither field could explain independently. The study asked people for directions in central Frankfurt while recording their interactions for later transcription, and presented interesting insights on the interaction between local deictics ("here", "there", "right", etc.), situational awareness, and the need of a clear task plan as a condition for a successful completion of the task. Di Eugenio (1992) is also considered a precursor of the analysis of instruction following, suggesting that a successful analysis of an instruction should focus not only on the syntax of an instruction, but also on its purpose. Unlike previous linguistic work, Di Eugenio presents also an early algorithm for instruction following. These and similar works would introduce the idea of "instruction following" as a concrete task, collecting and releasing data that would be analyzed by researchers for many years to come.

"Instruction following" would be more formally defined in the influential HCRC Map Task Corpus of Thompson et al. (1993). Originally designed for speech recognition, the HCRC Map Task Corpus comprises recordings of a task in which two human participants must collaborate to successfully navigate a map with named locations. In this task, both the Instruction Follower (IF) and Instruction Giver (IG)[1] were provided with copies of a 2D map, and the IG was asked to guide the IF across a route that's only visible to him. In line with previous studies, the released corpus includes digital copies of the maps, recordings of the participants' conversation, and their transcriptions. This data would be later used as training data for the earliest automated, data-driven IF systems. Similar corpora of natural language instructions obtained from human participants is still released to this day with data from more challenging environments, orders of magnitude larger, and/or including a wider range of test subjects. Examples can be found in the work of Kollar et al. (2010), who collected instructions, 3D LIDAR scans and camera data of real office environments, and in the vasts amounts of crowdsourced data released by de Vries et al. (2018).

The HCRC Map Task Corpus inspired the development and testing of systems capable of interpreting the natural language navigational directions provided in the corpus (Levit and Roy, 2007; Vogel and Jurafsky, 2010). This research would later expand towards more complex domains, highlighting a need for a more systematic data collection process and tasks with a more focused approach. As a result, modern research on Instruction Following is typically performed in *virtual* environments, rather than in real-life scenarios.

[1]The terms "Instruction Follower" and "Instruction Giver" are first introduced in this paper

## 2.2 Automated planning

*Automated planning* is defined as the process of finding a sequence of actions which, if executed by an agent, result in the achievement of a set of predefined goals. This sequence of actions is known as a *plan* (Partalas et al., 2008), and a computer system that generates this plan is known as a *planner*. Instruction Following as a task relies heavily on planning: deciding which steps are required in which order to achieve the goal of the task, deciding which instruction to present, and evaluating the progress made on the task are all aspects of an IF system that require planning.

To make *planning problems* tractable, *classical planning* makes simplifying assumptions: the environment is assumed to be completely observable, deterministic, finite, static (changes only occur when an agent acts) and discrete in time, actions, objects, and effects (Russell and Norvig, 2004). In addition, approaches that do not rely on all of these simplifications are the focus of extensive modern research.

STRIPS (STanford Research Institute Problem Solver) is both the name of an influential classical planner developed by Fikes and Nilsson (1971) and the name for the problem-representation formal language of the planner. STRIPS represents states as a sequence of connected, positive propositional variables or *conditions* where unmentioned conditions are assumed to be false (closed world hypothesis). A problem is then described in terms of:

- An *initial state* describing which conditions are true at the beginning of the problem.
- A *goal* describing the conditions that we would like to satisfy.
- A set of *actions* specified in terms of the *preconditions* that must be satisfied before the action is executed and the *effects* describing how the execution will change the state of the problem.

**Example: A STRIPS problem**

Imagine that we are inside a dark room, and we wish to make the room bright. We can describe the initial state of the problem with the logical expression $\{In(R_1) \wedge Dark(R_1) \wedge Room(R_1) \wedge Lamp(L_1)\}$. Our desired goal can be described as $\{In(R_1) \wedge Bright(R_1)\}$, and we can reach it through the following actions:

**switch_lamp_on** $(lamp)$
PRECOND: $Near(lamp) \wedge Lamp(lamp)$
EFFECT: $\neg Dark(R_1) \wedge Bright(R_1)$

**move_closer** $(room, object)$
PRECOND: $In(room)$

EFFECT: $Near(object)$

One possible way to reach the goal is applying the sequence of actions $\langle$**move_closer**$(R_1, L_1)$, **switch_lamp_on**$(L1)\rangle$. We then reach the state $\{In(R_1) \wedge Bright(R_1) \wedge Near(L_1) \wedge Room(R_1) \wedge Lamp(L_1)\}$ which satisfies all the conditions of our goal.

Following STRIPS, further research on automatic planning showed that this formalism was not expressive enough for certain real domains, leading to the development of new formalisms. ADL (Action Description Language) is one of the most important ones. Based on an open world hypothesis (where unmentioned literals are not false, but rather unknown), ADL extends the expressive power of logical formulas allowing for negation in states, allowing conjunctions and disjunctions in goals, and introducing features such as types, equality comparisons, and conditional effects among others (Pednault, 1987). Both STRIPS and ADL have been included into PDDL (Planning Domain Definition Language), a common formalism for describing planning domains that allows researchers to share and compare problems and results (McDermott et al., 1998).

In this thesis, we make use of planning in two specific circumstances. Section 3.2.1 uses the output of a planner to discretise user behavior, linking user behavior to the actions of a planner. In addition, the baseline P1 system used in Chapters 7 and 8 includes planning at two stages: the sequence of movements and object manipulation actions that the automated Instruction Giver implements is provided explicitly by the GIVE Framework, and referring expressions are generated using a planning-based approach to sentence generation (Garoufi and Koller, 2011b; Koller and Stone, 2007).

## 2.3 Natural Language Generation

Reiter and Dale (1997) define Natural Language Generation (NLG) as "the sub-field of artificial intelligence and computational linguistics that is concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information". Gatt and Krahmer (2018) argues for a less restrictive definition, using instead NLG to refer to "systems that generate text from non-linguistic data". But even this definition may not be enough: tasks such as Automatic Summarization and Paraphrasing receive linguistic data and input. For the purposes of this thesis, we define NLG simply as the family of tasks that support the generation of natural text and where the "final product" is interesting not only because it's text, but rather because of the *semantics* of that text. A word processor can generate text, but it is not an NLG application.

A way to address the problem of NLG is in terms of the sub-problem(s) that a specific system solves. Although there is no single agreed taxonomy of NLG tasks, the classification of Reiter and Dale (1997, 2000) is the most widespread one. This classification includes the following families of problems, with Figure 2.1 showing an example of how all steps can be involved in the task of generating weather reports.

**Content determination** Decide which part(s) of the input data should be included in the text that will be generated.

**Text structuring** Decide in which order should data be presented.

**Sentence aggregation** Decide the content of each sentence that will be generated.

**Lexicalization** Decide how will the content of a sentence be expressed as a sentence.

**Referring Expression Generation** Decide which attributes of objects are necessary to identify them in a given domain.

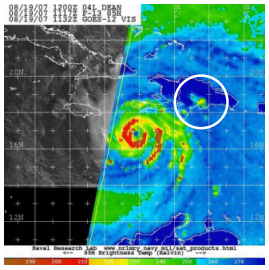
**Linguistic realization** Combine individual words into full sentences, and individual sentences into a full text.

The main challenges in **content determination** are deciding which information is relevant and which information can be ignored. As an example, and given the same input data, a weather report system for a daily news report could extract unremarkable, average information from the areas surrounding major cities while a weather warning system would look for specific patterns all over the country. Figure 2.1.a. illustrates this step.

Deciding in which order to present information is the problem that **text structuring** focuses on. For the weather warning example, the system might decide that the type of event should be introduced first, then the areas more likely to be affected, and finally the characteristics of the event (Figure 2.1.b).

The next step, **sentence aggregation**, decides how should information be combined into specific sentences. For our running example in Figure 2.1.c, the system might decide that the time and date of the event should be presented in the same sentence, merging the events $[hurricane, sunday]$ (which we can interpret as the event "a hurricane is expected on Sunday") and $[Dominican\ republic, south\ coast]$ ("the affected area is the south coast of the Dominican Republic") into a single sentence such as "a hurricane will approach the south coast of the Dominican Republic on Sunday".

The step that follows, **Lexicalization**, is the step in which we no longer decide what information should be shown, but rather settle on how should this information be displayed in a sentence. This step identifies



(a)

[ hurricane, sunday ], [ Dominican Republic, 18.70090, -70.16545 ], [ high winds, heavy rains ]

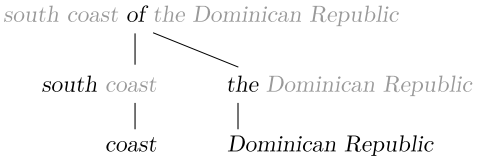
(b)

[ hurricane, sunday, Dominican Republic ], [ 18.70090, -70.16545 winds, rains ]

(c)

Event
TYPE dynamic
PRED approach
TENSE future
ARGS [ TIME sunday, PLACE Dom. Rep., REGION south-coast ]

(d)




(e)

*A hurricane will approach the south coast of the Dominican Republic on Sunday. High winds and heavy rains are expected.*

(f)

Figure 2.1: Example of all steps involved in a weather report system. a) Content determination. b) Text structuring. c) Sentence aggregation. d) Lexicalization. e) Referring Expression Generation. f) Linguistic realization.

**Referring Expression Generation** is the task in charge of "naming" the objects to which we want to refer. In our example, we could use expressions such as "the hurricane", "Hurricane Gilbert", "the storm", and so on. This is the step shown in Figure 2.1.e, and a detailed discussion is presented in the next Section.

Finally, **linguistic realization** combines the information obtained in previous steps into a single, coherent text. This is the step that generates the final text seen in Figure 2.1.f.

### 2.3.1 Referring Expression Generation

Referring Expression Generation (REG) is characterized by Reiter and Dale (1997) as "the task of selecting words or phrases to identify domain entities".

Early work on REG is typically traced back to the 1970's and 1980's, starting with the work of Winograd (1972) on Natural Language Understanding. Their work presented SHRDLU, a program capable of both natural language understanding and generation, the latter of which used a relatively simple REG algorithm: it considered a number of attributes in a fixed order (type, color, size, the object that it supports, and the object that is near), and added them until it reached a uniquely identifying RE. Researchers belonging to what Krahmer and van Deemter (2012) calls the 1980s "California School" such as Appelt and Kronfeld studied REG in the context of larger speech acts, hoping to better understand the complexities of human communication. Their approach, reflected in programs such as KAMP (Appelt and Kronfelt, 1987), considered that "research progress was best made by investigating hard, anomalous cases that pose difficulties for conventional accounts"[2]. Their systems generate REs as part of a planning system and account for complex interplays between the speaker's and hearer's knowledge, but this approach would later fall out of favor.

In the 1990s REG would be once again considered as a task by its own right, focusing on the problem of determining which properties of an object should be chosen for the identification of a referent. Research of this period would be strongly influenced by the work of Dale and Reiter. Their first breakthrough would be the Full Brevity Algorithm (Dale, 1989), which selects the minimum number of properties required to identify a referent, at the cost of a high algorithmic complexity and REs that did not always match those of human speakers. This work was later extended with the Greedy Algorithm (Dale, 1992) that selects at every step the property that excludes the largest number of remaining distractors, achieving good results with better complexity. But the most influential paper of this era would be the Incremental Algorithm (IA) of Dale and Reiter (1995). This

[2]Doug Appelt, personal communication

algorithm operates in a similar fashion as Full Brevity, but selecting at every step the most *preferred* attribute (which is given as a parameter) rather than the most distinguishing one. Later work would modify the IA to account for properties such as the saliency of an object (Krahmer and Theune, 1999) or relational descriptions (Horacek 1996; Krahmer and Theune 2002; Kelleher and Kruijff 2006).

Research up until this point focused on efficiently computing which properties of an object were required to make a distinguishing description. Research in the 2000s would focus instead on more challenging REG sub-tasks. Krahmer and van Deemter (2012) characterize modern REG in terms of limitations of previous approaches and how modern algorithms deal with them, a classification that we follow here.

The most straightforward extension of the IA are algorithms capable of generating REs for sets rather than individual objects. Given the set $\{b_1 : (red, chair), b_2 : (green, book), b_3 : (blue, book)\}$, a simple RE for $\{b_1, b_2\}$ would generate "the red chair and the green book", but a better approach would generate "the objects that are not blue". An approach for such Boolean expressions was first proposed by van Deemter (2002) and refined by Gardent (2002), bringing back the Full-Brevity algorithm to prevent situations in which excessively large REs are generated. Horacek (2004) would improve even further by expressing logical propositions in Disjunctive Normal Form rather than the Conjunctive Normal Form of previous approaches, a change that led to shorter REs and that future work adopted as the standard.

Describing objects in relation to other objects would prove a challenge to the IA. A typical difficult situation is a scene where neither "the button" nor "the window" are uniquely identifying but "the button above the window" is. These type of REs do not lend themselves well to incremental strategies, a problem further exacerbated by the possibility of finding loops (as in "the button above the window underneath the button above the . . . ").

A different set of problems is posed by properties that are not crisply defined, gradable, vague, and/or context dependent. An object described as "the thin sumo wrestler" in a specific context could easily be described as "the large man" in a different one. REs such as "the tall man" can also cause problems in borderline cases. As a simple solution, the later case can be solved if we stop algorithms from using these properties when the gap between the target object and distractors is not big enough (Gorniak and Roy, 2004).

A special case of context-dependent properties is saliency. The question of when is an object salient enough than an otherwise vague RE suffices is one that has not yet been solved. Current approaches model these situations in terms of *focus stacks* or *context sets*, a topic from the Pragmatics literature that we discuss in Page 34. DeVault et al. (2004) generation approach assumes that a vague RE can be correctly resolved to the highest

object in the focus stack that matches the description, while Krahmer and Theune (1999) present a more nuanced approach in which the position of an object in the stack is replaced by salience weights.

A final issue to consider is how knowledge about the world is represented. Modern approaches are expected to be efficient enough to allow for the development of fast algorithms, and yet remain flexible enough to allow algorithms to be reused in multiple domains. Modern approaches tend to gravitate towards search-based (Kelleher and Kruijff, 2006), logic-based (Areces et al., 2008), or graph-based approaches (Krahmer et al., 2003).

Search-based approaches consider REG as the problem of finding a specific goal through a large space of possible states. Each state can be modeled as a triple $\langle L, C, P \rangle$ consisting of a description $L$ of the target that's also true for all distractors $C$ and a set of properties $P$ of the target whose inclusion remains to be decided. The initial state is the state $\langle \emptyset, C, P \rangle$ with no selected properties and no discarded distractors, while the goal state $\langle L, \emptyset, P' \rangle$ contains a description of the target $L$ that is not true for any other object.

Logic-based approaches are based on the premise that an RE can be understood as a formula of Description Logic, turning REG into the problem of finding a formula that denotes the target set of objects. Finally, Graph-based approaches model the knowledge base as labeled, directed scenes graph where objects are nodes and properties and relations are edges. The problem of REG can be then understood as the search for a distinguishing graph that is also a sub-graph of the scene.

In this thesis, however, we are particularly interested in a new kind of approach, namely, a *grammar-based* approach. This approach models REG in terms of standard parsing techniques in a formalism known as *Interpreted Regular Tree Grammars*.

### 2.3.2 Interpreted Regular Tree Grammars

All REs in this thesis are generated using the grammar formalism known as *Semantically Interpreted Grammars (SIG)* (Engonopoulos and Koller, 2014; Koller and Engonopoulos, 2017), an approach that integrates both REG and surface realization. It is defined as "a synchronous grammar formalism that relates natural language strings with the sets of objects in a given domain which they describe", and it is a specific application of Interpreted Regular Tree Grammars (IRTG).

A SIG is defined as a triple $(\mathcal{G}, \mathcal{I}_\mathcal{S}, \mathcal{I}_\mathcal{R})$. $\mathcal{G}$ is a *regular tree grammar* that describes languages of derivation trees; $\mathcal{I}_\mathcal{S}$ is a *string interpretation*, a function that maps any derivation tree over $\mathcal{G}$ to a string. Finally, $\mathcal{I}_\mathcal{R}$ is a *relational interpretation* that maps any derivation tree over $\mathcal{G}$ to a class of *relations*. This SIG can then relate strings with the sets of objects they refer to, and vice versa.

In the relational interpretation $\mathcal{I}_\mathcal{R}$, objects are represented as elements

in a set, properties are represented as sets, and relations between objects are represented as ordered pairs of elements. For example, the set $blue = \{b_1\}$ indicates that the object $b_1$ is blue, and the set $left-of = \{(b_1, b_2)\}$ indicates that the object $b_1$ is to the left of the object $b_2$.

Figure 2.2 shows an annotated example of a SIG capable of generating REs such as "the window", "the button to the [left/right] of the yellow button", "the blue button above the window" and so on. This SIG constructs $\mathcal{I}_\mathcal{R}$ combining the denotations of atomic predicate symbols provided as a world model and the following operations:

$proj_i(R)$ projects every element $r = \langle r_1, \ldots, r_k \rangle \in R$ to its $i$-th component, or $\emptyset$ if $i > k$.

$R_1 \cap_i R_2$ is the intersection on the $i$-th component of $R_1$, i.e., the set of elements $r = \langle r_1, \ldots, r_k \rangle \in R_1$ such that $r_i \in R_2$ (or $\emptyset$ if $i > k$).

$uniq_a(R)$ evaluates to $\{a\}$ if $R = \{a\}$, and to $\emptyset$ otherwise.

$member_a(R)$ evaluates to $\{a\}$ if $a \in R$, and to $\emptyset$ otherwise.

To better understand the generation algorithm, we use now the SIG from Figure 2.2 and the world model shown in Figure 2.3 to generate the RE "the blue button above the window", which uniquely identifies the button $b_1$. The derivation steps are shown in Figure 2.4 with the derivation tree in $\mathcal{G}$ in the center, the string interpretation $\mathcal{I}_\mathcal{S}$ of each subtree on the left, and the relational interpretation $\mathcal{I}_\mathcal{R}$ of each subtree on the right.

The step (1) of the derivation tree shown in Figure 2.4 uses the grammar rule $\mathrm{N}_a \rightarrow button_a$ and yields the terminal *button* whose string interpretation is the string "button" and whose relational interpretation is the set $\{b_1, \ldots, b_4\}$. Applying the rule $\mathrm{N}_a \rightarrow blue_a(\mathrm{N}_a)$ in (2) yields the tree $blue_a(button_a)$, the string interpretation "blue • button", and the relational interpretation

$$\begin{aligned} \text{“blue } \cap_1 \text{ button”} &= \{b_1, b_4\} \cap_1 \{b_1, \ldots, b_4\} \\ &= \{b_1, b_4\} \end{aligned}$$

Rule (3) is similar to (1), but (4) applies the rule $\mathrm{NP}_a \rightarrow def_a(\mathrm{N}_a)$ that guarantees that its non-terminal $\mathrm{N}_a$ contains a single element. The tree interpretation of $\mathrm{NP}_a \rightarrow def_a(window_a)$ is the string "the window", and its relational interpretation is the set $\{w_1\}$.

The result of applying rule (5) is the string interpretation "blue button • above • the window" and the relational interpretation:

for all $a \in U$:
$\mathrm{NP}_a \rightarrow \mathit{def}_a(\mathrm{N}_a)$
$\mathcal{I}_S(\mathit{def}_a)(w_1) = \mathsf{the} \bullet w_1$
$\mathcal{I}_R(\mathit{def}_a)(R_1) = \mathsf{member}_a(R_1)$

**String interpretation:** the string "the" concatenated with the string interpretation $w_1$.
**Relational interpretation:** $R_1$ if $a \in R_1$, and an empty set otherwise.

for all $a \in \mathsf{button}$:
$\mathrm{N}_a \rightarrow \mathit{button}_a$
$\mathcal{I}_S(\mathit{button}_a) = \mathsf{button}$
$\mathcal{I}_R(\mathit{button}_a) = \mathsf{button}$

**String interpretation:** the string "button"
**Relational interpretation:** the set *button*

for all $a \in \mathsf{window}$:
$\mathrm{N}_a \rightarrow \mathit{window}_a$
$\mathcal{I}_S(\mathit{window}_a) = \mathsf{window}$
$\mathcal{I}_R(\mathit{window}_a) = \mathsf{window}$

Same principle as above

for all $a \in \mathsf{blue}$:
$\mathrm{N}_a \rightarrow \mathit{blue}_a(\mathrm{N}_a)$
$\mathcal{I}_S(\mathit{blue}_a)(w_1) = \mathsf{blue} \bullet w_1$
$\mathcal{I}_R(\mathit{blue}_a)(R_1) = \mathsf{blue} \cap_1 R_1$

**String interpretation:** the string "blue" concatenated with the string interpretation $w_1$.
**Relational interpretation:** intersects the set *blue* with $R_1$ on the first component. The result is the set of all elements in *blue* whose first component is a member of $R_1$.

for all $a \in \mathsf{yellow}$:
$\mathrm{N}_a \rightarrow \mathit{yellow}_a(\mathrm{N}_a)$
$\mathcal{I}_S(\mathit{yellow}_a)(w_1) = \mathsf{yellow} \bullet w_1$
$\mathcal{I}_R(\mathit{yellow}_a)(R_1) = \mathsf{yellow} \cap_1 R_1$

Same principle as above

for all $a, b \in \mathsf{left-of}$:
$\mathrm{N}_a \rightarrow \mathit{leftof}_{a,b}(\mathrm{N}_a, \mathrm{NP}_b)$
$\mathcal{I}_S(\mathit{leftof}_{a,b})(w_1, w_2) = w_1 \bullet \text{to the left of} \bullet w_2$
$\mathcal{I}_R(\mathit{leftof}_{a,b})(R_1, R_2) = \mathsf{proj}_1((\mathsf{left-of} \cap_1 R_1) \cap_2 R_2)$

**String interpretation:** concatenates the string interpretation $w_1$ with the string "to the left of" and the string interpretation $w_2$.
**Relational interpretation:** the set $\mathit{left-of}$ contains ordered pairs of elements, making $(\mathsf{left-of} \cap_1 R_1)$ the set of ordered pairs whose first component is in $R_1$. This set is then intersected with $R_2$ on the second component, yielding the set $\{\langle a, b\rangle \in \text{left-of} | a \in R_1, b \in R_2\}$. The final operation $\mathit{proj}_1$ maps all ordered pairs into their first component.

for all $a, b \in \mathsf{right-of}$:
$\mathrm{N}_a \rightarrow \mathit{rightof}_{a,b}(\mathrm{N}_a, \mathrm{NP}_b)$
$\mathcal{I}_S(\mathit{rightof}_{a,b})(w_1, w_2) = w_1 \bullet \text{to the right of} \bullet w_2$
$\mathcal{I}_R(\mathit{rightof}_{a,b})(R_1, R_2) = \mathsf{proj}_1((\mathsf{right-of} \cap_1 R_1) \cap_2 R_2)$

Same principle as above

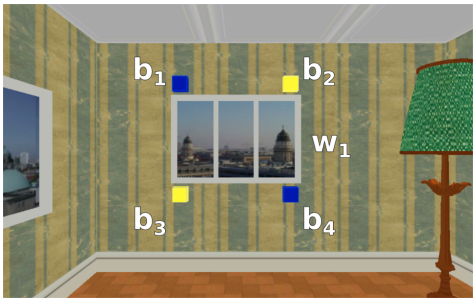
for all $a, b \in \mathsf{above-of}$:
$\mathrm{N}_a \rightarrow \mathit{above}_{a,b}(\mathrm{N}_a, \mathrm{NP}_b)$
$\mathcal{I}_S(\mathit{above}_{a,b})(w_1, w_2) = w_1 \bullet \mathsf{above} \bullet w_2$
$\mathcal{I}_R(\mathit{above}_{a,b})(R_1, R_2) = \mathsf{proj}_1((\mathsf{above-of} \cap_1 R_1) \cap_2 R_2)$

Same principle as above

for all $a, b \in \mathsf{below-of}$:
$\mathrm{N}_a \rightarrow \mathit{below}_{a,b}(\mathrm{N}_a, \mathrm{NP}_b)$
$\mathcal{I}_S(\mathit{below}_{a,b})(w_1, w_2) = w_1 \bullet \mathsf{below} \bullet w_2$
$\mathcal{I}_R(\mathit{below}_{a,b})(R_1, R_2) = \mathsf{proj}_1((\mathsf{below-of} \cap_1 R_1) \cap_2 R_2)$

Same principle as above

Figure 2.2: Annotated example of a SIG




(a) Graphic representation of the world model of the example

$$
\begin{array}{lcl}
button & = & \{b_1, b_2, b_3, b_4\} \\
window & = & \{w_1\} \\
yellow & = & \{b_2, b_3\} \\
blue & = & \{b_1, b_4\} \\
right-of & = & \{(b_2, b_1), (b_4, b_3)\} \\
left-of & = & \{(b_1, b_2), (b_3, b_4)\} \\
above-of & = & \{(b_1, w_1), (b_2, w_1), \\
 & & (w_1, b_3), (w_1, b_4)\} \\
below-of & = & \{(w_1, b_1), (w_1, b_2), \\
 & & (b_3, w_1), (b_4, w_1)\}
\end{array}
$$

(b) Sets and relations defined for the world model

Figure 2.3: World model for the example

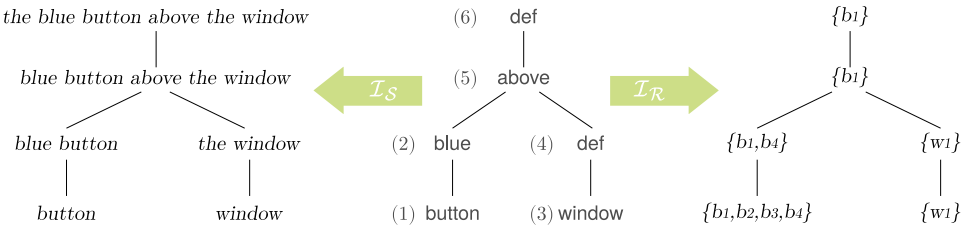



Figure 2.4: Example derivation tree (center) with the string interpretations of every subtree on the left and the relational interpretations of every subtree on the right.

$$
\begin{array}{ll}
 & proj_1((\text{above-of} \cap_1 \{b_1, b_4\}) \cap_2 \{w_1\}) \\
= & \text{<Expansion of } above-of\text{>} \\
 & proj_1((\{(b_1, w_1), (b_2, w_1), (w_1, b_3), (w_1, b_4)\} \cap_1 \{b_1, b_4\}) \cap_2 \{w_1\}) \\
= & \text{<Select all pairs in } above-of \text{ with first component } b_1 \text{ or } b_4\text{>} \\
 & proj_1(\{(b_1, w_1)\} \cap_2 \{w_1\}) \\
= & \text{<Select all pairs of elements whose second component is } w_1\text{>} \\
 & proj_1(\{(b_1, w_1)\}) \\
= & \text{<Project the first element of every pair in the set>} \\
 & \{b_1\}
\end{array}
$$

Just like rule (4), rule (6) ensures that the subtree to which it is applied contains a single element. The result of applying this final rule is the string interpretation "the blue button above the window", and the relational interpretation $\{b_1\}$.

To make the generation process more efficient, it is possible to build a chart such that it represents all REs for a given target in a compact way. A chart is "a packed data structure which describes how larger syntactic representations can be recursively built from smaller ones" (Engonopoulos and Koller, 2014). While chart-based algorithms are common in parsing

$\mathrm{N}_{b_1}/\{b_1,\ldots,b_4\} \rightarrow \mathit{button}_{b_1}$
$\mathrm{N}_{b_1}/\{b_1\} \rightarrow \mathit{blue}_{b_1}(\mathrm{N}_{b_1}/\{b_1\})$
$\mathrm{N}_{b_1}/\{b_1,b_4\} \rightarrow \mathit{blue}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_4\})$
$\mathrm{N}_{b_1}/\{b_1,b_4\} \rightarrow \mathit{blue}_{b_1}(\mathrm{N}_{b_1}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_1}/\{b_1\} \rightarrow \mathit{leftof}_{b_1,b_2}(\mathrm{N}_{b_1}/\{b_1\},\mathrm{NP}_{b_2}/\{b_2\})$
$\mathrm{N}_{b_1}/\{b_1\} \rightarrow \mathit{leftof}_{b_1,b_2}(\mathrm{N}_{b_1}/\{b_1,b_4\},\mathrm{NP}_{b_2}/\{b_2\})$
$\mathrm{N}_{b_1}/\{b_1\} \rightarrow \mathit{leftof}_{b_1,b_2}(\mathrm{N}_{b_1}/\{b_1,\ldots,b_4\},\mathrm{NP}_{b_2}/\{b_2\})$
$\mathrm{NP}_{b_1}/\{b_1\} \rightarrow \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1\})$
$\mathrm{NP}_{b_1}/\{b_1\} \rightarrow \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_4\})$
$\mathrm{NP}_{b_1}/\{b_1\} \rightarrow \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_2}/\{b_1,\ldots,b_4\} \rightarrow \mathit{button}_{b_2}$
$\mathrm{N}_{b_2}/\{b_2\} \rightarrow \mathit{yellow}_{b_2}(\mathrm{N}_{b_2}/\{b_2\})$
$\mathrm{N}_{b_2}/\{b_2,b_3\} \rightarrow \mathit{yellow}_{b_2}(\mathrm{N}_{b_2}/\{b_2,b_3\})$
$\mathrm{N}_{b_2}/\{b_2,b_3\} \rightarrow \mathit{yellow}_{b_2}(\mathrm{N}_{b_2}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_2}/\{b_2\} \rightarrow \mathit{rightof}_{b_2,b_1}(\mathrm{N}_{b_2}/\{b_2\},\mathrm{NP}_{b_1}/\{b_1\})$
$\mathrm{N}_{b_2}/\{b_2\} \rightarrow \mathit{rightof}_{b_2,b_1}(\mathrm{N}_{b_2}/\{b_2,b_3\},\mathrm{NP}_{b_1}/\{b_1\})$
$\mathrm{N}_{b_2}/\{b_2\} \rightarrow \mathit{rightof}_{b_2,b_1}(\mathrm{N}_{b_2}/\{b_1,\ldots,b_4\},\mathrm{NP}_{b_1}/\{b_1\})$
$\mathrm{NP}_{b_2}/\{b_2\} \rightarrow \mathit{def}_{b_2}(\mathrm{N}_{b_2}/\{b_2\})$
$\mathrm{NP}_{b_2}/\{b_2\} \rightarrow \mathit{def}_{b_2}(\mathrm{N}_{b_2}/\{b_2,b_3\})$
$\mathrm{NP}_{b_2}/\{b_2\} \rightarrow \mathit{def}_{b_2}(\mathrm{N}_{b_2}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_3}/\{b_1,\ldots,b_4\} \rightarrow \mathit{button}_{b_3}$
$\mathrm{N}_{b_3}/\{b_3\} \rightarrow \mathit{yellow}_{b_3}(\mathrm{N}_{b_3}/\{b_3\})$
$\mathrm{N}_{b_3}/\{b_2,b_3\} \rightarrow \mathit{yellow}_{b_3}(\mathrm{N}_{b_3}/\{b_2,b_3\})$
$\mathrm{N}_{b_3}/\{b_2,b_3\} \rightarrow \mathit{yellow}_{b_3}(\mathrm{N}_{b_3}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_3}/\{b_3\} \rightarrow \mathit{leftof}_{b_3,b_4}(\mathrm{N}_{b_3}/\{b_3\},\mathrm{NP}_{b_4}/\{b_4\})$
$\mathrm{N}_{b_3}/\{b_3\} \rightarrow \mathit{leftof}_{b_3,b_4}(\mathrm{N}_{b_3}/\{b_2,b_3\},\mathrm{NP}_{b_4}/\{b_4\})$
$\mathrm{N}_{b_3}/\{b_3\} \rightarrow \mathit{leftof}_{b_3,b_4}(\mathrm{N}_{b_3}/\{b_1,\ldots,b_4\},\mathrm{NP}_{b_4}/\{b_4\})$
$\mathrm{NP}_{b_3}/\{b_3\} \rightarrow \mathit{def}_{b_3}(\mathrm{N}_{b_3}/\{b_3\})$
$\mathrm{NP}_{b_3}/\{b_3\} \rightarrow \mathit{def}_{b_3}(\mathrm{N}_{b_3}/\{b_2,b_3\})$
$\mathrm{NP}_{b_3}/\{b_3\} \rightarrow \mathit{def}_{b_3}(\mathrm{N}_{b_3}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_4}/\{b_1,\ldots,b_4\} \rightarrow \mathit{button}_{b_4}$
$\mathrm{N}_{b_4}/\{b_4\} \rightarrow \mathit{blue}_{b_4}(\mathrm{N}_{b_4}/\{b_4\})$
$\mathrm{N}_{b_4}/\{b_1,b_4\} \rightarrow \mathit{blue}_{b_4}(\mathrm{N}_{b_4}/\{b_1,b_4\})$
$\mathrm{N}_{b_4}/\{b_1,b_4\} \rightarrow \mathit{blue}_{b_4}(\mathrm{N}_{b_4}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{b_4}/\{b_4\} \rightarrow \mathit{rightof}_{b_4,b_3}(\mathrm{N}_{b_4}/\{b_4\},\mathrm{NP}_{b_3}/\{b_3\})$
$\mathrm{N}_{b_4}/\{b_4\} \rightarrow \mathit{rightof}_{b_4,b_3}(\mathrm{N}_{b_4}/\{b_1,b_4\},\mathrm{NP}_{b_3}/\{b_3\})$
$\mathrm{N}_{b_4}/\{b_4\} \rightarrow \mathit{rightof}_{b_4,b_3}(\mathrm{N}_{b_4}/\{b_1,\ldots,b_4\},\mathrm{NP}_{b_3}/\{b_3\})$
$\mathrm{NP}_{b_4}/\{b_4\} \rightarrow \mathit{def}_{b_4}(\mathrm{N}_{b_4}/\{b_4\})$
$\mathrm{NP}_{b_4}/\{b_4\} \rightarrow \mathit{def}_{b_4}(\mathrm{N}_{b_4}/\{b_1,b_4\})$
$\mathrm{NP}_{b_4}/\{b_4\} \rightarrow \mathit{def}_{b_4}(\mathrm{N}_{b_4}/\{b_1,\ldots,b_4\})$
$\mathrm{N}_{w_1}/\{w_1\} \rightarrow \mathit{window}_{w_1}$
$\mathrm{NP}_{w_1}/\{w_1\} \rightarrow \mathit{def}_{w_1}(\mathrm{N}_{w_1}/\{w_1\})$

Figure 2.5: Chart containing derivations for the object $b_1$. This chart is based on a simplified grammar that does not include [above/below]-of relations between objects.

and realization tasks, this algorithm is the first one to use them for REG.

The computed chart contains all derivations of the grammar corresponding to a given set and, by extension, the set of all possible REs that the grammar can generate for this set. The chart is represented as a finite tree automaton (Comon et al., 2007), which is written here as a Context-Free Grammar whose strings correspond to REs for the given set. An example chart is given in Figure 2.5, showing an example for the world model from Figure 2.3 where the above/below relations were removed for simplicity.

Nonterminals in a chart are of the form $\mathrm{N}_b/\{a_1,\ldots,a_n\}$. Each nonterminal symbol consist of three parts: a syntactic category given by the synchronous grammar ($N$), the referent for which the RE is currently being constructed ($b$), and the set of objects $\{a_1,\ldots,a_n\}$ to which the entire subtree refers.

**Example: Derivation tree for the RE "the blue button"**

To generate the RE "the blue button" we follow a bottom-up process. Our first selected rule is:

$$\mathrm{N}_{b_1}/\{b_1,\ldots,b_4\} \rightarrow \mathit{button}_{b_1}$$

The nonterminal symbol of this rule denotes the syntactic category "Noun" (N), the referent for which the RE is being constructed ($b_1$), and the set of objects that this subtree actually refers to ($\{b_1,\ldots,b_4\}$). That is: this rule was created while constructing an RE for $b_1$, but the sub-tree built by this derivation rule (whose string interpretation would be "button") *actually* refers to the set $\{b_1,\ldots,b_4\}$ of all buttons. Next, we apply the following rule:

$$\mathrm{N}_{b_1}/\{b_1,b_4\} \rightarrow \mathit{blue}_{b_1}(\mathrm{N}_{b_1}/\{b_1,\ldots,b_4\})$$

This rule takes as input a Noun intended for $b_1$ (but actually referring to $\{b_1,\ldots,b_4\}$) and returns an RE referring to $\{b_1,b_4\}$. It's string interpretation would be "blue button". We apply the third and last rule:

$$\mathrm{NP}_{b_1}/\{b_1\} \rightarrow \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_4\})$$

This rule generates a Noun Phrase (NP), is intended for $b_1$, and returns an RE that refers to $b_1$. The string interpretation of this final tree is, as expected, "the blue button".

In addition to showing that a chart of valid REs can be computed, Engonopoulos and Koller (2014) introduce a Viterbi-based algorithm for computing REs that maximize the probability of being correctly understood. This probability is obtained modeling the probability $P(b|t)$ that a listener

will resolve the RE $t$ to the object $b$ with a log-linear model.

The log-linear model is defined in term of feature functions $f(a, t, M)$ representing properties of an object $a$ in the context of a derivation tree $t$ and a world model $M$. For example, a feature $f_{round}(a, t, M)$ could return 1 if the root label of $t$ is $round_a$ and $a$ is round in $M$. Each one of these features look at information that is local to a specific subtree of the RE and represent varied properties of objects in the current scene, such as whether an object is of a certain shape, it is located to the left of another object, or whether the user is looking in its direction.

## 2.4 Virtual Environments

Informally, an *environment* is the context in which a certain action takes place. Actions and context are in constant interaction: the context limits which actions can take place at any given time, and the result of an action is a modified context. Formally, an environment is defined as a tuple of 3 elements $(\mathcal{A}, \mathcal{S}, f)$, where $\mathcal{A}$ is a set of **actions** that may occur at any time, $\mathcal{S}$ is a set of **states**, and $f : \mathcal{A}x\mathcal{S} \rightarrow \mathcal{S}$ is a **transition function** that applies an action to a given state and returns a new, modified state[3]. The transition function $f$ does not necessarily accept all actions in all states. If an action is possible in a given state, we say the action is **affordable**. Deciding how best to go from one state to a new one is what we call a *planning* problem, a topic discussed in detail in Chapter 2.2.

**Example: Actions in an Environment**

Imagine a subject inside a dark room, standing close to a light switch. Their position inside the room, the light level, and the light switch's position are all components of the *state*. Pressing the light switch is a possible *action*, and the resulting *transition* would be to a state where the light switch is pressed and the room is no longer dark.

If the switch were outside the subject's reach, we would say that the action `press_button` is not *affordable* in the current state. A plan to achieve the intended state "make the room not dark" could involve walking towards the light switch, fetching a long stick, or opening a curtain.

A *Virtual Environment* (VE) is any non-physical environment retaining the properties of an environment. Interactive 3D worlds are an intuitive example, but word processors, websites, and role-playing games are also members of this category. VEs are preferred by researchers because they significantly lower the barrier of entry that data collection and testing require in the physical world. A well-designed VE allows researchers to

[3]For a more detailed definition of Environment, Branavan et al. (2009) presents a formalization suited for Reinforcement Learning tasks

closely replicate observed behavior in the real world, making their results generalizable to the world at large while being simpler to develop, test, and deploy.

Since VEs retain the critical properties of the Environment they are abstracting, MacMahon and Stankiewicz (2006) suggest classifying them according to the same guidelines used to classify real, physical spaces. A systematic, unified categorization of physical spaces was introduced by Freundschuh and Egenhofer (1997), classifying spaces based on whether objects in the space can be manipulated, whether the space requires locomotion to be experienced, and the size of the space.

**Example: Environment vs Virtual Environment: word processors**

A typewriter presents a physical environment: the position of the paper and the characters on it determine the state, pressing a key performs an action, and the result of pressing a key is a modified state with either an extra printed letter or a new position for the cursor on the paper. It's virtual counterpart, the word processor, behaves by design in the same way that a typewriter would, but abstracts away problems such as paper jams or empty ink ribbons. Correcting errors is also easier.

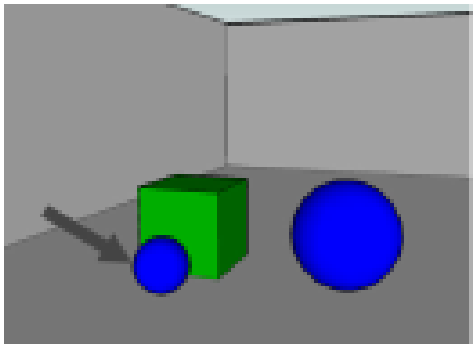
In the classification of Freundschuh and Egenhofer (1997), the VE "word processor" presents a small-scale, manipulable object space where no locomotion is required.

Virtual Environments have aided research for a long time and in a large array of tasks. They are not typically introduced by themselves, but rather in the context of a specific task or objective.

Small-scale VEs are of great help for experiments analyzing single, specific phenomenon. Fernández and Schlangen (2007) uses virtual Pentomino boards as a VE to explore the type of referring expressions generated in a scenario of limited interactivity, while Clark and Wilkes-Gibbs (1986) designed a collaborative task over figures from the Tangram board game to show how the presentation of an object must be accepted by all parties in a dialogue before it can move forward. Both Lau et al. (2009) and Branavan et al. (2009) use web pages as small-scale VEs to present strategies for the automatic interpretation of human-written instructions. The former models written how-to instructions as actions over the environment, while the latter applies automatic instruction understanding to troubleshooting guides.

In the context of language generation tasks, classical examples of small-scale VEs are the visual domain of the GRE3D corpus and the TUNA corpus (Viethen and Dale, 2008; van der Sluis et al., 2007). GRE3D represents simple geometrical shapes in a 3D environment, and it was designed to evaluate the generation of Referring Expressions that require the use of referential attributes w.r.t. other objects on the scene. Figure 2.6 shows a

sample image.



| | Objects | | |
|---|---|---|---|
| **Attrib** | $d_1$ | $d_2$ | $d_3$ |
| Color | blue | green | blue |
| Shape | ball | cube | ball |
| Size | small | large | large |
| Rel | $bef(d_2)$ | $beh(d_1)$ | $nt(d_2)$ |

(a) Image from the visual domain of the GRE3D corpus

(b) Objects and their attributes for the sample image

Figure 2.6: Visual domain of the GRE3D corpus and its corresponding attributes. $bef$="before", $beh$="behind", $nt$="next to".

TUNA makes the task more challenging, introducing corpora containing complex objects (such as furniture and/or pictures of people), more distractors per scene, and scenes in which the intended RE refers not to a single object but rather to a set. More recent work explores the description of objects in realistic visual scenes, and/or at a larger scale. Figure 2.7 shows examples from the people and furniture domains.

(a) Furniture domain. Properties defined for each object in the scene: x-dimension, y-dimension, color, orientation, type, size.
Reference RE for selected targets: "the fan and couch on bottom row".

(b) People domain. Properties per object: x-dimension, y-dimension, age, orientation, hairColour, hasSuit, hasShirt, hasTie, hasBeard, hasGlasses, hasHair.
Reference RE for selected target: "a man with glasses".

Figure 2.7: Images and attributes from the TUNA corpus domains

As for medium-sized VE, Wu et al. (2018) introduce an interactive 3D environment based on human-designed indoor scenes, and which was em-

ployed by Das et al. (2017) to design an embodied QA task. In this task, an agent must navigate the VE and gather information in order to answer questions such as "What color is the car?" VEs presenting immersive experiences in medium-sized, familiar environments have been used to treat phobias such as fear of spiders, heights, or flying (Glantz et al., 1997). These environments present subjects with tasks in which they were asked to pick up virtual spiders in a virtual kitchen, walk across bridges of varying heights and stability, or virtual flying under different turbulence conditions. Ilinykh et al. (2018) presents a collaborative task in which two players navigate a virtual house, with the current room being shown as a photograph and where participants must indicate once they both believe to be in the same room. With more general-purpose research in mind, and on the larger side of the spectrum, Bülthoff and van Veen (2001) developed a virtual representation of the city of Tübingen for the study of human perception in an immersive VR environment. This VE can be explored freely, and contains texture maps for over 700 individual houses.

In the specific context of VEs for Instruction Following tasks, Matuszek et al. (2010) present a system that learns how to follow navigational instructions in maps of real environments created with laser range-finder data, while Chen and Mooney (2011) introduce a similar approach that improves on previous work by leveraging landmark information. Their approach learns how to interpret navigational instructions on a map from pair of recorded instructions and map traces, with no prior linguistic knowledge. For a modern extension of this task, de Vries et al. (2018) collected 10k human-human dialogues in which a "tourist" and a "guide" navigate towards a specific location in a 360 degrees, photographic reconstruction of New York neighborhoods.

Finally, and in the context of research about multi-agent collaboration in situated tasks, Byron and Fosler-Lussier (2006) presented annotated, multimodal corpora of human partners collaborating on a treasure-hunt style task. This corpora, collected using the Quake engine, contains recordings in which a participant must guide another through a sequence of tasks. The virtual environment comprises around 15 rooms and 2 staircases, and the task required changing the position of seven objects within the virtual world. The released corpora contains two movies (one from the point of view of each participant) and transcriptions of the audio. This research would be later expanded by Stoia et al. (2008) when releasing the SCARE corpus, a similar corpus containing the necessary required resources and aligned data for researchers to fully recreate (rather than just observe) the games in their own computers. Figure 2.8 shows an example of the corpus from a player's point of view.

With the proliferation of human-human task data recorded in VEs, the next step would be the design a VE where researchers could test and compare automated systems. An influential test bed for this type of research,

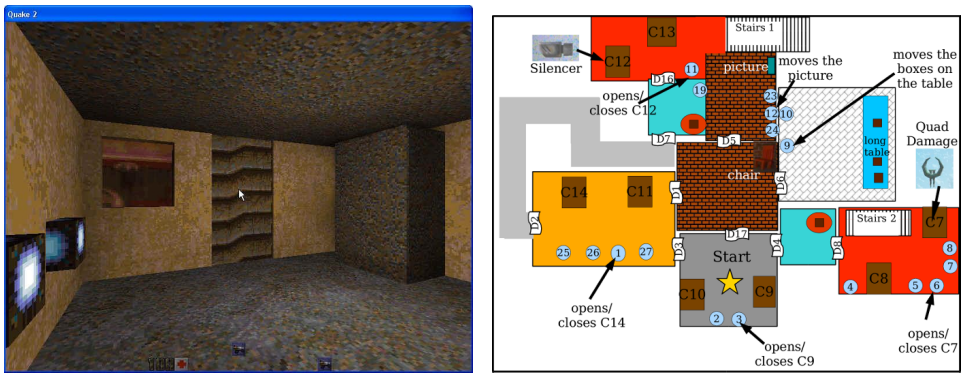



Figure 2.8: SCARE Corpus - First-Person view and upstairs floor map

and the main VE in this thesis, would be provided by the GIVE Challenge.

## 2.5 The GIVE Challenge

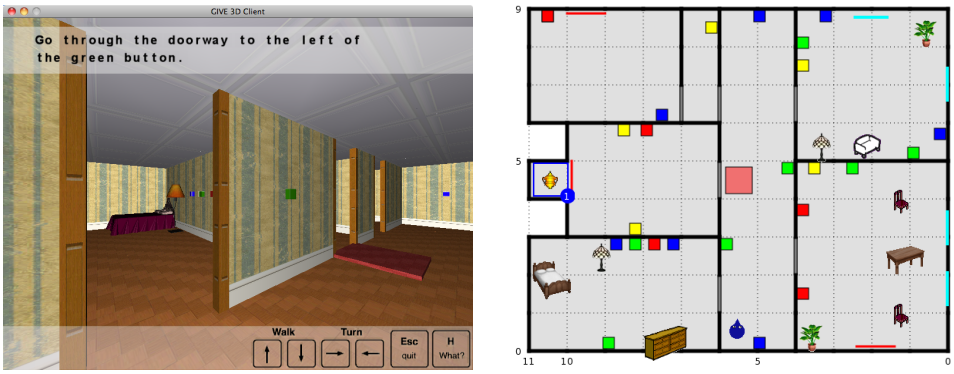
GIVE (Generating Instructions in Virtual Environments) is the name of the VE designed for the GIVE Challenge, an NLG task in which a human Instruction Follower (IF) is paired with an automated Instruction Giver (IG) in a maze-like 3D world (Koller et al., 2010).

GIVE presents a medium-scale indoor environment where players can interact with buttons, doors, and alarms. The IF has a 3D view of the environment, can move forwards and backwards, and is allowed to turn in 360 degrees[4]. The IG does not have access to the IF's point of view, having access instead to a full representation of the environment that includes the layout of the space, the effect that pressing buttons has on the environment as a whole, and the current position and angle of the IF.

To successfully complete a *game* in the GIVE Challenge, the IF and IG must cooperate to retrieve a trophy locked inside a safe. This safe can only be unlocked pressing certain buttons in a specific order, but each button presents its own set of challenges: some of them are behind alarms and/or closed doors, and pressing a wrong button can cause backtracking or even an instant loss. Walking over an active alarm also causes the players to lose the game. The IG communicates with the IF via written instructions, and the IF can request help pressing the 'H' key. Figure 2.9 shows both the design of a GIVE world and a first-person view.

Although the GIVE Environment supports data collection from human-human pairs, the main objective of the Challenge is to test whether automated IG systems are capable of guiding humans in the role of the IG. For this purpose, each instance of the Challenge connected human IFs all over

[4]Except in GIVE-1, where users move and turn in discrete steps




(a) First-person view (b) Map view

Figure 2.9: First-person and Map view of a GIVE world from the 2011 evaluation worlds (Striegnitz et al., 2011).

| | **GIVE** (Byron et al., 2009) | **GIVE 2** (Koller et al., 2010) | **GIVE 2.5** (Striegnitz et al., 2011) |
|---|---|---|---|
| Year | 2008-09 | 2009-10 | 2011 |
| IG Systems | 5 | 7 | 8 |
| Games | 1143 | 1825 | 536 |
| Countries | 48 | 39 | 34 |

Table 2.1: Statistics of the GIVE Challenge data

the world and matched them with automated IG systems developed by various research teams. The data recorded in each game contains the position and angle of the IF every few ms, the state of the world at every step, and the time and text of each instruction that the IG presented to the IG. Table 2.1 shows the main statistics about the data collected on these Challenges.

To participate in the Challenge, human IFs must first download the *GIVE Client*. This client will contact the *Matchmaker*, a service that connects human IFs and automated IGs with each other, selects a world for the task, and records all interactions A more technical description of the entire pipeline is presented in Chapter 6.2.

Several problems must be solved to develop a mature, automated IG system for GIVE. Selecting the next best step requires careful automated planning, guiding the IF requires navigational instructions, and identifying a single object out of many distractors demands crafting precise REs.

## 2.6 The pragmatics of dialogue, misunderstandings, and focus

In his book *Using Language*, Clark (1996) affirms that language is used for "doing things". One specific way in which things can get done is through a *dialogue*, defined as "a discussion between representatives of parties to a conflict that is aimed at resolution".This conflict can be as trivial as figuring out what time is it, or as complex as a peace treaty. In contrast to a *conversation* (defined as "an oral exchange of sentiments, observations, opinions, or ideas"), a dialogue has an explicit purpose and is not restricted to a spoken setting. Under Clark's framework a dialogue, like all instances of language use, is a form of *joint action* in which two or more people must coordinate in order to move their joint task closer to a resolution through their individual actions. To understand how this coordination takes place we turn to the theory of *illocutionary acts*, introduced by Austin (1962) and later expanded by Searle. Under this theory, a speaker typically performs at least three different kinds of acts: *utterance acts* (morphemes, sentences), *propositional acts* (referring and predicating), and *illocutionary acts* (stating, questioning, commanding, etc) Searle (1969). It follows that sentences, rather than words, is where our focus should be if we are interested in how a joint action is moved forward — phonemes and individual words are not enough to understand how an intention is communicated, because "only in the context of a sentence do words have meaning"(Frege, 1884). In speaking, notes Searle,

> I attempt to communicate certain things to my hearer by getting him to recognize my intention to communicate just those things. I achieve the intended effect on the hearer by getting him to recognize my intention to achieve that effect, and as soon as the hearer recognizes what it is my intention to achieve, it is in general achieved.

We turn to Clark's writing on construals to explain how a listener recognizes a speaker's intention. In social psychology, construals are "how individuals perceive, comprehend, and interpret the world around them, particularly the behavior or action of others towards themselves" (Gilovich et al., 2015). Clark asserts that social events are hard to construe:

> I see a fish, and I construe it as a fish, as a trout, or as food for a grizzly bear. Social events aren't always so easy. I see a strange man walking toward me. Is he approaching me by accident, or by design? Does he want to ask me directions, rob me, or what?

To reach a joint construal between speaker and listener it is often useful for each party to display their own construals publicly via signals such as

verbal confirmation, congratulations, apologies, etc. More often, construals are displayed by the next step each person takes in the social process they are engaged in. If the listener displays a clear understanding of the speaker's proposal, they have accepted the construal and achieved a joint, verified construal. If the listener reacts in an unexpected way, the speaker can detect that the listener has misconstrued their proposal and react accordingly with a correction, an acceptance of the revised construal, or by narrowing the original construal. The misconstrual might also go undetected, in which case the error may or may not be caught later on.

While this process takes place, both interlocutors establish a *common ground*, a shared belief about information they are both aware of. For something to be part of the common ground the speaker must be aware of it, the listener must be aware of it, and the speaker must be aware that the listener is aware of it (and vice-versa). The process of establishing a common ground is known as *grounding*.

### 2.6.1 Misunderstandings

If a single misconstrual goes undetected, we say that a misunderstanding took place. One source of information for detecting such misconstruals is the mutual beliefs of the interlocutors about the current situation. Both speaker and listener share common knowledge that allows them to understand each other such as language, laws, and social norms.

A speaker can spot misconstruals by matching a publicly displayed signal against the common ground: if the speaker offers the listener a drink, and the listener responds by saying "thanks" while showing his open palm, it should be clear to the speaker that the listener is declining the offer in a polite manner: it is part of their common ground that an open hand is a symbol of rejection, and they are both aware that each other knows this. At the same time, a mismatched common ground can also be a source of misconstruals: a speaker from a different culture where this signal is unknown would probably assume that the listener is accepting the drink, since they have not uttered an explicit rejection. The listener assumed incorrectly that the hand signal was in their common ground, and now the dialogue must be repaired.

One specific type of misconstrual is the unsuccessful recovery of intended referents. Whenever a speaker refers to an entity, they can do so in a myriad of ways: the speaker could use a definite expression ("Gabriel García Marquez"), anaphoric pronouns("him"), noun phrases ("the writer"), and so on. Following Searle, this work uses the term *referring expression* (RE) as short for "any expression which serves to identify any thing, process, event, action, or any other kind of 'individual' or 'particular' "Searle (1969)[5].

[5]Searle writes that "it is by their function, not always by their surface grammatical form

### 2.6.2 Focus

No matter what type of RE the speaker chooses, they believe that the listener will correctly infer who the intended referent is in accordance with Grice's maxim of quantity Grice (1975). For the intended referent of a RE to be successfully recovered the search space must first be reduced to a small set of alternatives. The search can be further directed exploiting the salience of specific discourse referents, what we call the *focus structure* of the utterance.

Sanford and Garrod (1981); Garrod and Sanford (1982) introduce the term *focused memory* to refer to short-term/working memory, whose contents are similar to a set of discourse referents where some elements are more prominent than others. The more prominent an element is, the more ambiguous an RE can be: a highly prominent individual can be identified by an ambiguous anaphoric personal pronoun such as "he", while less prominent individuals might require full definite noun phrases such as "my older brother".

The elements present in what Sanford and Garrod call *focused memory* can be mapped to what Smith and Lieberman (2013) calls a *context set*, namely, the viable candidates for an interpretation process, which evolves over the course of dialogue. An in-depth discussion of short-term memory from a psycholinguistics perspective can be found in Chapter 5, and a detailed analysis of context sets is presented in Chapter 7.

Another factor in this referring process is the use of stress. It has been long documented that stress and intonational markers affect how REs are resolved Akmajian and Jackendoff (1970). The question of *how* does markedness help a listener recover an intended referent is approached by Bosch (1988) who suggests that, in general, markedness is a property of those choices that deviate from the default choice in a given situation. This is not the only use of markedness, according to Bosch: in a scenario where there is a unique expectable referent, markedness can be used to confirm that this is indeed the intended one.

It is clear that markedness is a device that a speaker can use to prevent misconstruals in REs. If a speaker believes that a listener will interpret their next utterance in a way that they do not intend, they can add stress to the utterance and direct the listener's focused memory in the correct direction

## 2.7 Conclusion

This Chapter introduced the task of Instruction Following, presenting the required background that we need to understand what a typical task looks

or their manner of performing their function, that referring expressions are to be known", but refines this definition further to "singular definite expression used for referring to particulars". This work follows the same convention.

like, which roles are involved, and how are common sub-tasks performed. Taking our early example, "a daughter explains to his father over the phone how to use the printer", we can now say that ...

- ... the daughter takes the role of Instruction Giver, and the father takes the role of Instruction Follower.
- ... the task takes place in a Virtual Environment, where objects can be manipulated (i.e., buttons can be clicked), the space requires locomotion (i.e., the IF must "navigate" to specific windows), and the space is of considerable size (i.e., the sum of all possible windows that the IF could visit).
- ... the task requires careful planning: before selecting the correct printer from a list, the IF must open the window listing all printers.
- ... the IG must carefully choose descriptions that the IF can understand: whether "click the printer icon" is more effective than "click the third button from left to right in the toolbar" depends on the task at hand, the experience of the IF, and the common ground between both. In either case, both REs are more likely to be successful for this task than "click the button that's 31mm to the left and 26mm from the top".
- ... the IG must account for misunderstandings: if the IF clicks on the "Export" button instead of the "Print" button, then it's the IF's job to detect that a misunderstanding took place, decide on a plan to correct for it, and present the new plan to the IF. Even better, the IF should account for the similarity between both buttons and minimize the possibility of a misunderstanding to begin with.

With this frame of reference in mind, we now have a clearer picture of all the components that are presented in this thesis: generating effective Referring Expressions that maximize the probability of being correctly understood, detecting whether a misunderstanding took place, and generating contrastive REs to correct these misunderstandings are the building blocks of an automated IG system for (Navigational) Instruction Following in Virtual Environments.

Going from theory to practice, the next Chapter introduces our first main research results built around a simple question: is following instructions really *that* hard? And if so, *why*?

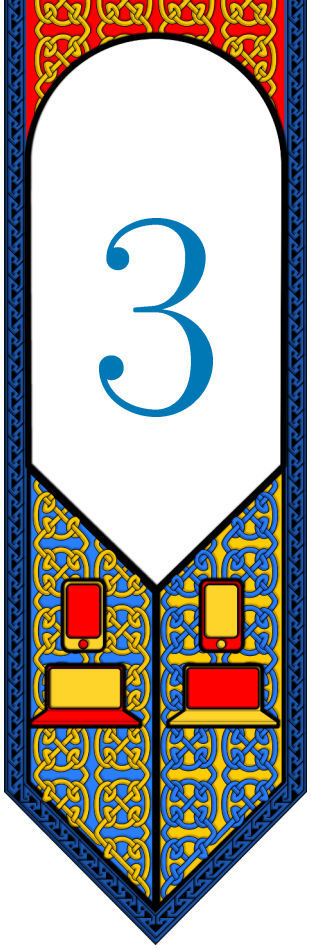
## 2.8 Further reading

For a larger overview of Virtual Environments, van Deemter (2016) presents a detailed analysis of several VEs in the context of different Referring Ex-

pression generation tasks.

Detailed surveys on Natural Language Generation are regularly published and updated. Reiter and Dale (1997, 2000) are the most cited surveys to date, while Gatt and Krahmer (2018) present an updated and expanded view on the topic.

Each instance of the GIVE Challenge has released detailed reports on the task, the performance of individual systems, and aggregated data from the competition (Byron et al., 2009; Koller et al., 2010; Striegnitz et al., 2011). In addition to this aggregated data, each system is accompanied by its own paper with developer insights.



# 3

# Following instructions

In an ideal world, a Referring Expression that uniquely identifies a target would never cause a misunderstanding. But Section 2.6 has shown us that this position is unrealistic: even if human language where not ambiguous, which it is, the process according to which common ground is established is rife with opportunities for misconstruals to occur undetected. We can easily picture the following situation: Max follows instructions in his phone to reach the library, and suddenly the instruction "go South" shows up. If Max takes a different path than the expected one, how difficult would it be to get him back on track? If he didn't pay attention to the question, then repeating it should be enough. Or maybe he doesn't know where "South" is, and we would prefer a reformulation. Perhaps we don't even need to say anything - how do we know that a simple "no, try again" wouldn't be enough?

Before we move forward with our plan of preventing misunderstandings, we need to focus on these questions to obtain a better sense of how hard the problem really is – in other words, we need to establish a baseline of what a reasonable interaction between Instruction Follower and Instruction Giver should be, and then focus in those areas where this baseline falls short. Presenting a system to take as our baseline is the purpose of this Chapter.

This Chapter is divided in three main Sections. The first Section is dedicated to explain the design choices that we took when designing a simple

This Chapter is based on the publications "Corpus-based Interpretation of Instructions in Virtual Environments" and "Interpreting Natural Language Instructions using Language, Vision, and Behavior" (Benotti et al., 2012, 2014).

Instruction Following system capable of following instructions and reacting to misunderstandings. When presented with an instruction, this system compares the instruction to recorded human behavior data and selects the previously-observed behavior that more closely matches the current situation. How to formalize "previously-observed behavior" is the main topic of Section 3.2.1, while several approaches to what it means to be a "close match" are presented in Section 3.2.2. Keeping this approach simple, this system can be implemented in several Virtual Environments in a straightforward way and requiring little to no manual data annotation.

The second Section of this Chapter details the results we obtained after applying this system to data collected from the GIVE Challenge. Results show that our system provides competitive accuracy to human annotators, supporting our choice of this system as a baseline for the study of misunderstandings.

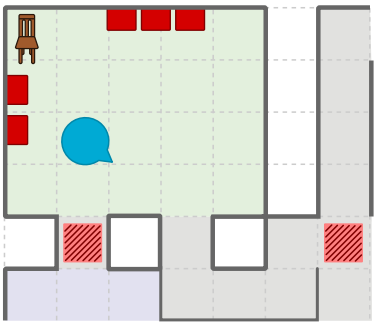
Finally, and closing this Chapter, we take a deep look at those scenarios in which our baseline system fails and identify its weak points. Identifying the scenarios in which our baseline system has no difficulties provides us with a foundation for the types of difficult problems that a simple approach cannot solve. Performing an error analysis on our system gives us a chance to understand why this system cannot be extended to other domains without significant changes, what should be changed, and which assumptions can we make regarding instruction following misunderstandings. These assumptions will accompany us throughout the remaining chapters.

## 3.1 A simple strategy for instruction following

We intend to create a simple Instruction Follower (IF) that can both understand instructions and recover from misunderstandings. The key idea for our approach is straightforward: given a corpora containing instructions and the subsequent recorded human reactions, we first classify them according to *what was said*, in *which location*, and *with which result*. Once this data was processed, our automated IF is ready: whenever the IF receives an instruction, it searches for the pair (instruction, location) that more closely matches a previous interaction, and reacts in the same way the human IF did. Similar approaches has been successfully applied in NLP tasks, and it is considered a good approach for fast and semi-automated prototyping of Dialog systems Chen and Mooney (2011); Haponchyk et al. (2018).

**Example: Underspecified instructions**

Imagine there are two Instruction Followers (IF) —one human, one automated. They are both inside a room with several color buttons, trying to reach a specific goal. The situation is illustrated as follows:



The human IF attempts the task first: standing in the room, they receive the (somewhat underspecified) instruction *"red left of chair"*. They press the button that's located to the left of the chair from their point of view, and they can now move forward to the next step of their task.

Now it's the automated IF's turn. This IF is standing in the same position where the human IF was and receives the same instruction. What should the automated IF do? Clearly, the automated IF should press the same button *even if there is no mention of a button in the instruction*, because that's what the human did after receiving the same instruction under the same circumstances.

## 3.2 Implementing our IF

Implementing our IF requires two major steps: identifying suitable data and annotating it automatically, and implementing the IF system that would make use of it.

### 3.2.1 Data collection and segmentation

Our first major task was obtaining a corpus of user interactions. This corpus should contain data from human IFs completing the tasks that we expect our IF to fulfill.

Given that we require no manual annotation, recording enough data to reconstruct all interactions that took place inside a VE is straightforward: our corpus must include the IF's position, the state of the VE at instruction time, and all interactions with the environment. Given that the IF acts in reaction to instructions, the content of these instructions should be also recorded.

It is not unreasonable to assume that the Instruction Giver's (IG) instructions follow a certain plan. While not strictly necessary, it is a good idea to also store the plan's objective for each step, as it will allow us to reason about the IF and IG's intentions. Asserting that *"according to the plan,*

*I must generate a instruction that leads the IF to X"* associates each individual interaction with a specific intention.

Once all data has been collected, it is split into *episodes*. An episode comprises a single interaction between IG and IF[1], and can be understood as a 4-tuple $(pos_0, world, inst, reaction)$, where

- *pos* is the position of the IF at instruction time
- *world* is the state of the world at instruction time
- *inst* is the text of the instruction
- *resp* is the reaction to the instruction

The "reaction" field *resp* requires that we make a decision regarding what should be considered the *canonical reaction* to an instruction. Going back to our example in Figure 3.1: imagine that, after receiving the instruction "go through the door on the left", the human IF went through the door, turned right, walked into the main hall, and stopped in front of a statue. How many of these steps were taken as a direct reaction to the instruction, and how many were taken by the IF on their own?

The answer to this question dictates how many of these interactions we must take into account, and how many should be discarded. In this work, we differentiate between two possible segmentation strategies: Visibility-based segmentation (Vis) and Behavior-based segmentation (Bhv).

The Vis segmentation strategy collects only the first action performed by the IF, and discards the rest. The argument is as follows: the first action performed by the IF is the most likely to have taken into account the context that was *visible* to the IF at instruction time. To imitate the recorded human behavior, our automated IF could be optimized to interpret instructions based only on features of the current, visible environment.

The Bhv segmentation strategy collects all actions performed by the IF, up until the beginning of the next episode.

Whether we decide to use Vis or Bhv, we need to define what an "action" is. In our approach we discretize the recorded behavior *resp* with the help of a planner. Using an automated planner and a planning representation of the task, the planner calculates the optimum sequence of states that connect the state of the world *world* at the beginning of the episode and the state of the world at the beginning of the next episode. In our example the planner would include steps such as "walk through the door" and "enter the Main Hall", but would remove superfluous steps such as "give one step forward", "turn 90 degrees", and so on.

Doing so, we arrive at the definition of a *canonical reaction cr*. If $S_k$ is the state of the world when instruction $i_k$ was presented to the IF, and $S_{k+1}$ is

[1]Note that later chapters use a similar notion of Episode, but it is not identical

the state of the world when the reaction ends, then the canonical reaction $cr$ to the instruction $i_k$ is either the sequence of actions determined by the planner that lead from state $S_k$ to state $S_{k+1}$ (if using Bhv) or the first action of the sequence (if using Vis). $cr$ substitutes $resp$ in our segmented data.

Using the pair $(pos, cr)$ as keys, we have now collected our training corpus for our IF system.

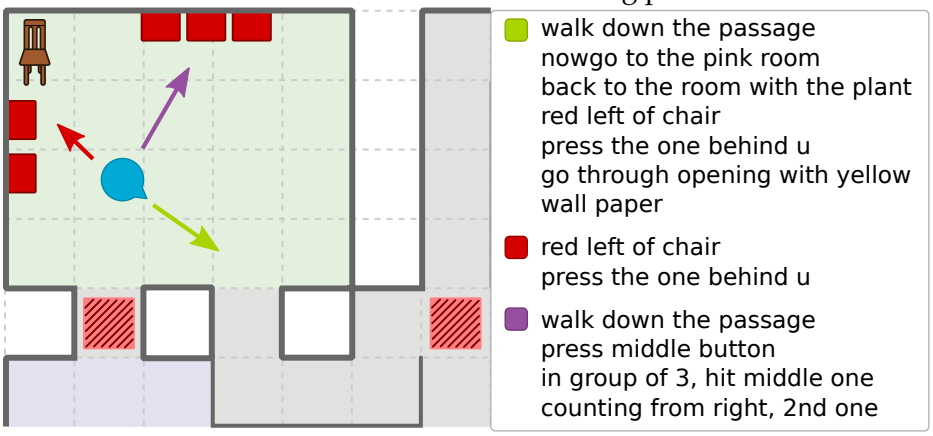
### 3.2.2 Interpretation

Our IF interprets instructions from the IG in the following way: once the IG presents an instruction $inst_k$, the IF located at position $pos_k$ in the VE retrieves from the corpus all episodes taking place in the same location $pos_k$. These episodes are then clustered together according to their canonical reaction $cr_k$, yielding clusters of the form $(pos_k, cr_k, I = \{inst_0, inst_1, \ldots\})$ where $I$ is the set of all recorded instructions $inst_k$ that where presented in location $pos_k$ and elicited a canonical reaction $cr_k$.

The IF must then decide which canonical reaction $cr_k$ is appropriate for the instruction $inst_k$, selecting the cluster with instructions that are the most similar to the current instruction.

**Example: Choosing an action cluster**

The IF receives the instruction $inst_k$="click the red button to the left of the chair" in the situation shown in the following picture:




When previous players encountered themselves in the current position $pos_k$, they did one out of three things: they left the room (1), they clicked a button out of a set of two (2), or they clicked a button out of a set of three (3). These are the possible canonical reactions for the current position. We would expect our IF to select cluster 2: although the literal text of the new instruction is not contained in this cluster, $inst_k$ is more similar to the instructions in this cluster than to those in clusters 1 and 3.

The IF algorithm can be described as follows:

**Require:** $instruction \neq$ ""
1: **procedure** INTERPRET(instruction, player_pos)
2: $clusters \leftarrow \{x : record \in corpora | x.position = player\_pos\}$
3: $best\_cluster \leftarrow \underset{x \in clusters}{\text{argmax}} similar(x.instructions, instruction)$
4: $perform(best\_cluster.reaction)$
5: **end procedure**

We now face the problem of deciding what the $similar(x, y)$ function should be. We treat the problem of deciding which set of instructions is most similar to the new instruction as a classification problem. We compared six different algorithms, divided in three different methods: word similarity measures (implementing Levenshtein, Jaccard, and Overlap measures), machine translation (using BLEU), and machine learning (implementing both Decision Trees and SVMs).

### 3.2.3 Group selection by word similarity

The first methods use nearest neighbor classification with three different similarity metrics.

Our first two metrics are Jaccard and Overlap. Both measure the degree of overlap between two sets, differing only in the normalization of the final value (Nikravesh and Bensafi, 2005). The Jaccard coefficient is defined as:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

while the Overlap coefficient is defined as:

$$Overlap(A, B) = \frac{|A \cup B|}{min(|A|, |B|)} \tag{3.2}$$

We obtained the similarity between two sentences by turning them first into bags of words.

The third metric implements the Levenshtein distance over words (Levenshtein, 1966). This metric segments both instruction into sequences of words and returns the minimum number of operations (insertion, deletion, and substitution) required to convert one sentence into the other. Although this metric can also be applied over characters rather than words, this alternative performed poorly in early tests and was discarded.

In all cases, the similarity score between an instruction *inst* and a cluster containing a set of instructions $I = \{inst_0, inst_1, \ldots\}$ is defined as the average similarity between *inst* and all $inst \in I$:

$$\frac{\sum_i coeff(inst, inst_i)}{|I|}, coeff \in \{\text{jaccard, overlap, levenshtein}\} \tag{3.3}$$

### 3.2.4 Group selection with machine translation methods

We also approached the group selection problem from a machine translation perspective, using BLEU as a score function (Papineni et al., 2002). Given an input sentence $S_I$, BLEU evaluates how good a specific translation $T_{new}$ of $S_I$ is in relation to a set of reference translations $T = \{T_1, \ldots, T_n\}$ of $S_I$. Informally, we can imagine that $T$ is a set of human translations of $S_I$, $T_{new}$ is a translation generated by a machine translation system, and BLEU indicates how "good" the translation $T_{new}$ is when compared to previous ones. BLEU computes an n-gram overlap similarity between $T_{new}$ and the set of reference translations according to the equation

$$\begin{aligned} BLEU_n &= \frac{\sum_{t \in T} \sum_{n-gram \in t} Count_{clip}(n-gram)}{\sum_{t' \in T} \sum_{n-gram' \in t'} Count(n-gram')} \\ \text{where } Count_{clip} &= min(count, \text{max count of this n-gram} \\ &\qquad \text{in any reference text}) \end{aligned} \tag{3.4}$$

For our system, we model the situation assuming that all instructions $inst_k$ in a cluster are reference translations $T$ of the same unknown sentence $S_I$, and the new instruction $inst$ is our new candidate translation $T_{new}$. The cluster that achieves the highest BLEU score is considered as the cluster that better correlates with both the new sentence and our hypothetical unknown sentence.

BLEU has been shown to correlate well with human judgment for automatic translation. It has been used in several areas of NLP and, despite criticisms regarding its appropriateness as a metric in modern NLP, remains one of the most popular metrics for automatic evaluation (Sulem et al., 2018).

### 3.2.5 Group selection with machine learning

We also approached this classification problem from a machine learning (ML) perspective. We modeled our data as a classification problem, where data features include the IF's position, visible areas, and a bag-of-words representation of the newest instruction, and the output class is the canonical reaction. This setup provides these classifiers with more fine-grained data than in previous approaches, and therefore we expect them to perform better.

We trained both a Support Vector Machine (SVM) with a radial kernel (Cortes and Vapnik, 1995) and a decision tree (DT) classifier with the C4.5 algorithm (Murthy, 1998). The former is provided by the LIBSVM package (Chang and Lin, 2011), while the latter was provided by the WEKA workbench under the name "J48" (Hall et al., 2009).

An SVM is a supervised classifier that finds the maximum-margin boundary between classes in a binary classification problem. If the input data is linearly separable, SVMs identify the hyperplane that keeps the maximum separation or *margin* between the two data classes. SVMs can be trivially extended to a multiclass classification problem employing $n$ classifiers in a one-vs-all configuration. More important, SVMs can be used for non-linear classification problems using what's been called the *kernel trick*, in which non-linearly separable features are mapped into a higher feature space, making the modified feature space linearly separable.

The C4.5 algorithm, on the other hand, selects for given training data the feature with the lowest entropy — or, equivalently, the feature with the largest information gain. It then partitions the data on this feature, and recursively applies the algorithm to every subset. The result is a tree where every node is a decision based on the value of a specific feature, and the leaves are output classes. Unlike its predecessor ID.3, C4.5 can handle continuous data (finding an appropriate threshold and splitting data according to it) and can deal with missing values.

### 3.2.6 Corrections

Our simple system assumes a clueless user that does whatever they are told. And because our user is clueless, it is to be expected that it will eventually misunderstand our instruction and choose an incorrect canonical reaction. To keep our automated IF on track, we add a virtual button to it reading "in case of misunderstanding press this button". This button will bring the automated IF back to the previous position, and ask it to try again. Knowing that it has made a mistake, the automated IF has to choose a new canonical reaction. It is clear that they should *not* choose the same reaction as before, but which reaction should be chosen next?

As a baseline, our automated IF can choose literally any other reaction at random. However, since most of the strategies we have seen produce a score over all possible reactions, we could choose instead the next best ranked one. But we can do better: Purver (2004) shows that repeating the exact same instruction is not how humans behave - more than 80% of the corrections in this study are presented as rephrases of the original instruction. As a result, our IF receives also a new instruction from the same cluster. This is in line with our earlier point that, after a misunderstanding, we should not say the same thing that lead to it.

This strategy will trivially improve our accuracy, because the system has now one less target to select from. We can show, however, that a system with a good strategy will perform much better than random chance, and that the final system will rarely require more than one correction.

## 3.3 Experiments and Results

With our system implemented, we tested it on data taken from the GIVE Challenge. Since the GIVE Challenge contains data from both IFs and IGs, our experiments evaluate the performance of our automated IF when reacting to instructions given by a human IG.

We selected a subset of data consisting of instructions and reactions recorded over six virtual worlds. The IG's instructions are recorded with their timestamps, while the IF's actions and positions were recorded every 200 ms.

Our subset of data comprises two different corpora. The $C_m$ corpus Gargett et al. (2010) contains instructions given by several IGs, with interactions recorded in either English or German. The data was collected from 15 German-speaking pairs and 21 English-speaking pairs. All 30 German-speaking participants were native speakers of German — 17 were female and 13 male. Of the 42 English-speaking participants (16 female, 26 male), 35 were native English speakers; the remaining participants self-rated their English skills as near-native or very good.

The German corpus obtained in this way consists of 2,763 instructions, spread over 45 games. On average, each game contained 61.4 instructions (standard deviation SD=24.0) and took about 752 seconds (SD=245.8). The English corpus consists of 3,417 instructions over 63 rounds. Games consisted on average of 54.2 instructions (SD=20.4) and took about 553 seconds (SD=178.4).

The $C_s$ corpus Benotti and Denis (2011) was gathered using a single IG; it is divided into 63 games spanning 3417 instructions, and records 6:09hs of interactions collected over the internet. Each game consisted on average of 54.2 instructions (SD = 20.4) and lasted about 553 seconds (SD=178.4). We used this corpus to measure the effect that reduced variability on the IG's instructions would have on our IF.

We evaluated our system with both Vis and Bhv as segmentation strategies, using 5-fold training and evaluation. The plan actions required for segmentation were obtained using the classical planner FF (Hoffmann and Nebel, 2001). Table 3.1 shows average accuracy results for the $C_m$ corpora. Jaccard is the best word-similarity method, outperforming all others and BLEU. It is however outperformed by the ML methods, with SVM outperforming all the rest. This is not surprising given that the input data for the ML algorithms contains more information about the context of the instruction. When comparing results over the $C_m$ corpus with the $C_s$ corpus, we found an average accuracy improvement of 3.6 points, but this improvement was found to be not-significant.

We performed a human evaluation on the English corpus. Two human annotators were given the same information available to the ML algorithms (instruction, position at instruction time, and visibility area), and asked to

select the proper Vis canonical reaction. Their average accuracy with respect to the IF's recorded reaction is 81%, and a Cohen Kappa of 0.75, which is considered very good (Carletta, 1996). The annotators' disagreements were found to be highly correlated (.88) with the same instructions that our best algorithm failed to understand correctly.

Our error analysis found that typical misinterpreted instructions required a more detailed context and/or made references to previous actions. Examples of such instructions are "yes", "click it", "go back", "keep going", "exit the way you entered", and "press the green button again".

Our second set of experiments concerns corrections. More precisely, we measured accuracy as a function of the average number of required corrections for each strategy. We define an episode to be "successful at n tries" if it takes $n$ instructions for the IF to select the proper canonical reaction. Therefore, we define "accuracy at n tries" at the number of successful episodes that required no more than $n$ instructions.

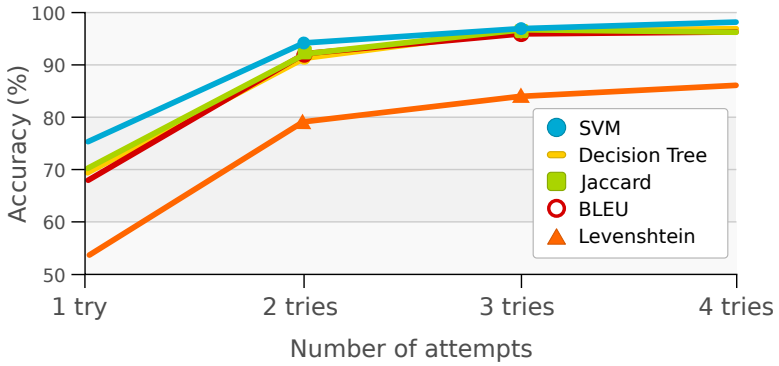



Figure 3.1: Accuracy values as a function of the number of corrections

The accuracy of the Vis segmentation strategy is further explored in Figure 3.1. The poor performance of our baseline strategy shows that accuracy for a bad strategy increases linearly, as expected, while a good correction strategy rarely requires more than one correction. Our best algorithm achieves 92% accuracy at 2 tries.

| **Algorithm** | **English - Bhv** | **English - Vis** | **German - Bhv** | **German - Bhv** |
|---|---|---|---|---|
| Levenshtein | 34% | 52% | 43% | 54% |
| Overlap | 44% | 70% | 51% | 69% |
| Jaccard | 45% | 70% | 53% | 69% |
| BLEU | 44% | 67% | 54% | 66% |
| DT | 48% | 73% | 57% | 70% |
| SVM | 50% | **77%** | 59% | **74%** |

Table 3.1: Accuracy comparison between Bhv and Vis, both in the English and German corpora

## 3.4 Lessons learned

The main weaknesses of our system are its inability to keep track of its context and its greedy approach where only the last received instruction is taken into account.

There are two main lessons to be learned from this experiment: the role of grounding in a collaborative task, and the importance of a Context Set.

Regarding Grounding, our system does not establish a real dialogue with the IG. When our system receives an instruction it makes a shallow word-level analysis, chooses an action, and forgets everything else. We have seen the downsides of this strategy: the highest source of errors in our experiments are those cases where the IG attempts to refer to the common ground between both. Adding memory to our system alleviates this problem, and even a short, one-instruction-long memory improves accuracy. A system with a medium-term memory could reasonably assume that an IF remembers a previous situation and make a shorter, more precise RE that establishes a direct link between old events and the current situation.

As for the role of a Context Set, the nature of our human-collected data provides the system with information about the *affordabilities* of the current context, and as a result our system quickly discards a large set of distractors improving its accuracy. The remaining objects, however, are considered equally likely.

A human IF would not assign probabilities to every affordability in this way: pressing the light switch should not be as likely as activating the fire alarm. Assigning probabilities to every affordability and selecting the most likely ones is a mechanism reminiscent of the *attention* that an IF pays to every object. These probabilities can be influenced by a multitude of factors, some of which we will explore in subsequent chapters.

## 3.5 Conclusion

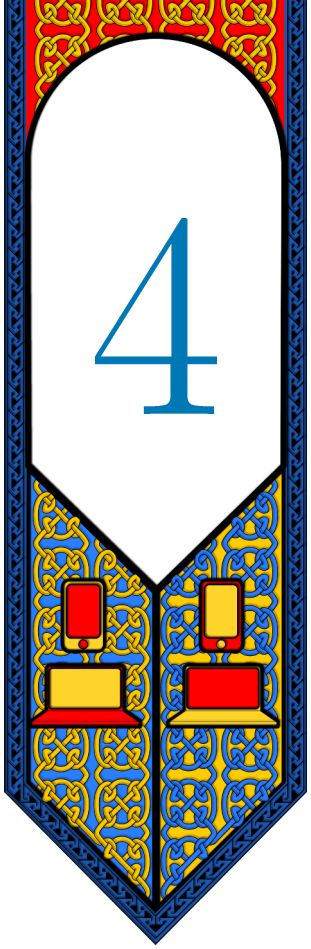
The system presented in this Chapter performs with good accuracy, requiring no manually annotated data. Given that training data can be easily obtained by recording humans interacting with the VE in an unscripted way, the system is well suited for rapid prototyping and simple domains. Results show a simple strategy can achieve good results, with a greedy strategy making the right choice in 80% of all cases. Having now a clear picture of how to deal with typical mistakes, accounting for the remaining 20% is where our next steps will focus.

The main weakness of our system is modeling an Instruction Following task as a sequence of individual, one-shot attempts at identifying a target. A good IG system must explicitly model a link between past events and the current situation, in a process that vaguely resembles the collaborative con-

struction of a common ground. The system's inability to refer to previously established landmarks is a key component in a collaborative task, and an aspect that we will exploit for the correction of misunderstandings. A second weakness of our system: it is missing a model of listener's understanding. Our automated IF uses a simple binary model in which some actions are either uniformly likely or not possible at all, but a human IF would assign individual probabilities to each possible course of action. And those probabilities would change over time.

Finally, our system confirms that merely presenting a correction helps, but it is not enough — good corrections do make a difference. Random corrections may eventually lead us somewhere, but good corrections will drastically reduce our error rate. Rephrasing the original instruction is a good first step that we explore in detail in later Chapters, leading finally to the development of an algorithm for contrastive feedback.

We have now seen the performance that can be expected from a system that does not stop an IF from making mistakes. Rather than letting them happen, our next Chapter presents a robust strategy for detecting them.



# 4

# Detecting misunderstandings

On a typical Instruction Following task, an Instruction Giver (IG) and an Instruction Follower (IF) are paired in order to complete a task: the IG knows which steps have to be completed and in which order, while the IF is the one who performs the steps. To succeed, the IG guides the IF through both navigational instructions and Referring Expressions (RE). A key problem in this scenario are misunderstandings: even if the IG's instructions are technically correct, there are several factors that could lead to the IF performing a different action that the one intended by the IG.

Misunderstandings in Instruction Following tasks are common and even expected. A familiar case from the GPS navigation task is the now-defunct term "recalculating", used by GPS's until 2013 to indicate that the driver had deviated from the established plan and a new route was needed. The term became so familiar as to enter popular culture, highlighting that mistakes happen to *everyone* even in mundane circumstances. Mistakes can happen for a multitude of reasons: studies have shown that our decisions are regularly affected by factors as diverse as first impressions, the order in which alternatives are presented, overconfidence in our intuition, excessive optimism, attempting multiple simultaneous tasks, amounts of sleep, familiarity with the task, and many more (Hallinan, 2009). And they can be costly: no amount of recalculation on the GPS's part can undo a head-on

This Chapter is based on the publication "Predicting the resolution of referring expressions from user behavior" (Engonopoulos et al., 2013).

collision caused by a wrong turn.

Detecting a misunderstanding once it took place is trivial; detecting that a misunderstanding is *about to happen* is a more challenging task, and this Chapter is dedicated entirely to our research on whether is it possible to detect misunderstandings early enough to stop it from taking place. We model this problem with a statistical approach: we train a probabilistic model with recorded interactions from the GIVE Challenge, teaching our model to predict which object in a scene is more likely to be selected given both the text of a specific instruction and the observed behavior of IFs in response to it. We decompose this model into two: a *Semantic model* that makes predictions based on the text of the instruction alone, and an *Observational model* whose predictions are based on the observed behavior of an IF. We then test our models for accuracy (both individually and combined) at several points before the interaction, looking for the best trade-off between time to interaction (how early can we predict that a given object will be selected) and accuracy (how reliable that prediction is).

The insight gained from these models will be a strong foundation for subsequent Chapters: once we know how to detect misunderstandings, we can focus on preventing and correcting them.

## 4.1 Definitions

Before moving forward, we need to define some key terms and concepts that will be used throughout this thesis.

A GIVE world can be understood as a collection of interconnected rooms in the GIVE Virtual Environment (VE). This world is populated with *objects*, some of which are interactive (buttons, alarms) and some of which are merely decorative (lamps, chairs, windows).

In order to successfully navigate the GIVE world, the IG will guide the IF with utterances. Most utterances in a GIVE game can be classified as either *navigational* or *manipulation* instructions. Navigational instructions lead the IF from one object to another, while manipulation instructions ask the IF to interact with a specific object in the current environment.

Manipulation instructions typically refer to a specific object. In this situation, we will call this object the *intended object* of the utterance, while all other objects (interactive or not) are considered *distractors*. These instructions often take the form of a verb followed by an RE in the form of a definite NP, such as "press the green button".

We define an *Episode* as an interaction that begins with a manipulation instruction and ends up with the IF manipulating an object, with no other instructions in between. Each Episode will be represented as a triple $(r, s, \sigma)$, where $r$ is the text of the manipulation instruction, $s$ is the state of the VE when receiving the instruction, and $\sigma = \sigma_1 \dots \sigma_n$ is a sequence

of states or *frames* of the VE in chronological order[1]. The time interval between two successive states $\sigma_i, \sigma_{i+1}$ is 500ms. Figure 4.1 illustrates these concepts.

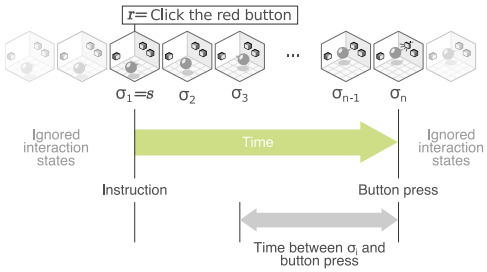



Figure 4.1: Anatomy of an episode

Episodes can be either *successful* or *unsuccessful*. A *successful* Episode ends with the IF interacting with the intended object of the utterance $r$, while an unsuccessful Episode ends with the IF interacting with any other object. An Episode can also be *easy* or *hard*, depending on the number of visible objects in $s$. An Episode is considered easy if there are no more than three visible objects in $s$, and hard otherwise.

## 4.2 A model of listener's understanding

In order to predict how an RE is understood by an IF, we model the grounding process in terms of probabilistic models

When a user receives an RE $r$ in a world state $s$, the user resolves the RE to the object $a$ and moves towards it exhibiting behavior $\sigma$. We assume the probability $P(a|r,s)$ of resolving the RE $r$ to $a$ in world state $s$ to be independent of the probability $P(\sigma|a)$ of exhibiting behavior $\sigma$ when attempting to interact with the object $a$.

Given this assumption of independence, we can now model our probability $P(a|r,s,\sigma)$ that a user who received the RE $r$ in a world state $s$ and exhibits behavior $\sigma$ will interact with the object $a$. According to the definition of conditional probability and our assumption of independence,

$$\begin{aligned} P(a|r,s,\sigma) &= \frac{P(a,\sigma|r,s)}{P(\sigma|r,s)} \\ &= \frac{P(\sigma|a)P(a|r,s)}{P(\sigma|r,s)} \end{aligned} \tag{4.1}$$

Using Bayes' theorem,

$$\begin{aligned} \frac{P(\sigma|a)P(a|r,s)}{P(\sigma|r,s)} &= \frac{P(a|\sigma)P(\sigma)P(a|r,s)}{P(\sigma|r,s)P(a)} \\ &\propto \frac{P(a|\sigma)P(a|r,s)}{P(a)} \end{aligned} \tag{4.2}$$

[1]Under this definition, $s = \sigma_1$

We assume for simplicity a uniform $P(a)$, arriving to our final model

$$P(a|r,s,\sigma) \quad \propto \quad P(a|\sigma)P(a|r,s) \tag{4.3}$$

As a result, we represent $P(a|r,s,\sigma)$ as the product of two probabilistic models, a *semantic model* $P_{Sem}(a|r,s)$ and an *observational model* $P_{Obs}(a|\sigma)$. The semantic model $P_{Sem}$models the understanding of the RE $r$ by the IF, while the probabilistic model $P_{Obs}$models the most likely understood target $a$ based on the observed behavior $\sigma$. The probability $P(a|r,s,\sigma)$ is approximated by the product of both models, in a model we've named $P_{Comb}$.

### 4.2.1 The Principle of Maximum Entropy and log-linear models

In a statistical classification problem we estimate a probability $p(a|b)$ of a "class" $a$ occurring with "context" $b$. For example, we might want to decide whether the word *that* is a determiner in a given sentence, or whether the phoneme /'ðeə/ in a sentence should be understood as the word *there*, *their*, or *they're*.

We often lack enough information to fully specify $p(a|b)$, and yet we need to select one of the infinite number of probability distributions capable of explaining the available evidence.

The *Principle of Maximum Entropy* states that the correct distribution $p(a|b)$ is that which maximizes entropy, or "uncertainty", subject to the available evidence Jaynes (1957); Ratnaparkhi (1997). Out of all possible distributions, this Principle states that we should pick the distribution that explains the available evidence making the least amount of assumptions. Entropy as a concept was originally introduced in thermodynamics, but it was Shannon who introduced the term in the context of information theory. The entropy $H(X)$ of a discrete variable $X$ is defined as

$$H(X) = -\sum_{x \in X} p(x) log\, p(x) \tag{4.4}$$

Entropy reaches its maximum value when $p(x) = c \forall x \in X$, i.e., when we have no reason to prefer any value $x$ over any other and all values are equally likely.

Adding evidence to our model reduces entropy. Given the set $\mathbb{A}$ of all classes and the set $\mathbb{B}$ of all contexts, the available evidence is represented as *feature functions* or *features* $f : \mathbb{A}x\mathbb{B} \rightarrow \mathbb{R}^2$.

Using the Principle of Maximum Entropy, we can combine our feature functions into a log-linear model

[2]Early work on Maximum Entropy Models required the image of a feature function to be the set $\{0,1\}$. Subsequent work extended this definition to continuous features in $\mathbb{R}$ (Yu et al., 2009).

$$\begin{aligned} p(a|b) &= \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_j^{f_j(a,b)} \\ Z(b) &= \sum_a \prod j = 1^k \alpha_j^{f_j(a,b)} \end{aligned} \tag{4.5}$$

where $k$ is the number of features $f_j$, $Z(b)$ is a normalizing factor, and $\alpha_j > 0$ corresponds to a "weight" for feature $f_j$. The weights $\alpha_j$ that best fit the training data can be obtained through maximum likelihood estimation Ratnaparkhi (1998).

The resulting model is an instance of a *log-linear model*, a function whose logarithm equals a linear combination of the parameters of the model. Its general form is

$$exp(c + \sum_i w_i f_i(X)) \tag{4.6}$$

where $c$ and $w_i$ are the model parameters. The $P_{Sem}$and $P_{Obs}$models discussed in the next section are log-linear, and so are the models introduced in Chapter 5.

The $P_{Sem}$and $P_{Obs}$models defined in this Chapter are log-linear, and therefore defined in terms of feature functions. Since both models capture related but not identical information, their feature functions are also similar. The Semantic Model $P_{Sem}$ estimates for every target $a$ in a scene $s$ the (initial) probability $P_{Sem}(a|r,s)$ that the IF will understand a given RE $r$ as referring to $a$. Therefore, the domain of its features is the tuple $(r,s)$. The Observational Model $P_{Obs}$ estimates for every target $a$ the probability $P_{Obs}(a|\sigma)$ that the IF will select $a$ based on observed behavior $\sigma(t) = (\sigma_1, \ldots, \sigma_n)$. The domain of its features is $\sigma(t)$.

The features presented here are the result of a selection process in which we trained a more complex model and removed features that did not show a significant impact on accuracy (Berger et al., 1996). All log-linear models were trained using the L-BGFS algorithm provided by the Mallet software package McCallum (2002).

### 4.2.2 Feature functions for $P_{Sem}$

The first set of features attempt to encode whether $r$ is a good description of $a$. They are called **Semantic features**. These feature functions are defined as follows: *IsColorModifying* evaluates to 1 if the color of object $a$ appears in $r$ as an adjective modifying the head noun in $r$, such as "the red button". *IsRelPosModifying* evaluates to 1 if $a$'s relative position to the IF is mentioned in $r$ as an adjective, as in "the left button".

Let $Adj_h(r)$ be the set of adjectives that modify the head noun of $r$. Then, these two features are defined as:

$$IsColorModifying(a,r,s) = \begin{cases} 1 & color(a) \in Adj_h(r) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

$$IsRelPosModifying(a,r,s) = \begin{cases} 1 & rel\_position(a,s) \in Adj_h(r) \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

Instead of looking for specific adjectives with a specific semantic role, we can look for these adjectives in the entire text of $r$. We theorize that a positive value for these new features could identify sources of confusion: a description such as "the button above the red button" when both buttons are red can lead the user to resolve $r$ to the lower red button. These expanded features are called **Confusion features**.

Let $Adj(r)$ be the set of all adjectives in $r$. Then, these two features are defined as:

$$IsColorModifyingConf(a,r,s) = \begin{cases} 1 & color(a) \in Adj(r) \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

$$IsRelPosModifyingConf(a,r,s) = \begin{cases} 1 & rel\_position(a,s) \in Adj(r) \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

**Salience features** are closely related to the concepts of affordabilities and visual salience: given all objects on the VE only some of them are affordable, and the visually salient remaining targets are more likely to be resolved from $r$ to $a$. *IsInRoom* evaluates to 1 if the IF and $a$ are in the same room, while *IsVisible* evaluates to 1 if $a$ is visible to the IF in $s$.

$$IsInRoom(a,s) = \begin{cases} 1 & \text{IF and } a \text{ are in the same room in } s \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

$$IsVisible(a,s) = \begin{cases} 1 & visible(a) \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

*IsTargetInFront* evaluates to 1 if the *angular distance*, the absolute angle between the camera direction and a straight line from the IF to $a$, is less than $\frac{\pi}{4}$ in $s$.

$$IsTargetInFront(a,s) = \begin{cases} 1 & angular\_dist(a,s) < \frac{\pi}{4} \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

Finally, *VisualSalience* approximates the visual salience algorithm given by Kelleher and van Genabith (2004), a weighted count of the number of pixels in which $a$ is rendered.

In Kelleher's algorithm, an object is first rendered onto a screen. Once rendered, said object is understood as a collection of pixels, and its visual salience is a weighted sum over them. The weight of each pixel is given by the formula $1 - \frac{P}{M+1}$, where $P$ is the distance to the center of the screen and $M$ is the maximum possible distance — namely, the distance from the center to a corner of the screen. A detailed theoretical background on this feature is presented on Chapter 5.1.1.

In our specific architecture the rendered screen is not available to the IG, and it is also not part of the context $s$. Re-rendering the scene is an option, but one that is too slow for our purposes of real-time prediction. As a compromise, *VisualSalience* approximates this value creating a virtual screen, dividing it into virtual pixels, and computing salience over them instead. The algorithm projects the set of coordinates in 3D space, projects them into the virtual screen in 2D space, and calculates salience over this approximation.

Assuming a virtual screen of width 1 and height 1/2, an object's horizontal position is $sin(\text{angle to player})$, and its size can be fixed to 0.1[3]. $M$ can be shown to be equal to $\frac{\sqrt{5}}{2}$.

The object's horizontal size projection by distance and angle is given by the equation

$$size' = 2.arctan(\frac{size}{2dist}) \tag{4.14}$$

giving the object horizontal coordinates in the virtual screen in the range

$$\begin{aligned} left = &\quad max(-1, sin(angle\_to\_player) - \tfrac{size'}{2}) \\ right = &\quad min(1, sin(angle\_to\_player) + \tfrac{size'}{2}) \end{aligned} \tag{4.15}$$

The vertical angle of the GIVE Challenge camera is fixed, and therefore we can simplify the equations by ignoring distortions over this axis. As a result, the object's vertical coordinates are in the range

$$[top, bottom] = [max(-1, \frac{size'}{2}), min(1, \frac{size'}{2})]. \tag{4.16}$$

The procedure for calculating VS is then given by the weighted sum over all points contained within this virtual square:

### 4.2.3 Feature functions for $P_{Obs}$

**Linear distance features** assume that the IF will interact with the closest affordable target $a$.
$IsInRoomSeq$ returns the number of frames $\sigma_i$ in $\sigma$ in which the IF and $a$ are in the same room. With $IsInRoom$ as defined by equation 4.11,

[3] All buttons in the GIVE Challenge have the same size, and only buttons are interactive

```
1: procedure VISUALSALIENCE(button)
2:     salience ← 0
3:     for i ← left; i < right; i ← i + step do
4:         for j ← bottom; i < top; j ← j + step do
5:             point_h ← i + step/2
6:             point_v ← j + step/2
7:             P ← sqrt(point_h^2 + point_v^2)
8:             salience ← salience + 1 − (P/(M+1))
9:         end for
10:    end for
11:    salience ← salience ∗ step^2        ▷ step^2 is the area of a virtual pixel
12: end procedure
```

$$IsInRoomSeq(a,\sigma) = max(1, \#(\sigma_i \in \sigma | IsInRoom(a,\sigma_i)) \tag{4.17}$$

*ButtonDistance* returns the distance between the IF and $a$ at $\sigma_n$ (divided by a normalizing constant):

$$ButtonDistance(a,\sigma) = \begin{cases} \frac{distance_{\sigma_n}(IF,a)}{max\_world\_length} & a \text{ affordable} \\ 1 & \text{otherwise} \end{cases} \tag{4.18}$$

**Angular distance features** look for patterns in the direction the IF is looking and/or how their orientation changes over time.
*TargetInFrontSeq* returns the angular distance towards $a$ at $\sigma_n$, or 1 if $a$ is not affordable. *AngleToTarget* returns the normalize angular distance between the IF and the target $a$ at $\sigma_n$.
With *IsTargetInFront* defined as in Equation 4.13,

$$TargetInFrontSeq(a,\sigma) = IsTargetInFront(a,\sigma_n) \tag{4.19}$$

$$AngleToTarget(a,\sigma) = \frac{angular\_dist(a,\sigma_n)}{\pi} \tag{4.20}$$

There is one **Combined distance feature** that measures both linear and angular distance in a single metric: the feature *DistanceK* returns a weighted sum of linear and angular distance between the player and $a$. This feature is called *overall distance* in Koller et al. (2012).

$$\begin{aligned} DistanceK(a,\sigma) &= \frac{turningDistance + walkingDistance}{c} \\ \text{where } turningDistance &= 0.5 * angular\_dist(a,\sigma_n) \\ walkingDistance &= dist(a,\sigma_n) \\ c &= \text{normalizing constant} \end{aligned} \tag{4.21}$$

*LinearRegAngleTo* applies linear regression to the recorded angular distances to $a$ over each frame of $\sigma$, and returns the slope of this regression. A negative value is correlated with the IF turning towards $a$, and vice versa. If $a$ is not affordable in $\sigma_i$, the angular distance is set to $\pi$.

$$\begin{aligned} LinearRegAngleTo(a,\sigma) &= lin\_reg(seq\_angles).slope \\ \text{where seq\_angles} &= map(\lambda x.ang\_dist(x), \sigma = \sigma_1, \ldots, \sigma_n) \\ ang\_dist(x) &= \begin{cases} angular\_dist(x) & x\,affordable \\ \pi & otherwise \end{cases} \end{aligned} \tag{4.22}$$

**Salience features** track the evolution of an object's visual salience during an Episode. Using the visual salience algorithm defined in page 56, *VisualSalienceSeq* performs linear regression over all frames in the Episode, and returns the slope of the regression. A positive value is correlated to increased attention towards the given object, and vice versa.

$$\begin{aligned} VisualSalienceSeq(a,\sigma) &= lin\_reg(seq\_vs).slope \\ \text{where seq\_vs} &= map(\lambda x.VisualSalience(x), \sigma = \sigma_1, \ldots, \sigma_n) \end{aligned} \tag{4.23}$$

*LastVisualSalience* simply returns the value of the visual salience value of the most recent frame.

$$VisualSalienceSeq(a,\sigma) = VisualSalience(\sigma_n) \tag{4.24}$$

Finally, *VisualSalienceSum* returns the sum of all visual salience values in $\sigma$, but where the most recent value is multiplied rather than summed. We expect the latest value to have the higher impact but we still take into account the entire observed behavior (unlike *LastVisualSalience*, where only the most recent frame counts).

$$VisualSalienceSum(a,\sigma) = VisualSalience(\sigma_n) * \sum_{2}^{n} VisualSalience(\sigma_i) \tag{4.25}$$

The last group is composed of **binary features** which look for specific behavior patterns. *IsCloseSeq* simply returns 1 when the player is less than 1 unit away from the object in the last frame. 1 unit in GIVE is roughly equivalent to 1m. Similarly, *IsVisibleSeq* returns 1 when the object is visible in the last frame.

$$IsCloseSeq(a,\sigma) = \begin{cases} 1 & dist(a,\sigma_n) \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.26}$$

$$IsVisibleSeq(a,\sigma) = \begin{cases} 1 & a \text{ is visible in } \sigma_n \\ 0 & \text{otherwise} \end{cases} \tag{4.27}$$

## 4.3 Experimental setup and evaluation

We tested our model using data collected from the GIVE-2 and the GIVE-2.5 Challenges (Koller et al., 2010; Striegnitz et al., 2011). The raw data includes 1833 games for GIVE-2 and 687 games for GIVE-2.5.

To show that our models can generalize to unseen environments, we used the GIVE-2.5 Episodes for training and GIVE-2 Episodes for testing. The data in each dataset was generated with different NLG systems and users. Feature selection was performed over a third dataset Koller et al. (2012).

We preprocessed the data by retaining only *valid games*, as defined in Koller et al. (2010). In the context of the GIVE-2 and GIVE-2.5 datasets, a game is considered *valid* if ...

... the game result is "success", "canceled", "lost", or "timeout", and

... the client did not crash, and

... the game is not a test game (checked via IP, questionnaire answer, and start time), and

... the player finished the tutorial section of the game

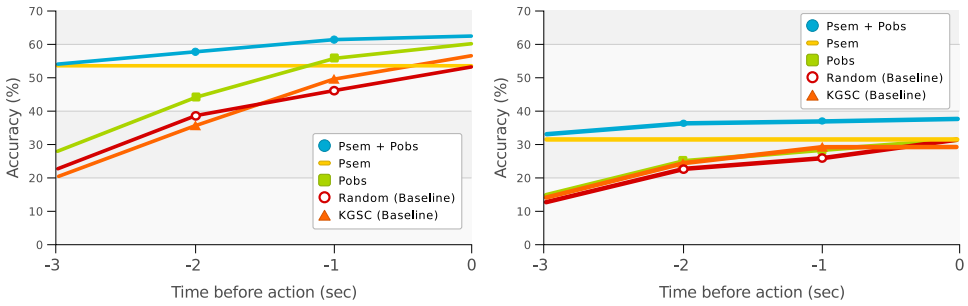
We extracted all Episodes from this data according to the definition in Section 4.1. We then removed all Episodes that took place during the tutorial part of the game, and all Episodes with a duration of less than 200ms (since it's clear that the button press could not be a result of following the instruction of the Episode). Training data for $P_{Obs}$was further refined by only using Episodes of length 2s or bigger, fixing $n = 4$, and ensuring that $\sigma_n$ took place 1s before the button press. Training data for $P_{Sem}$was refined by using only those Episodes where the instruction $s$ contains an RE, removing instructions such as "yes" or "try again". Testing data was not filtered according to either of this constraints. As a result, the final set of training Episodes comprises 6478 training instances for $P_{Obs}$and 3414 instances for $P_{Sem}$. The final testing set comprises 5028 Episodes for both models. We chose GIVE-2 as test set because the mean Episode length is higher (3.3s vs. 2.0s), making the evaluation more challenging.

We evaluated our models on two dimensions. We evaluated our model's *Prediction accuracy* as the ability to predict to which target $a$ has the IF resolved an RE, and our model's *Feedback appropriateness* as the ability to predict that a user misunderstood an RE.

### 4.3.1 Prediction accuracy

Given an Episode containing data $< r, s, \sigma, a >$, we compare the object returned by $\text{argmax}_a p(a|r, s, \sigma)$ to the one manipulated by the IF.

Our experiments included several models and baselines. *Semantic*, *Observational*, and *Combined* are the predictions of the $P_{Sem}$, $P_{Obs}$, and $P_{Comb}$ models respectively. *Random visible* is a baseline that randomly chooses one target among those that are visible in $s$, and *KGSC* is an implementation of the *DistanceK* feature that assumes the IF will choose the button with the minimal *overall distance*. This metric was used in Koller et al. (2012) by the "movement-based system", and was considered a competitive baseline.




(a) All Episodes (b) Hard Episodes

Figure 4.2: Prediction accuracy as a function of time

We evaluated each model and baseline at time 3s, 2s, 1s, and 0s before the button press. Figure 4.2a shows the results for each system, based on evaluation over the 2094 test instances with an episode length of at least 3s. As expected, accuracy increases as the difference between prediction time and button press time decreases. The Combined model $P_{Comb}$significantly outperforms not only the baselines, but also each of its individual components, indicating that each one of its component models contributes complementary information. *Random visible* does not approach 1 at button press time, suggesting that multiple buttons are often visible and confirming that the task is not trivial.

We further tested our models focusing on the 125 unsuccessful Episodes in our dataset, with results shown in Figure 4.2b. These are the hardest Episodes and, as expected, accuracy decreases for all models and baselines. Even then, we observe that $P_{Comb}$still outperforms both its individual components and all baselines.

### 4.3.2 Feedback appropriateness

In our second set of experiments we tested how good our models are at detecting that the IF has misunderstood the RE $r$. We designed an experiment in which we considered a misunderstanding to be detected if $p(a') - p(a) > \theta$ for some object $a' \neq a$, and where $\theta$ is a confidence threshold ($\theta = 0.1$ in our experiments).

We evaluated our models in the 848 test episodes of length 3s or more

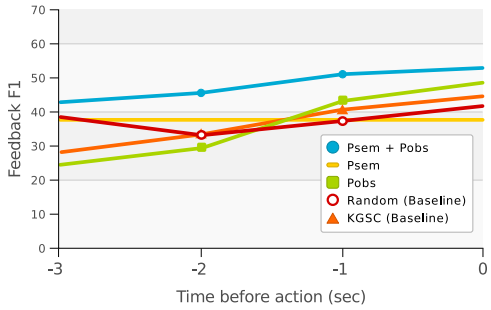



Figure 4.3: Feedback F1 measure as a function of time

where the IG NLG system logged the intended object $a$ for the RE $r$. We measured precision as the number of Episodes in which the IF pressed the wrong button among the instances in which the model indicates that feedback should be given; Recall is measured as the proportion of Episodes in which the model indicated that feedback was necessary over the Episodes where the IF selected an incorrect target. Fig. 4.3 shows the results in term of F1 measure.

Our results show the same tendency we've seen before, with $P_{Comb}$ outperforming all models and baselines. This difference is pronounced at an early stage, meaning that our system would not only accurately detect misunderstandings, but it would do it early enough that stopping the IF from making a mistake becomes feasible.

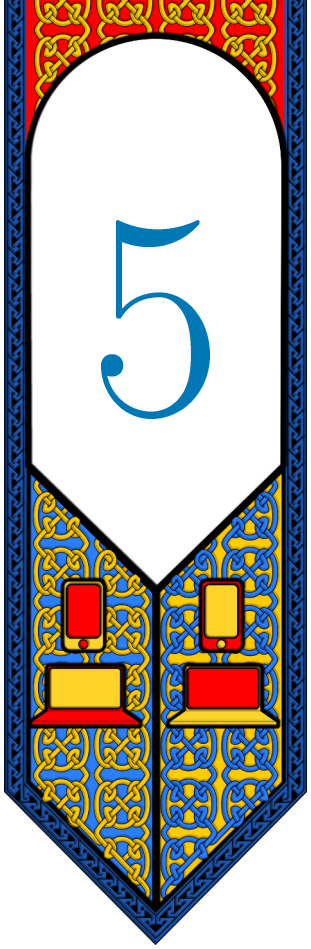
## 4.4 Conclusion

This Chapter introduces two log-linear, probabilistic models of listener's understanding that, when combined, are capable of accurately predicting which target will be selected by a user a couple seconds before the actual interaction. This prediction provides us with a double benefit: if the predicted object does not match our intended object we have detected that a misunderstanding took place, and we also know which specific object is the one that was misunderstood. This second piece of information is critical if we intend to correct the misunderstanding, a topic that we explore in detail in Chapters 7 and 8. This model can also be used as a component for algorithms that explicitly model understanding. The most direct application of this concept is the REG algorithm of Engonopoulos and Koller (2014), which details how to use such a model to generate REs that maximize the probability of being correctly understood.

Of all the features presented in this Chapter, Visual Salience features are most surprising. Our implementation of these features does not analyze the actual pixels on scene as originally intended in the algorithm of Kelleher and van Genabith (2004), but rather return values based on an

*approximation* of what the image on screen might look like. Although we would expect this process to show a high loss of precision for these features, Visual Salience features make it through the feature selection process of both $P_{Sem}$and $P_{Obs}$. These results encourage further research in two possible directions. The most intriguing line of research is whether features based in our abstraction can solve a long standing problem in Instruction Following tasks in the real world: the inability of modern computing devices for capturing the environment in a discrete and comfortable way. An algorithm capable of obtaining a reasonable approximation of the visual salience of an object without the use of cameras would definitely be of interest for researchers dealing with real-world tasks.

A second line of research is to ask whether our models could perform better if they used *real* Visual Salience instead of our approximation. This is a question that will remain open: instead of analyzing the effect of abstracted visual behavior that "proper" Visual Salience would provide, our research in the following Chapter is dedicated to the design of features capable of integrating recorded human gaze into our models.



# 5

# Tracking attention

The previous Chapter introduced our $P_{Comb}$model of listener's understanding as a product of the Semantic model $P_{Sem}$and the Observational model $P_{Obs}$, log-linear models expressed as a combination of *feature functions* with weights learned during training over recorded data. The final set of features for each model is the result of a feature selection process that removed redundant and/or unhelpful features proposed at earlier research stages.

Both models include features based on our approximation to Visual Salience in their final feature set. To understand why this specific feature performs so well we need to rely on both pragmatics and psycholinguistics: the former tells us that non-linguistic aspects are just as relevant to a dialogue as the words being uttered, while the latter explains the relation between our observed physiological behavior and cognitive processes. According to our theory, Visual Salience captures behavior related to gaze, and by extension detects which objects have captured the Instruction Follower's *Visual Attention*. If these features are a proxy to the Instruction Follower's attention, it would explain why these features are so important for each model.

In this scenario a straightforward improvement for our $P_{Obs}$model would include not only Visual Attention, but rather *actual* human gaze. Developing features that leverage gaze information and integrating them into our existing models is a challenge that we tackle as an extension to our current

This Chapter is based on the publication "The Impact of Listener Gaze on Predicting Reference Resolution" (Koleva et al., 2015).

research. Whether gaze information by itself is powerful enough to replace all previous features is a second question that this Chapter attempts to answer.

This Chapter explores the importance of visual features, introducing an extended $P_{Obs}$model called $EP_{Obs}$that uses eye-tracking features. These features are inspired in psycholinguistic research on visual attention, where eye-movement and attention have been shown to be correlated. As a result of including these new features, the extended model outperforms $P_{Obs}$in accuracy for hard Episodes.

We open the Chapter with an exploration of the meaning of "attention" from a psycholinguistic perspective. Section 5.1 answers what does it mean for an object to "capture an IF's attention", and how we progress from "this is what the IF is seeing" to "this is how this RE has been resolved". Section 5.2 introduces the new, gaze-based features of the $EP_{Obs}$model, the experiments we performed, and an analysis of the results.

## 5.1 Attention and Visual Attention

When we talk about attention, we talk about the process according to which we allocate perceptual or cognitive resources to a task or object. These resources are finite, and therefore allocating them to the task comes at the expense of not allocating them to something else (Harris and Jenkin, 2001). The process that we call "to attend" comprises then a dual function: it is both the cognitive process of deciding that available resources should be allocated to inspect an object, and the physical act of allocating those resources.

**Example: Eyes as a resource**

When a person looks at an object, they perform eye movements that align this object of interest with the central fovea of the retina. This specific type of attention is called "overt attention", and is but one of many mechanisms in which our limited resources are allocated (Folk, 2015).

The central fovea has finer spatial resolution than the peripheral retina, but is limited to a range of approximately 2 degrees from the center of the visual field. Therefore, to look at an object implies allocating the limited resource "visual acuity" to a specific object in detriment of all other objects on the scene.

It should be clear that our eyes are a finite resource, that their allocation is strictly dependent on biological characteristics, and that turning our head to the left implies losing sight of objects to the right. To "pay attention" comprises both the decision to align an object with the central fovea, and the physical eye movement that does so.

To understand attention, we need to understand what these limited resources are and how are they allocated. This has been the focus of extensive psycholinguistic literature, and this Chapter will focus on two main aspects: the early vs. late selection debate that cemented the basis for modern attention theory (and, therefore, the basis of this Chapter), and the emergence of computational models, to which our research is closely related.

In 1958, Broadbent (1958) introduced a "filter model" according to which the perceptual system has limited capacity, and it is the job of an attention filter to exclude irrelevant information as soon as a first, rudimentary analysis of the input is complete. This model would be later known as an "early selection" model, based on its thesis that the filtering step took place at an early stage of the perception process. Later developments would cast doubt on this model, proposing instead a "late selection" model, according to which information is perceived fully and only filtered at a later stage (Deutsch and Deutsch, 1963). Both positions were debated for well over forty years. A middle point between these two positions was proposed by *load theory*, according to which early or late selection occurs in relation with the load imposed by the current task: a task that reaches the limits of perceptual processing forces the perceptual system to apply early selection, while tasks with low perceptual load allow for the processing of other, task-irrelevant items as proposed by late selection models (Lavie and Dalton, 2014).

When restricted to visual attention alone, the debate over attention shifts to "visual search", defined as "the collection of processes that allow us to find what we are looking for by using spatial attention to combine the features of objects" (Vecera and Behrmann, 2001). Here, the more general early vs late selection debate focuses on whether object selection ("find what we are looking for") takes place before or after perceptual recognition ("combine the features of objects"). Just as load theory places itself in between the theories of early and late selection, we can reconcile both views with the help of the Theory of Visual Attention (TVA). This theory postulates that both recognition and selection of objects in the visual field occur at the same time.

In TVA, all possible visual categorizations compete to be encoded into visual short-term memory (VSTM) before it fills up. This competition is biased by attentional weights and perceptional biases, making certain objects and categories more likely to be consciously perceived (Bundesen and Habekost, 2014). This model was proposed by Bundesen (1990), and finds itself in line with what would be later called the "biased competition" principle introduced by Desimone and Duncan (1995). While the specific characteristics of VSTM vary from person to person, the capacity of VSTM is assumed to be around 4 objects (Luck and Vogel, 1997). This is one of TVA's main parameters, and can be tuned from person to person.

Based on this model, we can now introduce a working definition of

"attention": we say that an object has *captured an IF's visual attention* if the object has been encoded into the VSTM.

### 5.1.1 Visual Saliency

TVA is a computational approach to visual attention. Unlike a *theoretical* model, a *computational* model is a model that can process *any* visual stimulus, and make predictions "that can be compared to human or animal behavior or physiological responses elicited by the same stimulus" (Itti and Borji, 2014).

Computational models are classified either as *bottom-up* (or *stimulus driven*) or as *top-down* (or *goal-driven*). Bottom-up models assume that a certain visual input will provoke a certain reaction on an individual, while top-down models operate on the premise that an individual will only scan a visual image in search of an object that satisfies their intrinsic motivations. In a top-down model, and given an image, it is possible to assert which object in the scene will capture an individual's attention. A bottom-up model, on the other hand, asserts that a human will fixate on a jar of jam when it is the next object required to make a sandwich (Land and Hayhoe, 2001). The top-down/bottom-up dichotomy is ultimately artificial, as both models depend on each other: a top-down model requires a model of all objects on the environment that can only be obtained in a bottom-up fashion, while bottom-up models ignore an individual's own motivations and desires. Computational models like TVA are more tractable than pure top-down models, and as such have received more attention in recent years.

Both TVA and bottom-up models share their reliance on a *saliency map*, a topographical map presenting a "biased view" of an input image emphasizing interesting locations regardless of *why* those locations are interesting (Koch and Ullman, 1985). Figure 5.1 shows an example of such a map (Itti et al., 1998). The algorithm presented on page 54 is based on such a map, following the research of Kelleher and van Genabith (2004). In this approach, the the saliency of each object is computed based on the number of pixels that an object occupies on screen once it has been rendered, plus the individual distance of each pixel to the center of the screen.

Taking TVA as our model for visual attention, we can now explain the usefulness of our visual salience features used in experiments both in the previous and current chapters: TVA postulates that, in a computational model that can be successfully compared to human behavior, visual attention is driven by the saliency of currently visible objects. This saliency is captured by saliency maps, such as the one provided by our visual salience feature. This map was designed around the observed behavior of individuals, and is roughly equivalent to a TVA model with tuned weights and biases. Knowing which object captured the IF's attention acts here as a proxy to the IF's understanding of a RE — if the IF resolved an RE to a

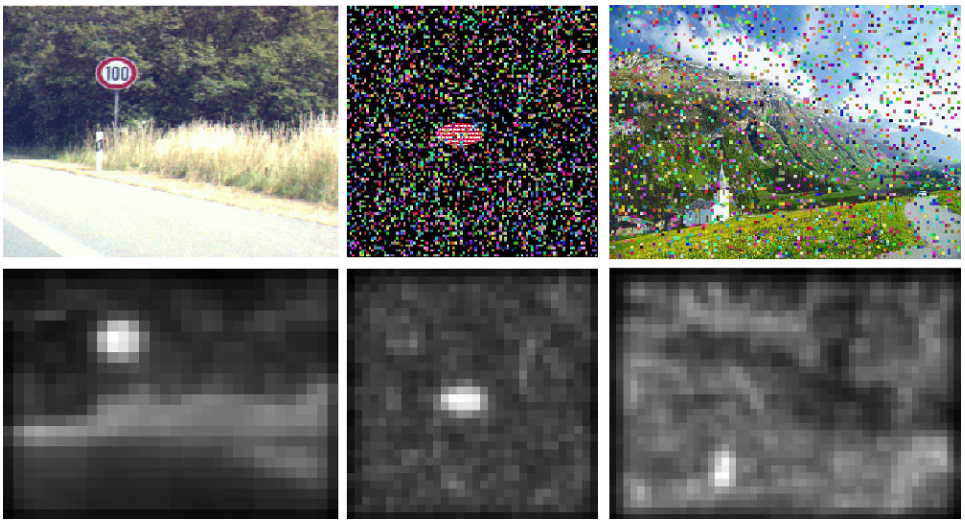



Figure 5.1: Input images and their corresponding saliency maps

certain object, it is expected that they'll focus their attention on it. Their subsequent actions (centering the object in their visual field and walking towards it) will increase the saliency of the resolved object, an increase that will be captured in the saliency map and exploited by our models.

The $P_{Sem}$and $P_{Obs}$model paint a final, more complete picture of why our $P_{Comb}$ model is a reasonable proxy to the user's attention: if $P_{Sem}$ provides a top-down model of a user's interests and $P_{Obs}$provides a collection of bottom-up features, the end result $P_{Comb}$models the same principles presented by TVA. Our model takes into account both the IF's own goals, and the effect that different scene stimuli will have on the IF's attention including (but not restricted to) a model of the IF's visual attention.

### 5.1.2 Eye-tracking

Salience maps are important for computational models, but they are only an approximation to the actual eye-movement observed in human subjects.

Researchers suspected a relationship between attention and gaze-control since at least the Late 1800s. Posner (1980) introduced the "spotlight metaphor" according to which our limited cognitive resources focus on a small area or object at the time, with eye movements following the mental spotlight. Under this model, *saccades* ("quick, simultaneous movement of both eyes between two or more phases of fixation in the same direction" (Cassin et al., 1984)) and attention are linked: it is possible to direct attention to a spot without making an eye movement, but it is impossible to make a saccade without shifting attention first to the target location (Theeuwes, 2014). Perceptual performance has also been shown to increase when perception and

Figure 5.2: Recording user data with an eye-tracker. The image shows the eye-tracker equipment on the bottom right, the player's point of view, and circles (not shown to the player) indicating the recorded user's gaze on the current scene.

saccade are directed to the same object (Bichot, 2001).

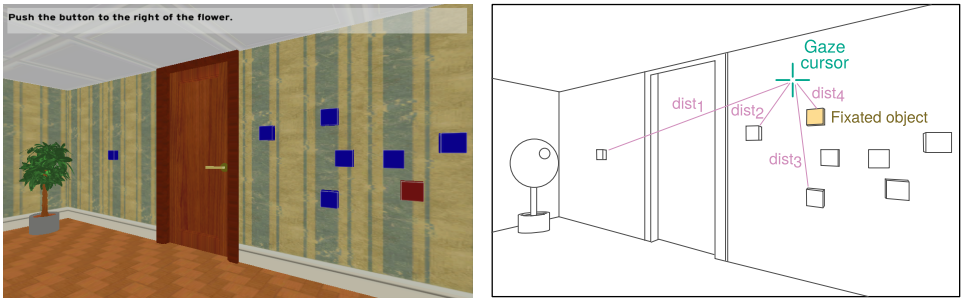
Monitoring gaze can then be used as a proxy to an individual's attention. An optical eye-tracker is a device capable of measuring an individual's eye position, pupil diameter, saccade events, and gaze direction, among others. The faceLab 5 device used by Staudte et al. (2012) reports an average margin of gaze direction error of less than 1°. Data from such a device can improve (and even replace) saliency-map features, as this Chapter shows. Figure 5.2 shows their data collection setup in action.

## 5.2 Eye-tracking and Extended Probabilistic model

Knowing that our $P_{Obs}$model is an accurate reflection of bottom-up attention features, we can show that replacing our approximation to a visual map with a more accurate, eye-tracking-based measure can only improve on our previous results. This section details how we created new sequential features that make use of eye-tracking information as an additional source of data. The resulting model is called *Extended* $P_{Obs}$ ($EP_{Obs}$).

The eye-tracking information is encoded as a single, bidimensional point on the screen where the IF's gaze was fixated at any moment during an Episode. We call this on-screen location the *gaze cursor*.This cursor by itself is not sufficient to determine exactly what an IF was looking at: even accounting for the slight noise introduced by the eye-tracker, human eyes rarely remain fixated on a single point – even if a single object captures an individual's undivided attention, the gaze would constantly move around the object to better discern its features. For this reason, our experiments use the term *fixated object* to refer to the object that is closest to the gaze cursor at

any specific moment in time, as illustrated by Figure 5.3. Should the closest object be more than 50px away from the gaze cursor, we say that no object was fixated at this point in time.




(a) Scene from the GIVE Challenge (b) Gaze cursor and nearby objects

Figure 5.3: Gaze cursor and fixated targets in the GIVE Challenge

### 5.2.1 New features

Being an extension of the Observational Model $P_{Obs}$, the $EP_{Obs}$model also estimates for every target $a$ the probability $EP_{Obs}(a|\sigma)$ that the IF will select $a$ based on observed behavior $\sigma(t) = (\sigma_1, \ldots, \sigma_n)$, where $\sigma_i$ is defined as the same behavior captured in the previous Chapter plus additional eye-tracking data. For simplicity, we use the same notation from the previous Chapter. Under this convention, the domain of all features is again $\sigma(t)$, with the caveat that $\sigma(t)$ now contains additional eye-tracking data.

*LookedAt* counts the number of frames in which the IF has fixated on $a$. *LongestSeq* returns the longest continuous sequence of frames in which the IF fixated on $a$.

$$\begin{aligned} LookedAt(a,\sigma) &= \textstyle\sum_i Fixated(\sigma_i) \\ \text{where } Fixated(f) &= \begin{cases} 1 & \text{if the IF fixated on } a \text{ in frame } f \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.1}$$

$$LongestSeq(a,\sigma) = \max\{|s| \subseteq \sigma : Fixated(\sigma_i) \forall \sigma_i \in s\} \tag{5.2}$$

We define the *linear distance* to $a$ as the euclidean distance in pixels between the gaze cursor and the center of $a$. The *LinearDist* feature returns an average of the linear distance to $a$ over all frames in $\sigma$. InvSquaredDist also returns an average, but using the inverse squared linear distance to $a$ instead.

$$\begin{aligned} LinearDist(a,\sigma) &= 1/n \textstyle\sum_i^n linear\_dist(a,\sigma_i) \\ \text{where } linear\_dist(a,s) &= \sqrt{(a_x - s_x)^2 + (a_y - s_y)^2} \end{aligned} \tag{5.3}$$

$$InvSquaredDist(a,\sigma) = \frac{1}{1 + LinearDist(a,\sigma)^2} \tag{5.4}$$

Finally, *UpdatedFixedObjects* counts the number of frames in which the target was either fixated or less than 50px away from the gaze cursor. This feature can be understood as the proportion of time in an Episode when the IF was interested in a specific target.

$$\begin{aligned} UpdatedFixedObjects(a,\sigma) &= \textstyle\sum_i SemiFixated(\sigma_i) \\ \text{where } SemiFixated(f) &= \begin{cases} 1 & \text{if the IF fixated on} \\ & \text{or 50px around } a \text{ in frame } f \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.5}$$

### 5.2.2 Time to interaction

When evaluating the $P_{Comb}$model in the previous chapter, we trained our model with features evaluated 1 second before the interaction took place. We then evaluated the prediction accuracy of the model with varying time intervals between the prediction of a target manipulation and the actual manipulation.

For this Chapter we decided to perform a more detailed analysis, evaluating the effect that both parameters have in the trained model. Altering the time before interaction we can paint a more detailed picture of the behavior of our trained model. We denote the interval chosen for training as $d_{train}$and the different testing intervals as $d_{test}$. We can then say that our $P_{Comb}$model was trained at $d_{train} = -1s$ and evaluated at $-3s \leq dtest \leq -1s$.

## 5.3 Experimental setup and evaluation

In order to evaluate $EP_{Obs}$, we compared its accuracy to that of $P_{Obs}$in a series of tests designed to replicate the experiment presented in the previous chapter. For our baselines, we included two single-feature log-linear models.

*InRoom* is a model whose only feature function *InRoom* is the defined as in Equation 4.17. This model gives equal probability to all targets in the same room as the IF, and therefore it was chosen as an appropriate random measure for our task. *VisualSalience* implements the sequential Visual Salience feature of Equation 4.23, and was chosen as a baseline to measure the degree of improvement that can be obtained when replacing an attention *estimator* with an attention *measure*. Finally, we also implemented a fifth model *GazeOnly* composed exclusively of feature functions introduced

in this chapter. This model was designed to test whether gaze features alone can replace all other previously studied features.

We trained all models with $-6s \leq d_{train} \leq -2s$ and evaluated them at times $d_{train} \leq dtest \leq 0s$ before manipulation.

The evaluation took place over a subset of GIVE 2.5 data containing recorded eye-tracking data. This dataset was collected by Staudte et al. (2012), involving worlds created by Gargett et al. (2010) designed to evaluate the performance of IG systems in situations of varying levels of complexity.

This corpus provides recorded eye-tracking data, collected with a faceLab eye tracking system remotely monitoring participants' eye movements on a 24" screen. Participants were told that their gaze would be recorded, and a calibration session took place before the actual game. Each participant played three complete games, each one in a different world and with a different IG system.

Following the definition of valid games presented in Chapter 4.3 and extending it to eye-tracking data, we only kept games where the client did not crash, was not marked by the developers as a test game, and the player completed the tutorial. We then used the eye-tracker calibration data to remove games where the eye-tracker data was not properly calibrated. Finally, following Staudte et al. (2012), we only kept interactions for which the eye-tracker calibration detected inspection of either the target or another button object in at least 75% of all referential scenes in an interaction.

The final corpus includes 75 games, with a combined length of 8 hours. We extracted 761 episodes, for a combined total of 47m 58s of recorded interactions and an average length of 3.78 seconds ($\sigma$ = 3.03s) per Episode. 261 Episodes are shorter than 2s, 207 are located in the 2–4s range, 139 are located in the 4–6s range, and 154 Episodes are longer than 6s.

This dataset is not only smaller than the dataset from the previous chapter, but also recorded for a single challenge. Therefore, we replaced the cross-corpora-challenge setup from the previous experiment with a 10-fold cross-validation setup. To keep the folds balanced, we first removed all Episodes of length less than $max(d_{train}, d_{test})$. The remaining Episodes were classified as *easy* or *hard*, depending on the number of visible objects at test time. An Episode was classified as *easy* if no more than three objects were visible at that point in time. For $d_{test}$=0, 59.5% of all Episodes were considered hard, a value that increased to 72.7% for $d_{test}$=-6.

### 5.3.1 Results

For each pair of the 25 possible $(d_{train}, d_{test})$ pair of values, we extracted all Episodes fulfilling the length criteria. After segmenting them into training and testing folds, we trained all models over the resulting training set.

We compared the accuracy values for $P_{Obs}$and $EP_{Obs}$over all combinations of the $(d_{train}, d_{test})$ parameters using a paired samples t-test. The test indicated that the accuracy of the $EP_{Obs}$model $(M = 83.72, SD = 3.56)$ is significantly higher than the accuracy of the $P_{Obs}$model $(M = 79.33, SD = 3.89)$, $(t(24) = 9.51, p < .001, Cohen's\, d = 1.17)$. These results indicate that our eye-tracking features are particularly helpful when it comes to predict which object will be selected by the IF in a hard scene. Figure 5.4 shows results for all models except *GazeOnly*, segmented into easy and hard instances. *GazeOnly* did not outperform any of the other models, and was removed from the graph for clarity.

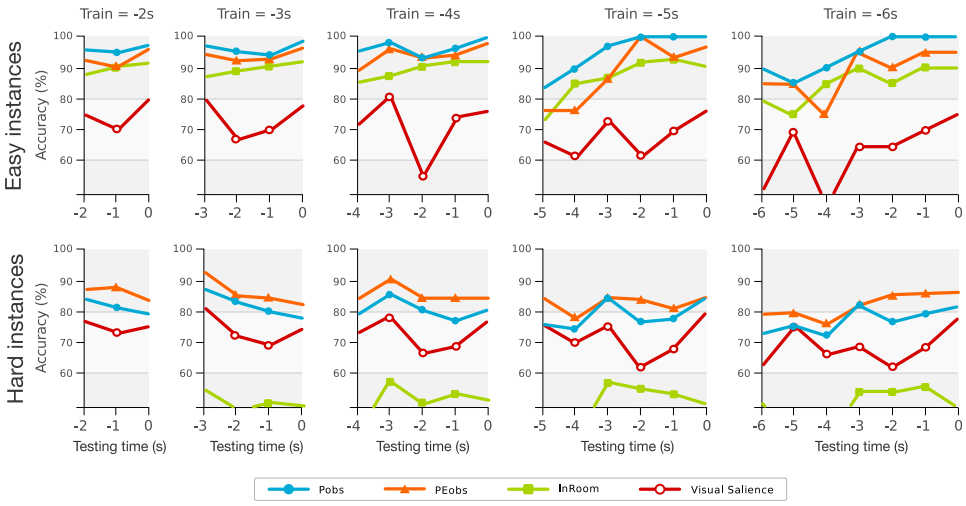



Figure 5.4: Accuracy as a function of training and testing time

Our results also show a peak in accuracy around the -3 seconds mark. We created a 2x2 contingency table to contrast correct and incorrect predictions for $P_{Obs}$and $EP_{Obs}$, and filled it with data from all Episode judgments for $-6s \leq d_{train} \leq -3s$ and $d_{test} = -3s$. McNemar's test showed that the marginal row and column frequencies are significantly different $(p < 0.05)$. Seeing how close this peak is related to the average Episode length (and, by extension, to the average required time between an utterance and the resulting manipulation), this result show that our model is more accurate precisely at the point in time when we expect the fixation to an object to take place.

## 5.4 Conclusion

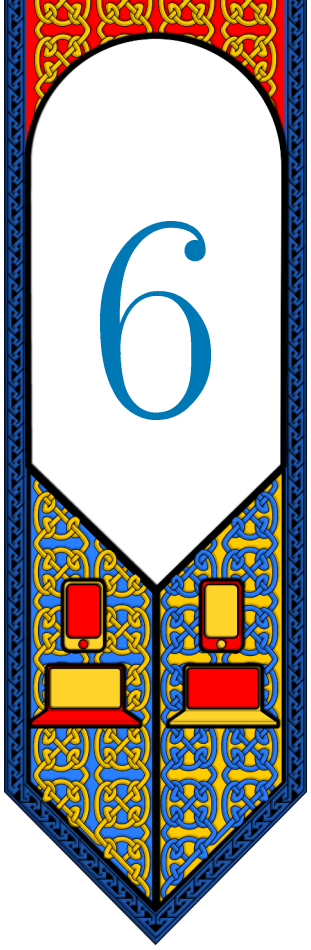
In this chapter, we have improved the accuracy of our model of listener's understanding developing feature functions that analyze the IF's gaze. As we have seen in our literature review in Section 5.1, gaze fixation, dura-

tion, and movement are closely related to an IF's attention. This suggests a plausible explanation for why the Visual Salience features of Chapter 4 are so relevant for the model's success: because they work as a proxy for our model to understand what the IF is paying attention to in a more direct way than, for instance, the speed at which they walk towards a target.

In a GIVE game we would expect the user's visual attention to provide valuable information about the IF's current goal, and our results are consistent with our expected improvements: for the prediction of RE resolutions, our $EP_{Obs}$model outperforms our previous $P_{Obs}$model in hard referential scenes, with significant accuracy improvements in early stages of an interaction. Accurate predictions at early stages give us time to correct a misunderstanding before the IF makes a mistake, saving both the IF and IG the trouble of repairing from a potentially costly mistake.

Even though gaze features improve the accuracy of our model, we have also established that these features are not powerful enough by themselves to outperform even the simplest baseline. To successfully use these features, a model must be combine them with other, simpler features that capture other pragmatic phenomena taking place during an Episode.

With these results, we are now confident in our design choices regarding the use of log-linear models for the prediction and detection of misunderstandings. We now explore practical considerations regarding crowdsourcing in Chapter 6, followed by our first approach to the correction of misunderstandings in Chapter 7.



# 6

# Crowdsourcing and cheating detection

The experiments in previous Chapters were performed over recorded data. Live data was not required because detecting misunderstandings does not alter the behavior of the person following instructions, and the role of the misunderstanding detection system is entirely passive. Correcting a misunderstanding, on the other hand, requires an active collaboration between the Instruction Given system and an Instruction Follower.

Performing laboratory experiments is the most common approach for obtaining results on interactive tasks like these, but planning laboratory experiments is not without difficulties: recruiting participants is often an expensive process, both in time and money. It is also not trivial to show how experimental results generalize when the test population is composed almost entirely of highly-educated students from a single University.

Although there is no *silver bullet* that can dispel all of these concerns, *crowdsourcing* promises comparable results to those obtained in the laboratory at a lower cost. Crowdsourcing is a practice in which a number of participants or *workers* from all over the world are recruited over the internet to complete typically short tasks or *HITs* that are easy for humans to solve but difficult for computers. It has become a popular mechanism for both academics and industry to collect experimental data, as it can be done quickly and cheaply.

There is no argument that the monetary cost of crowdsourcing is lower: crowdsourced workers are typically paid less than students, and performing multiple experiments in parallel drastically reduces the time required

to obtain results. But there is a hidden cost: development time. In addition to the overload that planning an online experiment requires (including knowledge of computer networks for experiments requiring more than a simple questionnaire), crowdsourced workers have a motivation to *cheat* that is typically absent in laboratory participants. Despite the existence of guidelines and best practices, there is no single approach that works for every single experiment. Our experiments are particularly ill-suited for standard cheating detection techniques: there is no "gold standard of human behavior" that we can use to detect that a player is exploring the Virtual Environment in a "dishonest" manner, and the excessive length (by crowdsourcing standards) of a typical GIVE experiment (about 10 min) makes our task a particularly attractive target for cheating behavior.

This Chapter explores the countermeasures we took for our experiments and the rationale behind them, presenting a view inspired both by best practices in security research and by psychology research on the motivations of crowdsourced workers. We also explore whether the cost of these extra steps still make crowdsourcing a viable alternative for typical laboratory experiments.

## 6.1 Cheating

This chapter uses the term "cheating" for two distinct but related situations:

**Malicious cheating** A player attempts to get paid for completing a HIT without following our instructions, done in an intentional manner.

**Unintentional cheating** A player behaves in such a way that no rule is technically violated, but where the resulting data (if any) is not fit for analysis.

**Example: What constitutes malicious behavior**

Players were told that they could only play the game once. A player that disguises his IP address in order to play more than once (and get paid for each one) is a *malicious cheater*, while a player that restarts the game because his web browser hung up (believing, perhaps, that the first attempt doesn't "count") is an *unintentional cheater*.

Even though the theoretical/moral implications for each type are different, the practical concerns are ultimately the same: whether a worker should be paid, and whether the quality of our data is high enough to be analyzed. Therefore, we treat both cases mostly as one (for a more detailed analysis, Gadiraju et al. (2015) present a more detailed taxonomy of untrustworthy workers).

We make a strong distinction between cheating and bias: in our experiments for Chapter 8 we observed that our collected data was skewed. This phenomenon is known as "presentation order bias", and it is to be expected from honest workers in certain situations (Buchholz and Latorre, 2011). We had to design our experiment in such a way as to prevent this bias from altering our results, and yet it should be clear that exhibiting this bias is not the same as cheating.

### 6.1.1 The security mindset

Schneier (2008) introduced the idea of the "security mindset". Under this approach, a security professional "can't walk into a store without noticing how they might shoplift" because they approach every situation wondering how they could make it fail. Schneier argues that this mindset (or the lack of it) is the source of many security failures: the designers are so focused on making the system work, that they forget to account for how it could be made to fail.

When it comes to detecting and preventing cheating, this mindset is crucial. In their paper, Berinsky et al. (2012) affirm that "[Crowdsourcing] is, in short, extremely inexpensive relative to nearly every alternative other than uncompensated students". Although our experience agrees with their finding, it is important to remember that there's more to the cost of an experiment that its monetary value. A typical laboratory experiment assumes a cooperative relation, in which test subjects are invested in the experiment and do their best. In contrast, a crowdsourced task should be better approached as a partially adversarial relation, where a single dishonest participant could derail an entire experiment: given that workers use forums to tell each other both about well-paid tasks and easily-cheatable ones, a weakness in the experimental setup can derail the entire experiment (Vaughan, 2018; Buchholz and Latorre, 2011).

Conti and Caroland (2011) assert that "the dissonance between how our adversaries operate and how we teach our students puts our students at a distinct disadvantage when faced with real world adversaries who inevitably do not play by the rules". This scenario comes into play when we deal with anonymous internet users who would derive monetary gain from our failures to properly secure the experiment. It should be clear that a properly designed crowdsourced experiment requires a a security mindset, and appropriate checks and tools to support it.

This is the extra price that experimenters must pay when crowdsourcing a task, requiring a new approach to typical laboratory tasks. The countermeasures detailed in this chapter are inspired by this mindset, presenting a case-study on how we adapted our existing infrastructure for a crowdsourcing setup.

### 6.1.2 Related work

The subject of cheating in crowdsourcing has received increased scrutiny in recent years, centered primarily around two questions: why do workers cheat, and what can be done to ensure that the collected data is "honest".

Studies have long asserted that the reliability of data collected via crowdsourcing can be comparable to that of data collected in a laboratory (Gadiraju et al., 2017). That doesn't mean that cheating does not occur. Suri et al. (2011) asked workers to roll a number of dice and self-report their results, concluding that "while few cheated a lot, many cheated a little". Other studies have reported nonsensical answers (Corrigan-Gibbs et al., 2015; Gadiraju et al., 2015) and the use of automation tools (Eickhoff and de Vries, 2013; Dreyfuss, 2018). For an extreme case of dishonesty, Stefanovitch et al. (2014) performed a detailed analysis of active attackers effectively disrupting a crowdsourced system in a document-reconstruction competition; the "multiple attackers" described in the article were later revealed to be just two disgruntled competitors (Harris, 2015). This incident serves as a cautionary tale of the damage that can be done to a task in the absence of appropriate quality checks.

On the topic of "why do workers cheat", the literature research performed by Corrigan-Gibbs et al. (2015) details such disparate factors as tiredness, being a darkened room, being previously primed with a text about philosophical determinism, time to contemplate their own actions, self-control, and being previously treated unfairly as causes for an increase in cheating rates. Their findings suggest that honor codes are ineffective at curbing cheating and suggest warnings about the negative consequences as a better alternative. They also found that the probability of cheating is negatively correlated with the age of a worker, and positively correlated with how long a worker has been a member of the platform. Suri et al. (2011) found that a change in the average gain that workers could earn by cheating did not significantly impact the levels of dishonesty, and found no significant correlation between cheating and gender, college education, income, nor ethnicity.

Mazar et al. (2008) affirm that, when deciding whether to cheat or not, a person finds a balance between two forces: a cost-benefit analysis on the reward of cheating and the likelihood of being caught, and the effect that cheating will have on their own self-image. The first factor is rooted in standard economic theory (Allingham and Sandmo, 1972), while the second is inspired by psychological studies on intrinsic motivation (Mazar et al., 2008).

Detecting cheating is very task-specific, and therefore no one-fits-all solution exists. IP addresses have repeatedly shown to be an unreliable method for identifying workers, given the evidence of multiple workers using a single address (Corrigan-Gibbs et al., 2015; Berinsky et al., 2012)

to either use multiple parallel accounts and/or conceal their real demographics. Gold-standard questions are the default control method but they are not available for all tasks, and are particularly inadequate for surveys. Gadiraju et al. (2015) has raised further questions regarding their effectiveness, asserting that gold-standards might not be enough to guarantee the reliability of the data since workers classified as *smart deceivers* have become adept at spotting them. Corrigan-Gibbs et al. (2015) detect cheaters via a combination of honeypots, IP address check (when possible), and plagiarism-detection techniques. Suri et al. (2011) performed a statistical analysis on the aggregate data of dice throws, and concluded the presence of cheating due to the statistical unlikeliness of the collected data. Buchholz and Latorre (2011) detect cheating using additional data about the task, checking whether workers that claimed to listen to an audio sample actually did it. Eickhoff and de Vries (2013) introduced agreement with results from other workers and checks for anomalies in task completion times.

## 6.2 Pipeline

In order to understand the quality checks we have implemented, this section presents a brief overview of our data collection tasks and the corresponding pipeline.

The experiments detailed in Chapter 7 required us to re-run the GIVE Environment described in Chapter 2.5, this time using crowdsourced players rather than volunteers.

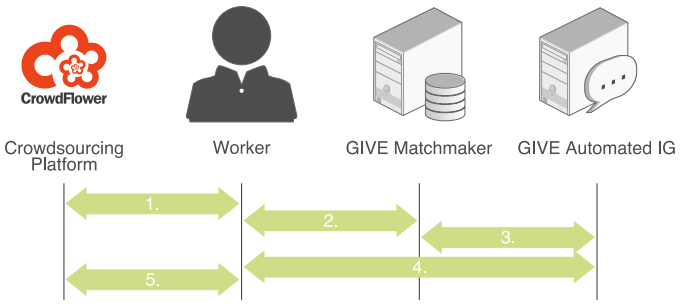



Figure 6.1: Crowdsourcing pipeline

The data collection pipeline for this task can be seen on Figure 6.1. A worker finds our task in Crowdflower (1) and reads the instructions of the task. They click on a link, and this opens a new window to the GIVE Unity client. The client contacts the GIVE Matchmaker (2), which contacts the automated IG system (3). The worker and the IG are then connected, and the Matchmaker monitors the task to log the game in a database (4). Once the game has ended, the worker returns to the Crowdflower interface (5),

and fills a questionnaire about their performance and their general opinions about the game.

### 6.2.1 The GIVE Matchmaker

In order to connect human IFs over the internet with IG systems from several research institutions, the GIVE Challenge provides a service known as the Matchmaker. This server receives connections from the GIVE Client (used by human players), selects an IG system, and connects one with the other. Additionally, the Matchmaker receives periodic update information from both services and stores them in a centralized database. These logs contain enough information to reproduce an entire game, and when collected they conform the raw data that experiments based on GIVE require.

After an assessment of the type of cheating that could be realistically expected, we decided to ignore possible manipulations of the Matchmaker from the participants: while a determined attacker could theoretically alter the reported results, the time required to research and perform such and attack vastly exceeds the cost of both playing the game honestly and other, more popular cheating strategies.

### 6.2.2 The GIVE Unity client

The data collected for the GIVE-2 and GIVE-2.5 Challenges was collected using a web client programmed in Java. This client could be downloaded as a single file requiring no installation, making it easier for players to join the Challenge. By the time we considered the collection of extra data, advances in web technology had made the Java client obsolete. Therefore the client was rewritten by Nikos Engonopoulos using the Unity web framework, running directly in the user's browser and requiring only the install of a standard browser plugin.

As with the Matchmaker, securing the Unity client was not considered a priority for our task given the amount of effort that modifying the binary and/or intercepting communication with the Matchmaker would require. We did ensure however that no task-critical information would be visible in the URL bar of the browser, as this information is plainly visible and easy to modify.

We also accounted for specific behavior in different browsers. Several workers attempted our task using browsers that were not in the list of explicitly supported browsers, leading to unexpected behavior such as loading the wrong GIVE world. Browser incompatibility is one of the main challenges when dealing with online tasks, and for this reason Crowdflower does not allow experimenters to reject workers based on this setting. We solved this problem bundling all parameters inside the binary and segmenting the task into several smaller sub-tasks, each one with their own

version of the binary.

### 6.2.3 The GIVE Automated IG

Given that the GIVE Challenge was designed as a web-oriented task, our IG system was prepared for internet connectivity from the start. Nonetheless, establishing connectivity between all components of the pipeline presents challenges that are typical for all but the simplest of crowdsourced experiments. Solving them is a perfect example of the hidden costs mentioned in this Chapter's introduction, as they can make the difference when deciding whether to use crowdsourcing.

At the simplest level, when a computer system connects with another over a computer network they do so using specific parameters. These parameters include the *IP addresses* of one another (a unique identifier of every computer in the network), the source and destination *port numbers*, and the *protocol* for that communication (TCP and UDP being the most popular). In a typical situation, a *server* will *listen* in a specific port for connections using a specific protocol. A *client* contacts the server sending a message to its IP address and the correct port and protocol. A *handshake* takes place to establish communication, and the interchange itself begins. The entire communication is segmented into *packages* of fixed size containing, among other information, both the sender and destination of said package (Tanenbaum, 1988).

Whether a connection can take place depends on whether both systems can contact each other and whether package exchange between them is allowed. In our experience the most common reasons for communication failure are either network topology problems where at least one of the computers is inside a *private network*, or a strict firewall configuration that blocks connection attempts.

Communication inside a computer network is mediated through *routers*. A router receives all packages from computers inside its network and redirects them to its intended destination. If the destination system is connected to the router, then the router delivers the package directly. Otherwise, the package is forwarded to a different router "closer" to the destination system, and the process repeats itself.

Even though all computers in a private network have individual IP addresses, these addresses are often invalid outside this network. To contact a computer outside the network, the router will often mask itself as the sender, forward all packages to another router, and will afterwards forward responses back to the original sender. This is possible because the router, unlike the sender, is placed in the boundary of two or more networks and can therefore contact other routers. Figure 6.2 illustrates this situation.

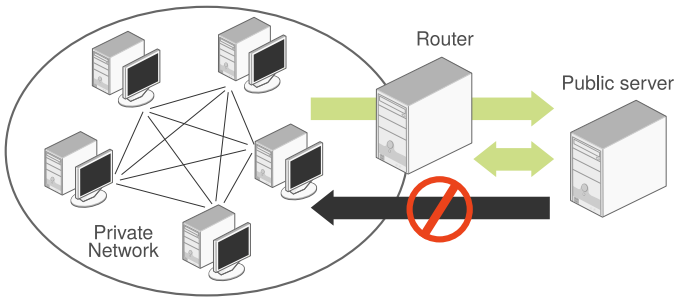



Figure 6.2: Network connectivity example. Computers inside a private network can connect with each other, but communication outside must go through a router. Public computers cannot contact computers inside the private network directly.

**Example: Visiting a website**

A typical private network will assign each computer an IP address in the range 192.168.X.X, where the last two 'X' are numbers in the [0, 255] range. These addresses are reserved, meaning that only computers in private networks can have an IP address in this format. If the computer with (private) IP address 192.168.1.10 wants to contact the website with (public) IP address 200.16.30.5, it sends a TCP package to this address using port 80 (the assigned port for HTTP connections, i.e., web sites).

The router receives this request, and changes the package's sender from 192.168.1.10 to its own public IP address — if it didn't, all packages would be refused because 192.168.X.X addresses are not allowed on the public internet. The website in 200.16.30.5 receives this package and sends a reply back to the router, who finally delivers the reply back to 192.168.1.10.

This process is unidirectional - it is by default impossible to send packages from the public internet to a private IP address because the package cannot be addressed: given that the recipient's IP address is reserved, any package in the public internet with a private recipient address would be immediately rejected. The usual solution to this problem is *port forwarding*, a configuration option in which the router agrees to forward all packages directed to a specific, configurable port to a specific computer.

In addition to whether a package *can* be addressed, routers are often paired with *firewalls* which accept and/or reject connections based on a security policy. In a typical configuration, a firewall may only accept packages directed to a specific port or coming from a trusted network.

Figure 6.3 illustrates the flow of information in our University network from a worker's point of view, as regulated by our institution's firewall and router. We placed the GIVE Matchmaker in a server facing the public


Figure 6.3: Network connectivity for our pipeline. The GIVE Matchmaker is on the public internet, while the GIVE Automated IG resides in a private network. Port forwarding is requires to allow communication between them.

internet, allowing for free flow of information to and from the workers' computers. The Automated IG, on the other hand, was placed inside the University's local network. We chose this setup because using our local computers made it easier to monitor in real time the status of the system.

As expected, placing each computer in a different network led to the communication problems that we explored in previous paragraphs. We solved this problem using *SSH Tunnels* (Ylonen and Lonvick, 2006) as a mechanism for port forwarding. This mechanism opens an encrypted connection from the Automated IG's computer to the GIVE Matchmaker's, keeps it open, and allows traffic in either direction to pass through. We then modified the GIVE Matchmaker to contact the Automated IG exclusively through this tunnel, circumventing the limitations of the network architecture.

Circumventing the firewall added an extra item to our security threat model: the possibility that an attacker would gain access to our internal network through it. Computers connected to the internet are under constant probing for vulnerabilities, and our tunnel could allow an attacker with enough knowledge to reach our internal network. Even though the likelihood of this event was considered low, we ensured that all tunnels were closed immediately after the end of an experiment.

This example illustrates the type of problems that must often be accounted for when dealing with crowdsourced projects. Researchers unfamiliar with computer networks might find these steps too complicated and expensive in time, to the point of making laboratory experiments more cost-effective.

For these reasons, the experiments in Chapter 8 ran entirely on rented, cheap internet servers. While more technically challenging to set up, any potential breach in one of these servers can be easily counteracted by simply deleting and reinstalling the entire server.

| | |
|---|---|
| started_at | When was the task started |
| created_at | When was the task ended |
| worker_id | Crowdflower identifier for this worker |
| country | Country reported by this worker |
| region | State and/or region reported by this worker |
| city | City reported by this worker |
| ip | IP address of this worker |

Table 6.1: Metadata provided by Crowdflower for each worker

### 6.2.4 The Crowdflower interface

The Crowdflower platform is built around the concept of a survey, in which researchers build standard question forms for workers to complete. A typical HIT would ask a worker to describe an image, translate a paragraph, complete a multiple-choice form, or transcribe a short audio recording. Crowdflower also allows researchers to restrict their demographics to workers from certain countries, workers with a minimum personal rating, and/or native speakers of a specific language.

Our GIVE games do not fit this pattern, since asking users to play a short game is not one of the typical crowdsourced tasks, and therefore it is not officially supported by the interface. To work around this issue, we implemented our game as a questionnaire: in the instructions of our HIT we asked users to click on a link, play a game there, and then come back to Crowdflower and answer some questions about their experience. These questions were taken from previous GIVE Challenges and, although we had no immediate use for them, were kept in all tests in case we needed them in the future.

Deviating from the official use cases came with an important downside: we lost the ability to use Crowdflower's quality control measures. In a typical HIT, researchers can intermix questions with known answers, and use them as control - a worker that answers these gold-standard questions incorrectly can be safely removed from the dataset. Our games have no "right" answer, and there is no communication between the platform and the GIVE Unity client. Therefore, there is no mechanism in place for us to indicate Crowdflower that a given answer is "wrong".

After an experiment has ended, Crowdflower provides a file with all the collected data. This file includes information from each worker, their reported answers to our questions, and metadata about both the workers and the task. A list of the most important metadata fields can be seen in Table 6.1.

Finally, the interface presented us with the possibility of contacting anonymous workers using the platform to judge our task. By asking them to suggest fair judgments for difficult situations, inquiring about the clarity of the

instructions, and receiving open feedback in general, we were able to both verify that our games were fair to workers and to improve those aspects where workers suggested improvements.

## 6.3 Quality control

Each part of our pipeline is subject to quality controls. In this way, we can ensure that every unexpected situation is taken care of.

In classification terms, we want to avoid both false positives (marking a worker as a cheater) and false negatives (accepting data from a cheater as valid). False positives make us collect more data than strictly required, incurring in extra expenses and requiring more time to completion; false negatives can poison our data, leading to conclusions that do not correspond with reality. For our experiments, we have chosen to prioritize the detection of false negatives over false positives: collecting more data is still cheap, while identifying and removing inaccurate data would simply move the problem to a sub-task as complex as the current one.

Given that the incentives for crowdsourced experiments are different than those performed in the laboratory, a crowdsourced GIVE game is subjected to extra quality controls than those detailed in Chapters 4 and 5. A crowdsourced game can only be considered valid if it contains at least one button press, a duration of at least 3 minutes (to avoid intentional loses), and it ends in either success, a canceled game, a loss, or a server crash (a result not contemplated in previous definitions). Games that do not fulfill these requirements are flagged as either suspicious or invalid, depending on the discrepancy.

Identifying whether a flagged game should be accepted or not became a large enough problem that further development was required. We established an extra "chain of custody" test in which we would not only identify whether a discrepancy between recorded and reported data was found, but would also identify the source of this discrepancy. For each worker in the Crowdflower-provided dataset, we check a set of conditions:

1. That a game exists in the Matchmaker database with matching IP, experiment ID, and secret words. This is important to ensure that the worker has actually started the client, and was critical for discovering dishonest reports: several players from several countries showed up with the same IP address, and several players would use the same pair of secret words for different games.

2. We then contrast the player-reported result with the database-reported result and, should a disparity arise, investigate its source. Example of discrepancies resolved in favor of the players would be a secret-word


Figure 6.4: Web interface of the cf-thesis project

mismatch for a successful game, games where the connection was interrupted, and games where players were disconnected immediately before ending the task.

3. An extra check was added here to ensure that players that won the game did it at their first try. Despite our instructions, many players would restart the game several times. In those cases, we only used the first game and discarded the rest.

In order to ensure that this "chain of custody" was respected, we developed a custom tool for the cross-reference of data (Lisovskaya, 2015). A screenshot is shown in Figure 6.4.

### 6.3.1 Payment scale

With payment being one of the worker's main motivations, we sought a balance between two opposed interests: we wanted the task to be better paid than other tasks to entice workers to take their time, while remaining cheaper than the typical cost of recruiting volunteers on campus.

We decided to implement a two-level pay scale. We set a base price of US$0.7 per completed game with a bonus of US$0.3 for successfully completing the task, which we estimated to be as high as European minimum wage. With typical crowdsourcing tasks paying below this amount (Ross et al., 2010), we had no shortage of volunteers for our task and each experiment was completed in less than a day. This surplus of volunteers allowed us to be more liberal when it came to discarding suspicious data.

### 6.3.2 Secret words

A downside of having a well-paid task was that the incentive for players to cheat became higher - the fastest they can complete a task, the sooner they

can move to another one. If the task can be exploited reliably, they might also let other workers know.

Our first countermeasure against cheating was adding a pair of random control words to every game. Players were shown two control words, one at the beginning and one at the end of a game, and then were asked to write them when filling the after-game report.

Players who didn't fill the secret words correctly could be removed from the dataset, as this would prove evidence that they did not follow the instructions correctly: either they did not follow the link that starts the game, did not read the instructions in the questionnaire, or both.

This strategy proved successful at identifying users who weren't interested in following the task. Several of the incorrectly entered words revealed players who didn't read the instructions, didn't understand them, and/or directly ignored them. For instance, ...

- ... workers that played the game several times, always repeating the same words from the first play
- ... workers whose computers did not respect our minimum system requirements, and therefore should have not accepted the task: some workers reported error messages for this field, where the text of the error reflected players that did not read/follow the instructions
- ... workers who made up words, since they were not in the list of possible secret words
- ... workers that entered random strings of text
- ... workers playing under different IP addresses but using the same keywords

These results are in line with those reported by Eickhoff and de Vries (2013), who detail that "workers tried to issue made-up confirmation codes, to resubmit previously generated codes multiple times or to submit several empty tasks and claim that they did not get a code after task completion".

### 6.3.3 Results

The pipeline that we presented here is designed with the goal of identifying data that is as good as data collected in a laboratory. The strategies that we presented here provided us with an iron-clad argument for identifying players whose data should be excluded from the final set, but it did *not* provide us with a reason to refuse payment to workers: according to Crowdflower's Terms of Service, all players that complete the questionnaire are entitled to payment for their time, and therefore they had to be paid regardless of whether their data can be used or not.

We also observed that some of our restrictions about workers were not met. We restricted our participants to workers of specific, English-speaking countries and with high English proficiency. Our data, however, provides evidence of workers using proxy servers to access our task from countries other than their own, and of players claiming good English skills while displaying a significant amount of grammatical errors in their responses.

According to our estimates, roughly 50% of the data was discarded due to low quality. After a cost-benefit analysis, we decided that paying all players regardless of the quality of their data was the best solution: the total cost of a game factoring discarded data was still below the cost of paid student volunteers, with the added benefit of fast turnaround between experiments. The dual pay scheme was kept for two reasons: to properly reward players who did their best, and because we observed a higher satisfaction rating for tasks with this incentive over those without.

As for the issue of controlling for English proficiency, we chose to make no statement regarding the English skills of our participants. Given that the original GIVE Challenge made no claims in this regard, simply asking volunteers to self-evaluate, we decided that dropping any claims about this issue was the best solution. Researchers making language-specific claims might decide for a different approach.

The final question is whether we still consider crowdsourced experiments cheaper than a laboratory experiment once all hidden costs are factored in. For standard tasks that follow the templates provided by crowdsourcing platforms, we do believe that the cost is worth it: with the advice presented here regarding a security mindset and the reduced attack surface that crowdsourcing platforms offer, it should be possible for researchers to set up experiments with a reasonable degree of certainty in their results. For non-standard tasks, however, our assessment is mixed: we would only recommend it to researchers who are either already familiar with or interested in computer networks, browser compatibility issues, and security. Otherwise, setting up and debugging a complex crowdsourced experiment might be too much of a time investment when compared to other, more familiar setups.

## 6.4 Conclusion

This chapter introduced a detailed example of an end-to-end crowdsourcing pipeline. While the specific measures detailed here might be too specific for our use case, we hope future researchers will benefit from the detailed analysis on how to secure a specific task using an attacker-oriented approach. Planning our experiments with this mentality enabled us to gather plenty of usable data at lower costs.

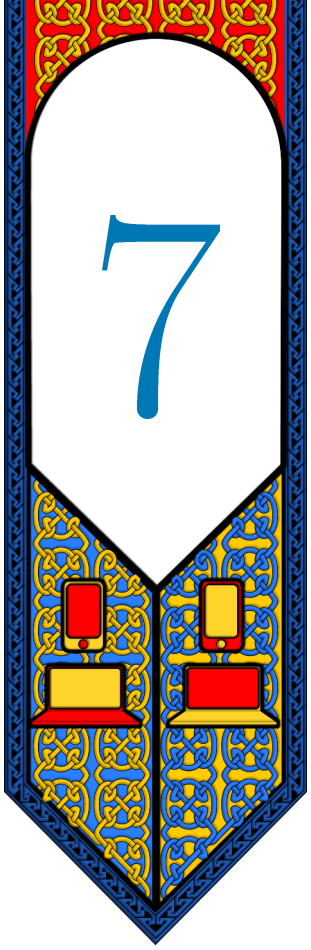
There is no single way to prevent cheating in crowdsourced tasks —

each experiment has its own rules and preconditions, and choosing a cheat detection strategy will always depend on them. But we can still identify some initial guidelines: giving workers motivation to complete the tasks, double-checking all data coming from the workers themselves, and using domain knowledge to filter obviously bad answers can go a long way.

It is also important to remember that the relation between researchers and workers should not be entirely adversarial, but rather a collaboration: a successful crowdsourced experiment should pay the workers well, avoid punishing all workers for the actions of a few "bad apples", and use all possible feedback mechanisms to improve the task using workers' opinions. A researcher should aim for a balance between rewarding good workers and defending against ill-intentioned ones.

We have also explored the technical hidden costs that a crowdsourced task involves. A web-oriented approach like ours requires researchers with a solid knowledge of computer network basics, and may even require institutional support regarding access to public servers and modification of firewall rules. Researchers without this level of technical knowledge are encouraged to either partner with other, more technical researchers, or make use of their crowdsourcing platform's pre-defined templates and tasks to simplify the setup process (at the cost of reduced freedom of choice for experimental setups).

Having established the rationale behind our network architecture, we now move to the experiments that made extensive use of it: the generation of Corrective Referring Expressions with an explicit Context Set.



# 7

# Correcting misunderstandings: reformulation

Once a misunderstanding in a Referring Expression was detected, and the misunderstood referent has been identified, the next step is to find an appropriate strategy to correct this mistake. The simplest possible approach would repeat the last utterance, hoping that the mere repetition of an instruction will be enough to redirect the user's attention from the misunderstood target to the correct one. This approach is sub-optimal for several reasons, as we have seen in Chapter 3: a repeated utterance could no longer apply to a changed context, be less likely to be successful, more complex to understand, or could simply be dismissed as a technical glitch, among others. It is however easy to implement, making it the default first choice for many implementations.

A more advanced approach would take advantage of what we know about the Instruction Follower's attention, namely, which objects have captured it, and which ones have not. The set of objects that captured at least part of the Instruction Follower's attention is called the *Context Set*, and whether we can build a robust error correction strategy around it is an open question that we explore in this Chapter.

In the first half of this Chapter we formalize the notion of a Context Set (CS) both from a Pragmatics and a Computational point of view. Our approach leverages both what we know about Context Sets and the probabilistic information about the environment provided by the models of lis-

tener's understanding that we developed in previous chapters. We build several strategies on top of this formalization, presenting Referring Expression Generation techniques that take advantage of this CS, and tested them experimentally.

Our main hypothesis suggests that Referring Expressions (RE) that are only uniquely identifying with respect to the CS could be more effective that a typical RE, given that they convey the same meaning using shorter utterances. An Instruction Follower that is undecided between a blue button and a red one could probably benefit from an RE such as "the red one" even if there are other red objects in the scene. We therefore present a modification to the SIG-based generation algorithm to model this feature explicitly.

Our experiments on the GIVE Challenge show that this hypothesis by itself does not present the whole story, and we therefore dedicate the second half of this Chapter to an error analysis of our strategy. Researching the type of situations in which our strategy underperforms (along with those where it outperforms the baselines) paints a broader picture of how a Context Set is created and modified, and the complex interaction between attention, instructions, and corrections. The lessons we learn in this Chapter include observations on how to properly account for previously-grounded objects, the importance of proactive feedback, and the effect that a correction can have in the content of a Context Set.

## 7.1 Attention and Context Set

In Chapter 5 we mentioned that the visual short-term memory (VSTM) has a capacity of around 4 objects, and that all objects in the visual field compete to be encoded into it. We also introduced TVA as a computational model that explains how are objects in a visual scene encoded into the VSTM, using weights and biases to model the intrinsic motivations of an individual.

Visual short-term memory is but one kind of memory. When reading a novel, for example, our memory will not be filled with letters and numbers but rather with the characters that populate it. Therefore, we use the more general term "short-term memory" to denote the set of objects that capture our attention, regardless of which of our senses (if any) we used to perceive them.

The concept of attention is intimately related to the concept of *common ground* presented in Section 2.6. In the visual attention literature, our attention is directed both by the saliency of objects in the environment (bottom-up approach) and by our own biases and motivations (top-down approach). The common ground, defined as a shared belief about information that both interlocutors in a dialogue are aware of, also assumes that

specific topics are more readily available than others: to successfully recover the intended referent of expressions such as "her" or "yes, that one", the intended referent must be so salient that both interlocutors are aware of its saliency in the current context, regardless of their own internal biases and weights. This similarity has been explored by Sanford and Garrod (1981), who assert that the candidate referents for an interpretation project are already encoded in short-term working memory.

This set of candidate referents for an interpretation is known in the literature as the *context set* (CS) or *distractor set*, and a formal definition has always proved difficult. Smith and Lieberman (2013) gives a first definition as "the viable candidates for an interpretation process, which evolves over the course of dialogue". Krahmer and Theune (1998) refine it further as "the set of objects speaker and hearer are currently attending to". Gotzner (2017) presents a detailed analysis of current theories, concluding that the set is constructed first with a broad set of alternatives (in line with the findings of Rooth (1992)), but it's then narrowed down by factors such as recency, locality, plausibility, etc. Despite these restrictions, asserts Gotzner, the end result is broader than what it's typically assumed in the literature.

Although the mechanisms that decide whether an object is part of the context set are not clearly defined, researchers have exploited this concept in their work and even presented formalizations of the CS that can be applied to specific domains. DeVault et al. (2004) use a CS to generate REs in an scheduling application, and remark in their results that the effect that specific REs have in the CS is not always easy to predict: in their examples, the expression "Please fill in the cc field of the message" will not only have the intended effect of adding "the cc field" into the CS, but it will also have the unintended side-effect of bumping up the saliency of "the message". Smith and Lieberman (2013) relies on this concept to disambiguate objects when their description uses vague descriptions.

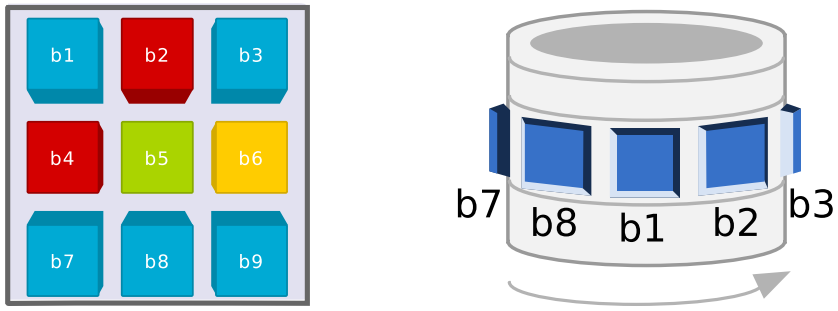
## 7.2 Generating with a Context Set

If we had access to the IFs context set, we could improve our generation algorithm in several ways: if a RE did not encode our intended object into it, we could immediately detect that a misunderstanding took place. We could also optimize our REs, designing them to exploit the most salient information and making them easier to resolve.

If our intended object is the most salient one in the CS, then feedback as simple as "yes, that one" can be effective (Koller et al., 2012). If it's not, we would expect that REs referring to objects in the CS would be resolved quicker, because the objects mentioned in the RE would already be loaded into visual short-term memory (VSTM). But generating REs that only refer

to objects in the CS may not always be possible. Depending on the situation, a CS can either help or hinder our algorithm.

Consider the scene presented in Figure 7.1a, where we want to generate an RE for $b8$ with the CS defined as the set $\{b7, b8, b9\}$. In this situation, we would prefer "the blue button underneath the red button" over "the blue button to the left of the blue button to the left of the blue button": the cognitive effort required to interpret an RE as long as the later would be higher than the effort involved in adding an object with a salient color to the VSTM. On the other hand, it would be impossible on a scene such as 7.1b to generate REs without even an approximation to a CS – the set of visible objects being a common one.




(a) Square grid of buttons

(b) Buttons around a pillar

Figure 7.1: Example of scenes with color buttons

To strike a balance between both situations, we defined an explicit policy for the generation of REs: out of all REs referring to our intended object, our policy considers only those that are uniquely identifying *with respect to the CS*. This policy is based on two hypothesis:

- REs that are not uniquely identifying are typically shorter, and shorter REs are assumed to be easier to understand, following Grice's Cooperative Principle and, specifically, it's maxim "be brief" (Grice, 1975).
- Discriminating between objects, all of which are already loaded into short-term memory, should require little effort. Bringing into play objects that are of no interest for the IF should not help with the resolution of the RE, and therefore should be avoided.

To describe this new property, we added rules to the SIG presented in page 22, replacing the rule shown in Figure 7.2a with the rules presented in Figure 7.2b. This change is required to introduce tighter control of ambiguous REs to the grammar: a not-yet discussed feature of SIGs is the possibility of marking some rules as *final*, and requiring all accepted trees to contain one such rule as their root. The unmodified rule is marked as final

for all $a \in U$:
$\text{NP}_a \rightarrow \mathit{def}_a(\text{N}_a)$
$\mathcal{I}_S(\mathit{def}_a)(w_1) = \text{the} \bullet w_1$
$\mathcal{I}_R(\mathit{def}_a)(R_1) = \mathsf{member}_a(R_1)$

for all $a \in U$:
$\text{NP}_a! \rightarrow \mathit{def}_a(\text{N}_a)$
$\mathcal{I}_S(\mathit{def}_a)(w_1) = \text{the} \bullet w_1$
$\mathcal{I}_R(\mathit{def}_a)(R_1) = \mathsf{member}_a(R_1)$

(a) Original SIG rule for producing uniquely identifying NPs

for all $a \in U$:
$\widehat{\text{NP}}_a! \rightarrow \mathit{hatdef}_a(\text{N}_a)$
$\mathcal{I}_S(\mathit{def}_a)(w_1) = \text{the} \bullet w_1$
$\mathcal{I}_R(\mathit{def}_a)(R_1) = \mathsf{uniq}_a(\mathsf{cs} \cap_1 R_1)$

(b) Modified SIG rules for producing NPs based on the context set

Figure 7.2: SIG rules that were altered to incorporate the CS into the RE generation process

in the grammar, allowing the generation of vague REs such as "the button". In contrast, the two rules in 7.2b introduce a different mechanism: the (non-final) top rule allows for sub-NPs beginning with "the" even if its semantic interpretation resolves to a set of more than one object (as in "the button"), while the bottom final rule introduces productions that are uniquely identifying when their semantic representation is intersected with the CS. We can imagine that the top rule introduces REs of the type "a button" while the bottom rule only generates rules for which "the button" would be unambiguous. While this would be a more syntactically correct approach, the use of "the" instead of "a" in the first case yields more natural REs. With these changes, we can generate REs such as "the button above the blue button" that are uniquely identifying without also requiring the sub-expression "the blue button" to be uniquely identifying.

We have seen in previous chapters how to compute the probability $P(a|r,s,\sigma)$ of a given RE $r$ being resolved to an object $a$ given a context $s$ and observed behavior $\sigma$. This probability is being constantly updated thanks to the $P_{Obs}$model, can be seen as a proxy for the common ground and, by extension, to the elements present in the CS. In our experiments, most objects on a GIVE world display a probability close to 0. Those with non-zero probability were often found to be relevant to the current Episode, and therefore merit its inclusion in the CS. As a result, our general definition of a CS for an Episode that starts with the RE $r$ for an intended object $a$ would be "the set of targets whose probability of being understood as the intended referent of $r$ is higher than 1 standard deviation over the mean".

Given such a CS, we can define a *distractor* $a' \neq a$ as any object whose probability of being understood as the intended referent of the RE $r$ is equal or higher than the probability of $a$.

$$\mathit{Distractors}_a = (a' \in CS | P(a'|r,s,\sigma) \geq P(a|r,s,\sigma) \wedge a' \neq a) \tag{7.1}$$

With this definition, we can now formalize two important concepts: we will say that a *misunderstanding took place* if the set $Distractors_a$ is not empty, and we can define the *misunderstood object* as the object $a' \in Distractors$ with the highest probability $P(a'|r, s, \sigma)$.

## 7.3 Experimental setup

### 7.3.1 Strategies for feedback generation

We implemented three RE generation systems to test our feedback strategies.

The *Baseline* system is a modified P1 system (Garoufi and Koller, 2011b) that does not give feedback unless the player selects an incorrect object. This system guides the player towards the intended object using exclusively simple navigation instructions, not generating any other RE beyond "the door" (as in "go through the door").

Once the intended object is visible and closer than a certain range, a manipulation instruction is given in the form "Click $< RE >$". If the player clicks the correct object then an affirmation is given, and the player is directed towards the next object. If the player clicks the wrong object, or if they leave the room before clicking any object, then the interaction is considered failed and the player receives a new instruction, be it navigational (if the player is now too far away from the intended object) or manipulation (if the player is still close enough to interact with the object). This navigational strategy remains constant for all systems.

The first feedback strategy is named *Repeat*, and it is more proactive towards giving feedback. Whenever the system estimates that the player's attention is not focused on the intended target (that is, if the CS contains more than one element), then it repeats the last instruction if the target is visible, or generates a new navigational instruction otherwise. This system tests our first hypothesis that even a non tailored feedback RE will perform better than no feedback at all as long as the timing is correct.

The second feedback strategy, *CS_Feedback*, follows the steps from the previous system, generating initially the same kind of REs. Should a misunderstanding be detected, however, then a corrective RE is generated, this time taking the CS into account: the new RE is only uniquely identifying with respect to distractors, as detailed in Section 7.2.

In all cases, players can request help at any moment. When this happens, a new instruction is given immediately, disregarding the current strategy and the CS (if it applies).

### 7.3.2 Results

The resulting corpora consists of 370 episodes collected from 69 valid games out of 90 total games from 180 participants. Each player was assigned an IG system at random, and no player was allowed to play more than once. Table 7.1 shows the success rate for each strategy.

| System | Games | Won | Lost | Cancel | Success rate |
|---|---|---|---|---|---|
| Baseline | 24 | 8 | 11 | 5 | 33.33% |
| Repeat | 18 | 6 | 12 | 0 | 33.33% |
| CS_Feedback | 27 | 16 | 9 | 2 | 59.26% |

Table 7.1: Accuracy for each RE Generation system.

Our two main measures of interest are resolution accuracy per button (how many Episodes for each target object were successful) and average Episode length (also per button), subdivided in easy and hard Episodes as detailed in page 71. All systems performed extremely well for easy episodes with accuracy over 97% in the worst case, and therefore are of no further interest for our analysis. Table 7.2 presents the results for hard Episodes, at which point the analysis reveals a less promising picture.

| | **Baseline** | | **Repeat** | | **CS_Feedback** | |
|---|---|---|---|---|---|---|
| | Accuracy | Duration | Accuracy | Duration | Accuracy | Duration |
| **Button** | (%) | (ms) | (%) | (ms) | (%) | (ms) |
| bblue9 | 90.00 | 12 570 | 93.33 | 13 063 | 68.37 | 13 979 |
| bhall1 | 93.65 | 4 887 | 98.61 | 6 610 | 94.50 | 5 355 |
| bhall6 | 84.44 | 11 164 | 90.29 | 8 310 | 73.72 | 8 958 |
| bhall8 | 85.71 | 9 946 | 92.13 | 14 520 | 81.08 | 11 568 |
| bhall9 | 57.14 | 8 343 | 45.45 | 7 715 | 75.00 | 7 320 |

Table 7.2: Average accuracy and Episode duration per button and system (hard episodes)

For hard episodes, the *Repeat* system outperforms on average both the *Baseline* system and the *CS_Feedback* system on accuracy, but neither system achieves statistical significance over the other on a paired t-test. There is a trend in favor of the *Repeat* system over the *CS_Feedback* system, but ultimately no system gets a clear advantage over each other. Removing *bhall9* from the dataset makes *Repeat* (statistically) significantly better than the other two, a result that the next section explores in detail. The average duration of an Episode for each system show no statistical difference between any two systems.


(a) Top view (b) 3D view

Figure 7.3: Button *bhall9* and surrounding environment

### 7.3.3 Error analysis

The results of our experiment are mixed: *CS_Feedback* achieves significantly more completed games, but its REs for hard Episodes are actually *harder* to understand. To complicate matters further, the *Repeat* strategy would show what we expected from previous chapters (that is, an improvement in per-button accuracy due to timely feedback), but this effect disappears once we include results for *bhall9*.

An analysis of these failures yields a better picture of the kind of behavior that is to be expected from human IFs interacting with our system, and provides insight into the strengths and weaknesses of *CS_Feedback*. We focus on three specific research questions:

1. What is special about *bhall9* that makes it critical to the overall success rate of a game?

2. If *CS_Feedback* generates harder instructions, why is the success rate per game higher?

3. Why are *CS_Feedback*'s instructions harder to understand?

To answer the first question, we need to look closer at the environment surrounding *bhall9*, as shown in Figure 7.3.

This button is close to an alarm, and players manipulating the button must pay special attention not to step on it. The REs generated by both *Baseline* and *Repeat* are uniquely identifying w.r.t. the current room and do not change depending on the visible objects, leading to very precise but sometimes complicated REs. In our experiments, complex instructions repeatedly led IFs to take steps back in order to get a better look at a complete scene, or to look for other buttons that they could have possibly missed. It is during this process that many of them step on the alarm and lose,


(a) Top view (b) 3D view

Figure 7.4: Buttons in an L-shaped configuration

while those that don't still spend a substantial amount of time maneuvering around it. *CS_Feedback* does not suffer from this problem because REs change based on the currently visible objects. While these REs may be more difficult to understand, their adaptation to a changing visual environment leads both to higher per-game accuracy (because players don't need to walk back as often, and therefore lose less) and higher accuracy for *bhall9* (because the button is eventually clicked, unlike in lost games).

**Lesson learned:** *bhall9* brings to the surface a specific pattern of human behavior. Explicitly accounting for it goes beyond the scope of either of our systems, but it is encouraging that *CS_Feedback* is not negatively affected by it. ■

A similar behavior pattern explains our second question of why does *CS_Feedback* achieve a higher success rate if its instructions are harder to understand. To explain that, we focus on the scenario shown in Figure 7.4a.

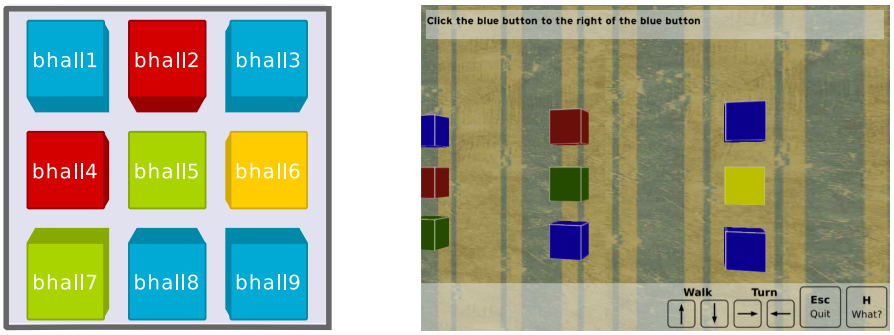
In this Episode, the intended object is *blue9*, while *blue7* triggers an alarm and ends the game. At a distance, all systems generate the RE "the blue button to the left of the blue button", as the green button *green8* breaks the "left-of" relation between *blue7* and *blue9*. However, it is clear when seeing the scene from the IF's perspective (Figure 7.4b) that this is too strict an assumption: many players resolve the RE to *blue7* and lose.

Both *Repeat* and *CS_Feedback* generate feedback seconds before the IF makes a mistake but, critically, *Repeat* presents the exact same RE that led to the mistake. *CS_Feedback* generates here a new RE that, although possibly confusing, is different enough to steer players in the correct direction.

**Lesson learned:** The combination of rephrasing and proactive feedback prevents here a costly misunderstanding, even in the presence of suboptimal REs. ■

With those two questions answered, we can now focus on our main research interest: why are *CS_Feedback*'s instructions so confusing, if their

timing is correct and rephrasing works? We tackle this question from the point of view of a specific Episode shown in Figure 7.5. The Episode evolves as follows:




(a) 2D view (b) 3D view

Figure 7.5: Example of failed Episodes from the IF's point of view

```
IG: Click the blue button to the right of the blue button
IF: <Clicks bhall9>
IG: Good!
IG: Now click the blue button to the left of the blue button
IF: <Clicks bhall1>
IG: No, that's not what I meant.
```

The second RE, "the blue button to the left of the blue button" in the example is intended to refer to *bhall8*, and was considered a good RE for two main reasons: the RE for *bhall8* was uniquely identifying w.r.t the IF's Context Set, and *bhall1* was not considered part of it because it is too far away and out of focus (as seen in Figure 7.5b).

In this Episode, the IF is likely to ground the first RE "the blue button to the right of the blue button" as follows:

$$\underbrace{\text{The blue button to the right of } \overbrace{\text{the blue button}}^{bhall8}}_{bhall9}$$

With *bhall8* grounded as "**the** blue button", the next RE is likely to be resolved in the following way:

$$\underbrace{\text{The blue button to the left of } \overbrace{\text{the blue button}}^{bhall8}}_{bhall1}$$

By referring to "the blue button" in the second RE, we observe that IFs assume that the same RE is used to refer to the same grounded object from the previous interaction. Our IG system, on the other hand, treats every Episode as independent, and does not follow these same steps. We are familiar with this type of error, as we've seen it before in our simple IF system from Chapter 3.

There are several issues at play here. Our first misstep is a definition of a CS that's perhaps too strict. Fraundorf et al. (2013) presents the case that the CS is typically bigger than expected, and that although context does constrain the set of viable alternatives, it only does so loosely. A similar "permissive view" of context sets is supported by Gotzner (2017), who finds in her experiments that "listeners have access to a broader set of alternatives rather than only the small contextually-restricted set". Given our own results, it would be hard to argue against the idea that our context set may have be too strict.

A second issue with our approach is the effect of contrast by itself Even if our CS estimation correlates with that of the IF, our model assumes that the act of presenting a corrective RE is an atomic event, as in:

$$\texttt{Original CS} \rightarrow \texttt{Correction} \rightarrow \texttt{New CS}$$

But there's psycholinguistic evidence that being suddenly made aware of the existence of alternatives (which is what presenting a correction does) might be enough to trigger a re-evaluation of the current scene by the IF and, by extension, to modify the CS even before reading the explicit text of the corrective RE. In that case, the process of interpretation of a correction may be better modeled as:

$$\texttt{Original CS} \rightarrow \texttt{Awareness of alternatives} \rightarrow \texttt{New CS} \rightarrow \texttt{Correction interpretation} \rightarrow \texttt{Updated CS}$$

Gotzner et al. (2016) report in their experiments that "focus particles trigger an active search for alternatives and lead to a competition between mentioned alternatives, unmentioned alternatives, and the focused element". In these experiments, a focus particle (only, also, even, etc.) is enough to trigger a search for alternatives even among elements that were never discussed before. If the presence of a particle by itself is enough to change the contents of the IF's context set, then it is feasible than the presence of a correction by itself can cause the same effect. If that were the case, the IF would interpret a corrective RE in the context of the `New CS` rather than the expected `Original CS`, and this mismatch could lead to the confusing REs that our results report.

**Lesson learned:** The dynamics of user behavior reflected by our CS are good but not perfect. Our *CS_Feedback* strategy does not fully model the dynamics of human RE interpretation, and the assumption that it does led to clearly confusing REs. ■

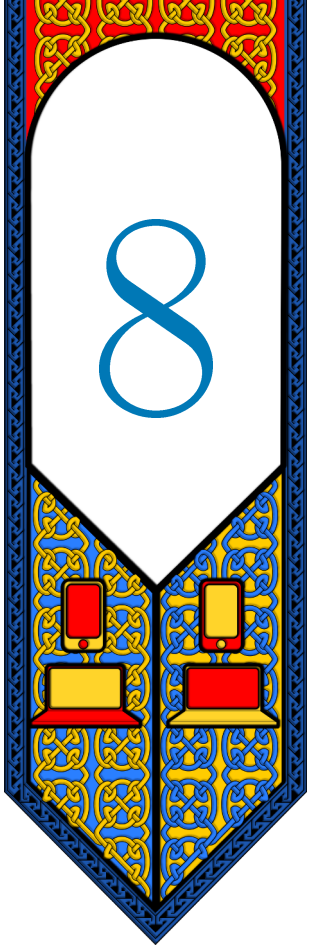
## 7.4 Conclusion

In this Chapter we have explored an unsuccessful attempt to model an explicit Context Set (CS) for our REG strategy. This strategy models a CS under the assumption that an object in this set would be assigned a high probability by the $P_{Comb}$model, an assumption that our results cannot confirm.

Incorporating a model of a CS in our generation strategy in the way we present here does not generate better REs. CS are reportedly difficult to model due because their contents are in constant fluctuation, and our CS model may be too simple to accurately reflect such a dynamic. Furthermore, the act of presenting a corrective RE by itself can change the content of the Set, a dynamic our model does not account for. Even if our CS contained an accurate reflection of the IF's internal short-term memory, REs generated using this information in the way we've presented are too fragile and context-dependent to be useful as a general algorithm. How to use this information as a feature for a different algorithm is a promising topic for future research.

The lessons we learned from this experiment give us a better understanding of the problem, enabling us to design better strategies. We confirm the importance of proactive feedback and rephrasing, preventing mistakes even with sub-optimal REs. We also learned that corrective REs that are not explicitly connected to their original, misunderstood RE can be confusing if the modeling of the environment is less than perfect. We have also confirmed the flexibility and expressive power of our SIG-based generation approach, enabling us to model a CS in a straightforward way.

All of these lessons are put to good use in the next Chapter, where we introduce a difference strategy for corrective REs. This strategy is based on explicit contrastive feedback, presenting REs with explicit contrast between a misunderstood object and the intended one.



# 8

# Correcting misunderstandings: Contrastive Referring Expressions

No matter how "good" an RE is, there is always a chance that a user will misunderstand it. Maybe they misread or misheard the RE; maybe they acted based on a personal belief rather than the actual text of the instruction, or maybe they just weren't paying attention. In either case, a robust REG system should be able to detect and correct these misunderstandings.

The strategy presented in Chapter 7 attempted to generate feedback that is *implicitly* contrastive, exploiting properties of both the misunderstood and intended objects. That approach assumed that the user would make a mental connection between two successive REs, an approach that was ultimately unsuccessful: if a misunderstanding was detected between a red and a blue button, the system failed to account for the situation in which a corrective RE such as "the blue button" could make the Instruction Follower pay attention to a previously unnoticed, also-blue button.

This Chapter generates corrections via *contrastive, corrective feedback* ("contrastive feedback", for short). In our new approach, the Instruction Giver corrects a misunderstanding immediately after detection with a new RE that marks explicit contrast between the intended target and the target the user understood. Such an RE would say, for instance, *"No, not the* **RED**

This Chapter is based on the publication "Generating Contrastive Referring Expressions" (Villalba et al., 2017).

*button, the* **BLUE** *button"*.Unlike the example from the previous paragraph, the system we present in this Chapter would not generate the example correction unless "the blue button" were by itself a good RE for the intended object. The benefits of this approach are multiple: the new contrastive instruction lets the user know that they have made a mistake, guides them away from their mistake and towards the intended target, and makes the link between misunderstanding and correction explicit in order to minimize new misunderstandings.

A key component of our new strategy is the reconstruction of a hypothetical RE that the Instruction Follower may have understood. We have established in previous chapters mechanisms for detecting that an RE for object $o_s$ was misunderstood, and was resolved instead by the Instruction Follower to an object $o_u$. This Chapter identifies also the "misheard" RE $r_u$ defined as the RE from the set of all possible REs for $o_u$ that can be derived from the original RE using the minimum number of edit operations. These edit operations are part of our modeling assumption, where a misunderstanding is seen as the result of a message altered by a noisy channel.

Combining theory on edit automatons, SIGs, and edit distance, we can find both the misunderstood RE $r_u$ and the sequence of edit operations that created it. Applying contrastive focus to all words in the original RE $r_s$ that were either modified or deleted by edit operations, we further generate contrastive REs for any pair of objects on a scene.

We evaluate our approach with crowdsourced experiments over two different domains, the GIVE Challenge and the TUNA people corpus. These experiments present contrast using two possible alternatives: *Emphasis* applies contrastive focus as above, while *Shortening* explores whether it is possible to generate shorter (but still effective) REs by removing information that was already correctly understood. Our results for *Shortening* reflect a similar pattern to those from Chapter 7, showing that making an RE shorter does not always make it better. *Emphasis*, however, significantly outperforms both *Emphasis* and our baselines.

## 8.1 Contrastive focus

When humans make corrections, they often use contrastive focus (Rooth, 1992; Krifka, 2008) to clarify both the intent of the original RE and the relation to the new one. In this context, *contrastive focus* refers to an utterance with either marked pitch accent (in the case of spoken utterances) or a visually distinct style like italics, bold font, uppercase, etc. (in the case of written utterances).

**Example: Contrastive focus**

Imagine that an IG presents the utterance "the big blue button" to an IF, but the IF interacts with a big green button instead. As a correction, the IG may present the utterance "No, the big **BLUE** button", where "**BLUE**" is presented with a marked contrast. This new utterance alerts the IF that a mistake took place, reiterates the information from the original RE, and marks explicitly the attribute that the IF must pay attention to. The contrastive RE makes it evident that the properties "big" and "button" of the intended object where properly understood, but "blue" requires extra effort.

Focus has been extensively studied in the literature. Following Rooth (1992), Krifka (2008) asserts that focus indicates "the presence of alternatives that are relevant for the interpretation of linguistic expressions". Under this theory, focus not only presents a contrast between two or more objects, but also alerts the listener that there *are* alternatives to begin with. Corrective focus, according to Bornkessel and Schlesewsky (2006), can even override syntactic requirements on the basis of "its extraordinarily high communicative saliency". For a more detailed overview on the theory of focus, see Rooth (1997); Krifka (2008).

Face-to-face conversation is "the basic and primary use of language, all others being best described in terms of their manner of deviation from that base" (Fillmore, 1974). We must then differentiate between focus in speech and focus in written form. In speech, focus is typically marked through intonation and pitch accents (Levelt, 1993; Pierrehumbert and Hirschberg, 1990; Steube, 2001), while concepts that can be taken for granted are de-accented and/or deleted. In contrast, focus in written text can be marked in several different ways. Print media and style manuals reserve italics for this purpose (Staff, U.C.P., 2017), but all-uppercase letters are a common alternative when the underlying medium lacks the capacity of displaying text in more than a single style. Bold-faced fonts, underlined text, and visual markers are not unusual. Fraundorf et al. (2013) tested whether font emphasis benefits memory in the same memory that contrastive pitch accent does in spoken discourse, finding positive results for both italics and capitals.

In this Chapter, emphasis will be marked through all-uppercase, bold text. De-accentuation is not typically marked in written text outside of specialized notations, and therefore it's excluded from the rest of this chapter.

The two closest research approaches to contrastive focus and REG are the work of Milosavljevic and Dale (1996) and Krahmer and Theune (2002). The former uses contrastive focus for side-by-side comparison of entities by means of encyclopedic descriptions. Their system pre-empts confusion between two similar entities by placing contrastive focus in properties that distinguish those two entities. Both the potential confusing entity and the

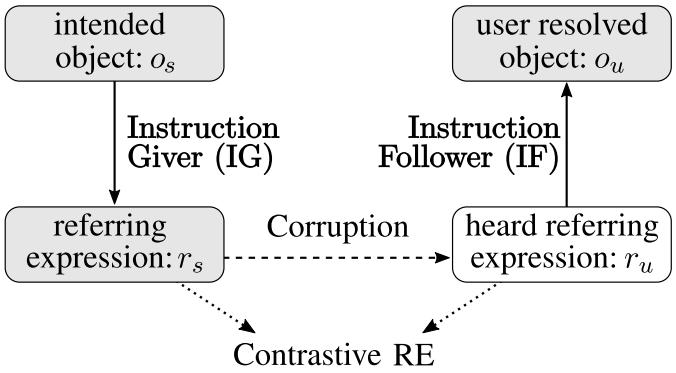



Figure 8.1: Corruption model

distinguishing properties are manually specified. The latter presents an extension of the Incremental Algorithm (Dale and Reiter, 1995) to mark attributes as contrastive. As such, and unlike our statistical approach, this algorithm creates contrastive REs relying on a fixed attribute order. Neither of these works evaluate the quality of the generated REs.

## 8.2 A minimum-distance approach to contrast

We have previously discussed in Chapter 2.6.1 how misunderstandings can take place, and we have introduced algorithms for detecting them. We now motivate our approach towards correcting them with a simplified corruption model.

Let's say that the IG wants the IF to interact with an intended object $o_s$, and presents the IF with an RE $r_s$. The IF, however, resolves the RE $r_s$ to an unintended object $o_u$. We assume that the reason the RE $r_s$ was resolved to $o_u$ instead of $o_s$ is because the RE $r_s$ was sent over a noisy channel, got corrupted, and was received by the IF as a new, different RE $r_u$. This situation is illustrated in Figure 8.1.

This noisy channel is a simplifying assumption with important benefits: the root cause of the misunderstanding is found at the string level of an RE, while remaining flexible enough to permit meaningful corrections for several types of misunderstandings. It should be clear, however, that there are many other reasons why an IF would misunderstand an RE: lack of attention, mismatched common ground, and so on.

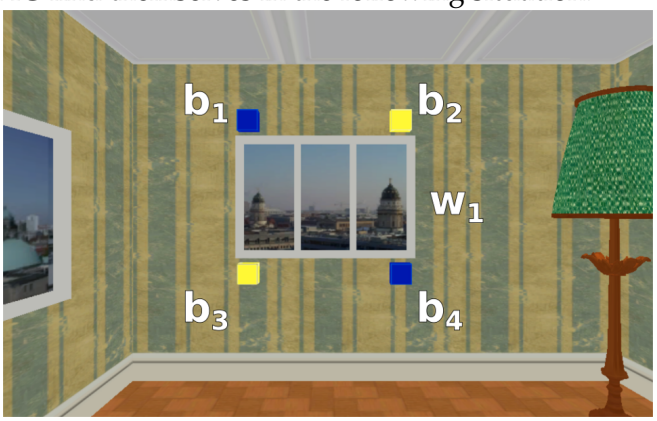
We would now like to present the IF with a corrective RE, but we cannot access $r_u$ directly because it exists only inside the IG's mind. We will instead reconstruct it, defining the noisy channel in more detail and selecting the most likely corruption of the original RE.

### 8.2.1 Finding the missing RE

We model the corruption of $r_s$ using edit operations at a word level in terms of the *Levenshtein edit distance* (Mohri, 2003). An RE is then a sequence of words over an alphabet $\Sigma$, and the noisy channel passes every word in $r_s$ applying either no operation ($K$), a deletion ($D$), or substituting the word with a new symbol $a \in \Sigma$ ($S_a$). The noisy channel may also insert new symbols $a \in \Sigma$ at any moment ($I_a$). Any sequence of such operations that could apply to $r_s$ is known as an *edit sequence for* $r_s$. If an edit sequence $s$ maps $x$ to $y$, we write $apply(s, x) = y$.

**Example: Finding $r_u$ (1/3)**

The IF and IG find themselves in the following situation:




The IG wants to refer to $o_s = b_4$, and presents the utterance $r_s =$ "the blue button below the window". The noisy channel corrupts this RE to $r_u =$ "the yellow button above the window", which the IF resolves to $o_u = b_2$. The corruption by the noisy channel could correspond, for instance, to the following edit operation sequence

| $r_s$ | the | blue | | button | below | the | window |
|---|---|---|---|---|---|---|---|
| edit operations | $K$ | $D$ | $I_{yellow}$ | $K$ | $S_{above}$ | $K$ | $K$ |
| $r_u$ | the | | yellow | button | above | the | window |

But it could also correspond to the following edit sequence

| $r_s$ | the | blue | button | below | the | window |
|---|---|---|---|---|---|---|
| edit operations | $K$ | $S_{yellow}$ | $K$ | $S_{above}$ | $K$ | $K$ |
| $r_u$ | the | yellow | button | above | the | window |

Next, we define a probability distribution $P(s|r_s)$ over edit sequences $s$ that the noisy channel might apply to the string $r_s$ as follows:

$$\begin{aligned} P(s|r_s) &= \frac{1}{Z}\prod_{s_i \in s} exp(-c(s_i)) \\ \text{where } c(s_i) &= \text{cost for using the edit operation } s_i \\ Z &= \text{normalizing constant, independent of } s \end{aligned} \tag{8.1}$$

We set $c(K) = 0$, and $c(S_a) = c(I_a) = c(D) = \mathcal{C}$ for any $a$ in our alphabet and fixed $\mathcal{C} > 0$.

Finally, let $L$ be the set of REs that could be resolved to $o_u$. Then the most probable edit sequence for $r_s$ that generates our missing $r_u \in L$ is given by:

$$\begin{aligned} s^* &= \underset{s\,:\,apply(s,r_s)\in L}{\arg\max} P(s \mid r_s) \\ &= \underset{s}{\arg\min} \sum_{s_i \in s} c(s_i) \end{aligned} \tag{8.2}$$

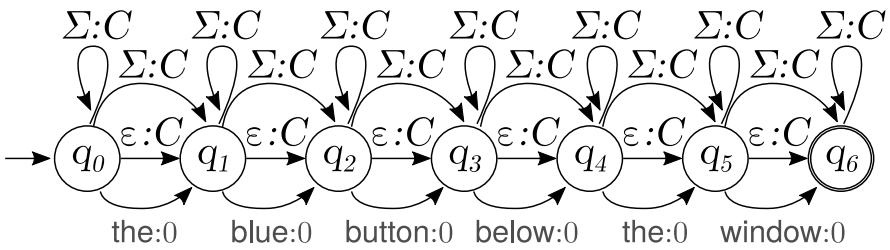
$s^*$ is the edit sequence that maps our original RE $r_s$ to an RE in $L$ with minimal cost. We assume that this is the edit sequence that corrupted $r_s$ into $r_u$. All that remains is finding this edit sequence.

Searching for $s^*$ by enumeration would be impractical: the set $L$ can be potentially infinite, and the set of possible edit sequences can be very large. Instead, we will use the chart representation that SIGs provide (see Chapter 2.3.2) represented as a Context-Free Grammar whose language $L = L(G)$ consists of these REs. This grammar is then intersected with a finite-state automaton (FSA) that keeps track of the edit costs, obtaining a second context-free grammar $G'$ from which $r_u$ can be efficiently obtained: the minimum-cost syntax tree of $G'$ is equivalent to the minimum-cost edit operation that transform $r_s$ into $r_u$.

Edit sequences can be compactly represented by a weighted, finite-state automaton $F(r_s)$ called the *edit automaton* (Mohri, 2003). Each run of the automaton on a string $w$ yields an edit sequence that transforms $r_s$ into $w$, and the sum of the transition costs is the cost of that sequence. $F(r_s)$ has a state $q_i$ for every position $i \in r_s$, with initial state $q_0$ and final state $q_{|r_s|}$. For each $i$, the state $q_i$ has three transitions from $q_i$ to $q_{i+1}$: the "keep" transition reads the word at position $i$ with cost 0; the "substitute" transition that reads any word with cost $\mathcal{C}$, and the "deletion" transition that read the empty string $\epsilon$ with cost $\mathcal{C}$. In addition, every state has an "insert" transition that loops back to the state $q_i$ with cost $\mathcal{C}$.

**Example: Finding $r_u$ (2/3)**

This is the edit automaton for $r_s$ = "the blue button below the window":




Note that every path through the edit transducer corresponds to a specific edit sequence $s$, and the sum of the costs along the path correspond to $-logP(s|r_s) - logZ$.

We obtain $G'$ intersecting the CFG $G$ and the FSA $F(r_s)$ using the Bar-Hillel construction (Bar-Hillel et al., 1961; Hopcroft et al., 1979). This construction intersects the languages of $F(r_s)$ and $G$ but, given that $F(r_s)$ accepts all strings over the alphabet, the languages of $G$ and $G'$ will be the same — namely, all REs for $o_u$.

At a glance, the process can be described as follows: for every production rule in $G$ we take a corresponding transition in $F(r_s)$, combine them into a new rule in the automaton $G'$, and assign this rule the right weight which corresponds with weights on $f(r_s)$. We simplify the process assuming that $G$ is in Chomsky Normal Form (CNF), where all rules have either the form $A \rightarrow a$ (where $a$ is a word) or $A \rightarrow BC$ (where $B, C$ are non-terminals). We represent non-terminals in the resulting grammar $G'$ as $N_{b,A,\langle q_i,q_k\rangle}$, where $a, B$ are as detailed in Chapter 2.3.2 and $q_i, q_k$ indicate that the string derived by this non-terminal was generated by editing the substring of $r_s$ from position $i$ to $k$.

For a production rule $N_{b,A} \rightarrow a$ of G, we take a transition $t = q_i \rightarrow \langle a : c\rangle q_k$ in $F(r_s)$, where $q$ and $q'$ are states of $F(r_s)$ and the transition cost is $c$. We create a context-free rule $N_{b,A,\langle q_i,q_k\rangle} \rightarrow a$ in $G'$ with weight $c$. If $k = i$ we have an Insertion operation ($I$), while $k = i + 1$ corresponds to either Substitution ($S$) or Keep ($K$).

With $G$ in CNF, the second type of rules are binary rules $N_{b,A} \rightarrow X_{b_1,A_1} Y_{b_2,A_2}$. With $q_i, q_j, q_k$ states of $F(r_s)$ and $i \leq j \leq k$, we add a rule $N_{b,A,\langle q_i,q_k\rangle} \rightarrow X_{b_1,A_1,\langle q_i,q_j\rangle} Y_{b_2,A_2,\langle q_j,q_k\rangle}$ to $G'$. These rules do not encode any operations, and therefore their weight is 0.

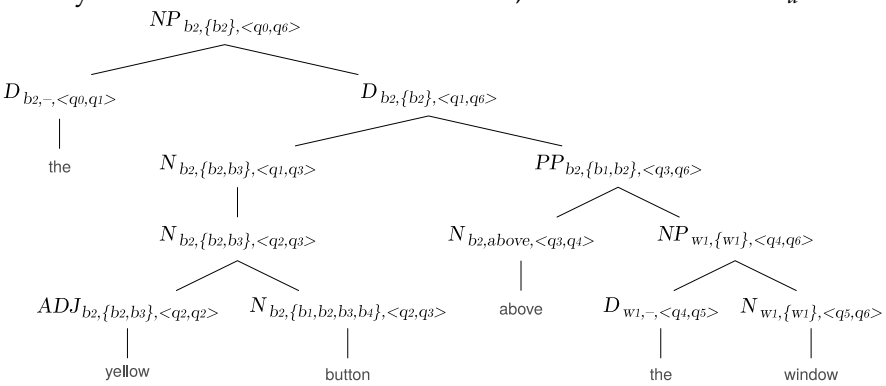
The next type of rules to account for are those that lead to deletions. If $N_{b,A}$ is a nonterminal symbol in $G$, and $q_h, q_i, q_j, q_k$ are states of $F(r_s)$ with $h \leq i \leq j \leq k$, we add a rule $N_{b,A,\langle q_h,q_k\rangle} \rightarrow N_{b,A,\langle q_i,q_j\rangle}$ to $G'$, which deletes the substrings both from positions $h$ to $i$ and from positions $j$ to $k$. The assigned weight is $\mathcal{C}((i-h)+(k-j))$, which corresponds to the number of $\epsilon$ transitions.

If $S_{b,A}$ is the start symbol of $g$, the start symbol of $G'$ is $S_{b,A,\langle q_0,q_{|r_s|}\rangle}$. As we said above, this construction accepts the same language of $G$, namely, all REs for $o_u$. Assigning weights as we've done it here, the weight for each

RE in $L(G)$ is the edit cost of that RE starting from $r_s$. Using the Viterbi algorithm, we compute the minimal-cost tree of $G'$, obtain $s^*$, and obtain $r_u = apply(s^*, r_s)$.

**Example: Finding $r_u$ (3/3)**

The following figure illustrates an example tree for the automaton show in the previous example. From the leaves of this tree, we obtain the string $w$ = "the yellow button above the window", which is an RE for $o_u$.




An analysis of the rules in $g'$ that led to this tree allows us to reconstruct the edit sequence that turned $r_s$ into $w$. "yellow" was created by an insertion, because the two states of $F(r_s)$ in the symbol above are the same. For "above", the two states in the preterminal symbol above it are different, meaning that the operation is either $I$ or $K$. We disambiguate using the weight of the rule, since $S$ operations have weight $C$ and $K$ operations have weight 0. Finally unary rules indicate deletions, since they "move forward" in $r_s$ without adding any new words to $w$. As a result, we obtain both $w$ and the corresponding edit sequence $\langle K\, D\, I_{yellow}\, K\, S_{above}\, K\, K\rangle$. If $w$ is the minimal-cost tree, then $w = r_u$.

## 8.3 Generation of contrastive feedback

Having the original RE $r_s$ and the corrupted RE $r_u$, we first generate contrastive feedback assigning focus to the words in $r_s$ that were changed by the corruption (that is, words to which either a Deletion or Substitution was applied). This strategy is called *Emphasis*

**Example: Contrastive focus 1/2**

For the edit sequence shown in Example 3/3, the words *blue* and *below* were changed by the noisy channel. Therefore, the correction should read "No, the **BLUE** button **BELOW** the window".

A second strategy attempts to generate shorter REs while retaining contrastive focus. We would expect these REs to be preferred by the IFs, since

shorter REs that do not violate the Maxim of Quantity are assumed to be preferred (Grice, 1975; Dale and Reiter, 1995).

Under this strategy, we attempt to remove information from $r_u$ that the IF understood correctly and should no longer need. This strategy is called *Shortening*, and can be described as follows.

The REs that our grammar generates are often a combination of an NP and a PP, such as "blue button (NP) below the window (PP)". If all errors are located in the NP, we can drop it entirely and generate a new RE of the form "the <NP>", with the NP emphasized as before. If all errors are located in the PP, we can generate a new RE of the form "the one <PP>", with the PP emphasized as before. Finally, if there is either no PP or errors in both of them, then the result is the same as in *Emphasis*.

**Example: Contrastive focus 2/2**
If we assume the intended object to be $b_4$ and $r_s$ to be "the blue button below the window", we could imagine that the user resolves the RE to $b_3$ due to the corrupted RE "the yellow button below the window". Both REs are a combination of the NP "blue/yellow button" and the PP "below the window". Given that the PP is the same for both, it can be dropped. As a result, *Shortening* generates the corrective RE "No, the **BLUE** button".

## 8.4 Experimental setup

We evaluated our strategies against several baselines using crowdsourced pair-wise experiments. Following Buß et al. (2010), we performed overhearer experiments that allowed us to test the effects of contrastive feedback without the navigational and timing challenges that a fully interactive system presents.

We created scenes in two different environments: the first set of scenes comprised images from the GIVE Challenge, while the second set used stimuli obtained from the "people" domain of the TUNA Reference Corpus (van der Sluis et al., 2007). The GIVE environment provided us with a familiar environment in which to test several attributes of an object at once, namely type of object, color, and spatial relation to other objects. The TUNA Corpus was chosen because it represents a more challenging domain, as the available properties for each object is larger: the corpus consists of photographs of men annotated with up to nine attributes, such as whether the person is wearing a shirt, a tie, or is looking right. Six of these attributes were included after preliminary studies showed that human REs often made use of them.

The test setup was identical for both environments: our crowdsourced participants were shown marked "before" and "after" screenshots, explain-

We wanted our player to select this button:

So we told them: *press the red button to the right of the blue button*.
But they selected this button instead:

Which correction is better for this scene?
- ○ No, press the red **BUTTON** to the right of the **BLUE BUTTON**
- ○ No, press the red button to the **RIGHT** of the blue button

(a) A sample scene from Experiment 1.

We wanted our player to select the person circled in green:

So we told them: *the light haired old man in a suit looking straight*.

But they selected the person circled in red instead.
Which correction is better for this scene?

- ○ No, the light haired old man **IN A SUIT LOOKING STRAIGHT**
- ○ No, the **LIGHT HAIRED OLD** man in a suit looking straight

(b) A sample scene from Experiment 2.

Figure 8.2: Sample scenes from our experiments in the GIVE and Tuna domains.

ing that we intended an imaginary player to choose the object in the first screenshot but they chose the second one instead. After showing our participants the text of the original RE, we asked them to choose one out of two possible corrections as the "better" one, where "better" was left up to interpretation. Figure 8.2 shows sample scenes from each domain.

In addition to our *Emphasis* and *Shortening* strategies, we included two baselines in our tests. The *Repeat* strategy presented the same original RE without any marked focus, to test whether the presence of explicit focus by itself is enough to prefer a certain RE. The second baseline strategy *Random* randomly capitalized adjectives, adverbs, and/or prepositions that were not capitalized by the *Emphasis* strategy. This strategy tests the assertion that, even if our participants were to prefer REs with contrastive focus due to the presence of focus itself, they would still prefer our main strategies because they place focus precisely were the subjects would expect it to be.

Following the guidelines detailed in Chapter 6, we included scenes with a clearly wrong answer (such as REs referring to the wrong target or a non-existent one) for quality control purposes. Players that either failed one of these random checks or answered in less than 10 seconds were excluded from the final set. Our participants were asked to rate up to 12 comparisons, shown in groups of 3 scenes at a time. The order in which the pair

of strategies were shown was also randomized, to avoid presentation order bias.

### 8.4.1 Experiment results

Our first experiment tested all four strategies against each other over the GIVE domain. Each subject was shown a total of 12 scenes, selected at random from 16 test scenes. We collected 10 judgments for each possible combination of GIVE scene and pair of strategies, yielding a total of 943 judgments from 142 subjects after quality control.

The results are shown in Table 8.1a. For each row strategy $Strat_R$ and each column strategy $Strat_C$, the table value corresponds to the formula

$$\frac{\#(Strat_R \text{ preferred over } Strat_C) - \#(Strat_C \text{ preferred over } Strat_R)}{\#(\text{tests between } Strat_R \text{ and } Strat_C)} \tag{8.3}$$

### 8.4.2 Experiment 2

Significance levels are taken from a two-tailed binomial test over the counts of preferences for each strategy.

| | **Repeat** | **Random** | **Emphasis** | **Shortening** |
|---|---|---|---|---|
| **Repeat** | – | 0.041 | -0.570* | -0.141 |
| **Random** | -0.041 | – | -0.600* | -0.109 |
| **Emphasis** | 0.570* | 0.600* | – | 0.376* |
| **Shortening** | 0.141 | 0.109 | -0.376* | – |

(a) Results for Experiment 1

| | **Repeat** | **Random** | **Emphasis** |
|---|---|---|---|
| **Repeat** | – | -0.425* | -0.575* |
| **Random** | 0.425* | – | -0.425* |
| **Emphasis** | 0.575* | 0.425* | – |

(b) Results for Experiment 2

Table 8.1: Pairwise comparisons between feedback strategies for experiments 1 and 2. A positive value shows preference for the row strategy, significant at * $p < 0.001$.

Our results show a significant preference for the *Emphasis* strategy over all others, providing evidence that our algorithm places contrastive focus in the correct places where a human would expect it. Although *Shortening* is numerically better than the other baselines, this result is not significant. It is significantly worse that *Emphasis*, a surprising result that will be discussed further in the next section of this Chapter.

Our second experiment repeats the same general setup over the TUNA domain, with minor modifications. The *Shortening* strategy was not included due to its poor performance both in the previous experiment and in further pilot experiments for this setup. Instead of using two screenshots, we presented the participants with a single 3x4 grid of randomly chosen people. This change was introduced to lower the required overhead of switching between two cluttered pictures. The REs used two to five attributes of each individual, but no information about their position with respect to other objects. We imposed this limitation to keep the length of the REs comparable to those of Experiment 1, and to avoid taxing our subjects' memory with extremely long REs.

We designed 8 different grids, and we obtained 240 judgments from 65 subjects (after quality control). Table 8.1b shows the results for this experiment, which once again confirms the preference of *Emphasis* over the baselines even in the presence of a larger number of attributes that could receive focus.

## 8.5 Discussion

Our experiments and results confirm that our strategy for computing contrastive REs works in practice. This is a welcome positive result, as it confirms that our simple corruption model can effectively discover which words need to be emphasized even when performing a shallow analysis at the word level.

Our results also show that users generally prefer REs with emphasis over unemphasized REs. This result is more pronounced in the TUNA corpus, where even *Random* is significantly preferred over *Repeat*. Even if the capitalization makes no semantic sense, we believe it does make *pragmatic* sense: the presence of contrastive focus signals publicly that a misunderstanding took place, and a correction is required. This behavior agrees with other linguistic studies. Calhoun et al. (2010) asserts that markedness encodes whether a word has a salient alternative in the context (contrast) or not (background). Gotzner et al. (2016) agrees with this interpretation, making an even stronger assertion: focus forces a listener to re-evaluate their context, as the presence of focus signals the existence of viable alternatives and triggers a "refresh" of the set of viable objects.

The poor performance of the *Shortening* strategy seems to be rooted on a similar set of challenges than those presented in Chapter 7.3.3. Our first instinct would tell us that a shorter RE should always be preferred, following the Gricean Maxim of Quantity (Grice, 1975). Our results, however, show that our shortened REs are *too* short, removing important information. How to shorten an RE without loss of clarity remains an open question.

## 8.6 Conclusion

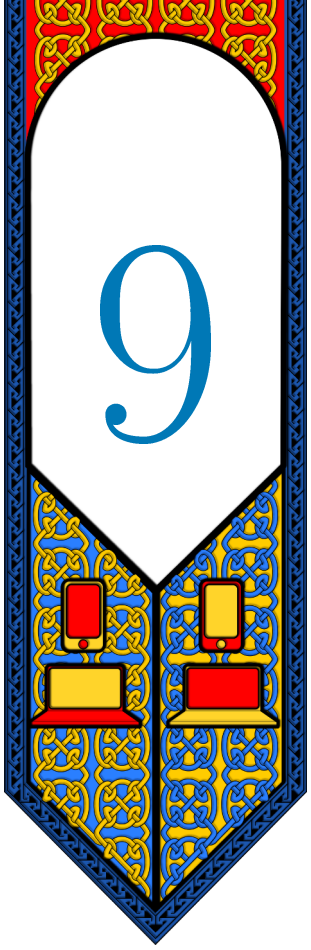
This Chapter introduces *Emphasis*, a strategy for the generation of contrastive, corrective feedback via contrastive focus.

We model misunderstandings as corruptions performed by a noisy channel causing an original RE $r_s$ intended for an object $o_s$ to be corrupted to an RE $r_u$ that resolves to a mistaken object $o_u$. Reconstructing $r_u$ as the RE which minimizes the corruption performed by the noisy channel, our strategy places contrastive focus in those words in the original RE $r_s$ that were corrupted by the noisy channel. This strategy is shown to be significantly preferred over other strategies of contrastive focus by human participants, as shown by crowdsourced preference experiments on scenes from both the GIVE Challenge and the TUNA people corpus.

The development of *Emphasis* completes the pipeline that we set to create for the prediction, detection, and correction of misunderstandings. The models of listener's understanding that we developed in Chapters 4 and 5 are of limited use without an effective way of correcting their predicted misunderstandings, making our strategy the key component that closes the feedback loop between Instruction Follower and Instruction Giver.

Unlike the strategies for corrective feedback presented in Chapter 7, *Emphasis* does not require a specific theory of mind for the Instruction Follower. Our strategy makes no assumptions regarding how REs are generated, requiring only a mechanism for representing all semantically-correct REs for a given object in a chart. Feedback generated this way is easy to compute and adapt to all kind of Instruction Giving systems.

Having developed our



# 9

# Conclusion

In this thesis we have explored and developed the components of a full, end-to-end system for the prediction, detection, and correction of misunderstandings.

Referring back to our motivating example in Page 1, we observed that an error correction strategy that waits until a misunderstanding takes place, ignores the source of the misunderstanding, and generates a completely new plan of action may be appropriate for cars, but it is sub-optimal when dealing with pedestrians. The algorithms and strategies we have developed and validated throughout this thesis would have painted a completely different picture: our probabilistic model would have reported at an early stage that Max' observed behavior does not match the behavior expected of a correctly resolved RE, revealing that Max is most likely confused. Instead of letting this mistake pass through, our system would have evaluated his current environment and behavior to decide which of all distractors is the most likely one to have captured Max' attention. Having decided on which misunderstanding is the most likely culprit for his observed behavior, our system would have generated a corrective, contrastive RE that would publicly display what their current joint objective is, that there is a possible mismatch in their common ground, and how to recover from it.

Having developed a robust strategy for dealing with misunderstandings in interactive dialog, this final Chapter briefly summarizes our approach, our main results, our challenges, and directions of future work.

## 9.1 Summary

The first two pillars of this thesis, the prediction and detection of misunderstandings, are introduced in terms of probabilistic, log-linear models of listener's understanding capable of inferring that a user is about to make a mistake. Our models can predict a misunderstanding at an early-enough stage of the interaction, providing an Instruction Giver with enough time to act on this misunderstanding before the user interacts with the wrong object. In addition to detecting that an RE was incorrectly resolved, these models also identify to which object was the RE resolved instead, opening the door for multiple correction strategies.

We also show that gaze information, obtained with eye-tracking equipment, improves the performance of our models in hard scenes. The results obtained using recorded gaze as a proxy to attention are doubly reassuring. The improved performance on hard scenes (but not on easy) are an indicator of our original, approximated features being appropriate for the task, as they can compete with real, actual measurings in all but the most challenging scenes. The poor performance of gaze and visual salience alone, on the other hand, hints at our models capturing more nuanced details of an interpretation process than those that might be obvious at first.

These topics were presented along detailed reviews of foundational work in the areas of pragmatics and psycholinguistics: pragmatics interprets misunderstandings as a common ground mismatch, psycholinguistics suggest explanations for the relation between eye gaze and attention, and both provide the reader with a solid background to understand the theoretical foundations on top of which our research rests.

The correction of misunderstandings, the third pillar of our work, is explored from two different perspectives: starting with an unsuccessful approach based in Context Sets our approach evolved towards *Emphasis*, a strategy for the generation of contrastive, corrective feedback that is shown to be significantly preferred by human participants over several others. The successes and failures of each one of our systems has been presented in detail, detailing a set of lessons learned that increase our understanding of how a user's focus changes in response to corrections.

Each step of this pipeline has been experimentally verified, and the end result is a robust combination of theory and practice that will no doubt be of help for future researchers on interactive dialog.

Furthermore, we have demonstrated how to strengthen the integrity of crowdsourced experiments with the help of a security mindset, a perspective that is vital for researchers dealing with anonymous participants over the internet. Researchers wondering whether crowdsourcing is right for them will find our analysis of great help.

## 9.2 Future work

The work presented here opens several avenues for further research. One such research direction, related to our $P_{Obs}$ model and our experiments in Chapters 7 and 8, is the need of further studies on the role of gaze in Instruction Following. Eye-tracking research would present the ultimate evidence on the strengths and limitations of our $P_{Comb}$ model and, in particular, on the usefulness of our $P_{Obs}$-based context set. Recording the eye-movements of human participants and matching them to the probabilities returned by our log-linear models could precisely measure the contributions of our model and would be an important addition to the psycholinguistics literature. Experiments on the eye-movements of subjects after receiving contrastive feedback would also shed light on the interplay between a Context Set, the effect that corrections have on it, and the type and amount of information that can be removed from a correction before it becomes confusing. Refining this idea will surely lead to even shorter, easier to understand corrective REs than those we have developed here.

A second venue of research is the implementation of a new $P_{Comb}$ model that does not require hand-crafted features. An approach based in Deep Learning, for instance, would focus on the design of a network that can infer features from the raw data rather than testing plausible features the way we've done it here. Recurrent Neural Networks, with their ability to deal with sequential input, seem particularly well suited for the task. How to train and implement a network with higher accuracy and/or more efficient than with our current log-linear models seems like a straightforward direction for our research to evolve.

Another point of interest for researchers working on indoor and pedestrian navigation is improving the *VisualSalience* feature introduced in Section 4.2.2. This feature has proved to be useful for estimating which objects have captured a user's attention based on very limited information, and presents an alternative solution to the problem of giving instructions based on visual features without forcing a user to hold a camera in front of them. We look forward to this model being applied in other areas of study in Human-Computer Interaction, and for more accurate versions of our algorithm to be developed.

Our experimental results have shown that our strategies for shortening an RE require further work. Whether based on Context Sets or Edit Operations, our strategies generate REs that users find confusing. How to properly determine which information can be removed from an instruction based on both a previous instruction and a model of user understanding, or whether some crucial piece of information is missing from our hypothesis are questions for future researchers to answer.

Finally, researchers could adapt the results presented here in the GIVE Environment to new Virtual Environments. With the rising popularity of

creative 3D environments and advances in VR, the results presented here can enhance Instruction Giving systems in new domains. While our results are based on the Virtual Environment of the GIVE Challenge, there is no reason why our models, features, and strategies cannot be ported to other environments and REG systems.

# Bibliography



Akmajian, A. and Jackendoff, R. (1970). Coreferentiality and stress. *Linguistic inquiry*, 1(1):124–126.

Allingham, M. G. and Sandmo, A. (1972). Income tax evasion: a theoretical analysis. *Journal of Public Economics*, 1(3):323 – 338.

Altamirano, R., Areces, C., and Benotti, L. (2012). Probabilistic refinement algorithms for the generation of referring expressions. In *Proceedings of COLING 2012: Posters*, pages 53–62, Mumbai, India. The COLING 2012 Organizing Committee.

Appelt, D. and Kronfelt, A. (1987). A computational model of referring. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 640–647.

Areces, C., Koller, A., and Striegnitz, K. (2008). Referring expressions as formulas of description logic. In *Proceedings of the 5th International Conference on Natural Language Generation (INLG)*.

Austin, J. L. (1962). *How to Do Things with Words*. Clarendon Press.

Bar-Hillel, Y., Perles, M., and Shamir, E. (1961). On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

Benotti, L. and Denis, A. (2011). Cl system: Giving instructions by corpus based selection. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 296–301. Association for Computational Linguistics.

Benotti, L., Lau, T. A., and Villalba, M. (2014). Interpreting natural language instructions using language, vision, and behavior. *TiiS*, 4(3):13:1–13:22.

Benotti, L., Villalba, M., Lau, T. A., and Cerruti, J. A. (2012). Corpus-based interpretation of instructions in virtual environments. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 181–186.

Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Berinsky, A. J., Huber, G. A., and Lenz, G. S. (2012). Evaluating online labor markets for experimental research: Amazon.com's mechanical turk. *Political Analysis*, 20(3):351âĂŞ–368.

Bichot, N. P. (2001). Attention, eye movements, and neurons: Linking physiology and behavior. In Jenkin, M. and Harris, L., editors, *Vision and Attention*, pages 209–232. Springer New York, New York, NY.

Bornkessel, I. and Schlesewsky, M. (2006). The role of contrast in the local licensing of scrambling in german: Evidence from online comprehension. *Journal of Germanic Linguistics*, 18(01):1–43.

Bosch, P. (1988). Representing and accessing focussed referents. *Language and Cognitive Processes*, 3(3):207–232.

Branavan, S. R. K., Chen, H., Zettlemoyer, L. S., and Barzilay, R. (2009). Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 1 of *ACL '09*, pages 82–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Broadbent, D. (1958). *Perception and communication*. Pergamon Press.

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

Bülthoff, H. H. and van Veen, H. A. H. C. (2001). Vision and action in virtual environments: Modern psychophysics in spatial cognition research. In Jenkin, M. and Harris, L., editors, *Vision and Attention*, pages 233–252, New York, NY. Springer New York.

Bundesen, C. (1990). Theory of visual attention. *Psychological review*, 97:523–547.

Bundesen, C. and Habekost, T. (2014). Theory of visual attention (tva). In Nobre, A. C. K. and Kastner, S., editors, *The Oxford Handbook of Attention*, chapter 37, pages 1095–1121. Oxford University Press, Oxford.

Buß, O., Baumann, T., and Schlangen, D. (2010). Collaborating on utterances with a spoken dialogue system using an isu–based approach to incremental dialogue management. In *Proceedings of the Special Interests Group on Discourse and Dialogue Conference (SIGdial 2010)*.

Byron, D., Koller, A., Striegnitz, K., Cassell, J., Dale, R., Moore, J., and Oberlander, J. (2009). Report on the First NLG Challenge on Generating Instructions in Virtual Environments (GIVE). In *Proceedings of the 12th European Workshop on Natural Language Generation (Special session on Generation Challenges)*.

Byron, D. K. and Fosler-Lussier, E. (2006). The osu quake 2004 corpus of two-party situated problem-solving dialogs. In *Proceedings of the 15th Language Resources and Evaluation Conference (LREC'06)*.

Calhoun, S., Carletta, J., Brenier, J. M., Mayo, N., Jurafsky, D., Steedman, M., and Beaver, D. (2010). The nxt-format switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation*, 44(4):387–419.

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Cassin, B., Solomon, S., and Rubin, M. (1984). *Dictionary of eye terminology*. Triad Pub. Co.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAIâĂŹ11)*, pages 859–865.

Clark, H. H. (1996). *Using Language*. Cambridge University Press.

Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, 22:1–39.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M., and Löding, C. (2007). *Tree Automata techniques and applications*. published online - `http://tata.gforge.inria.fr/`.

Conti, G. and Caroland, J. (2011). Embracing the kobayashi maru: Why you should teach your students to cheat. *IEEE Security and Privacy*, 9(4):48–51.

Corrigan-Gibbs, H., Gupta, N., Northcutt, C., Cutrell, E., and Thies, W. (2015). Deterring cheating in online environments. *ACM Transactions on Computer-Human Interactions*, 22(6):28:1–28:23.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Dale, R. (1989). Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 68–75.

Dale, R. (1992). *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*. The MIT Press.

Dale, R. and Reiter, E. (1995). Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. (2017). Embodied Question Answering. arXiv:1711.11543.

de Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., and Kiela, D. (2018). Talk the Walk: Navigating New York City through Grounded Dialogue. arXiv:1807.03367.

Desimone, R. and Duncan, J. (1995). Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18:193–222.

Deutsch, J. A. and Deutsch, D. (1963). Attention: Some theoretical considerations. *Psychological Review*, 70(1):1–10.

DeVault, D., Rich, C., and Sidner, C. L. (2004). Natural language generation and discourse context: Computing distractor sets from the focus stack. In *International Florida Artificial Intelligence Research Symposium (FLAIRS)*.

Di Eugenio, B. (1992). Understanding natural language instructions: The case of purpose clauses. In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics*, ACL '92, pages 120–127, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dreyfuss, E. (2018). A bot panic hits amazon's mechanical turk. *Wired*. [Online; posted 17-August-2018].

Eickhoff, C. and de Vries, A. P. (2013). Increasing cheat robustness of crowdsourcing tasks. *Information Retrieval*, 16(2):121–137.

Engonopoulos, N. and Koller, A. (2014). Generating effective referring expressions using charts. In *Proceedings of the 8th International Conference on Natural Language Generation (INLG)*, Philadelphia.

Engonopoulos, N., Villalba, M., Titov, I., and Koller, A. (2013). Predicting the resolution of referring expressions from user behavior. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, WA.

Fernández, R. and Schlangen, D. (2007). Referring under restricted interactivity conditions. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*.

Fikes, R. and Nilsson, N. (1971). Strips: a new approach to theorem proving in problem solving. *Artificial Intelligence*, 2:189–208.

Fillmore, C. J. (1974). Pragmatics and the description of discourse. In Schmidt, S. J., editor, *Pragmatics II*, pages 83–104. Wilhelm Fink Verlag, Munich.

Folk, C. L. (2015). Controlling spatial attention: Lessons from the lab and implications for everyday life. In Fawcett, J. M., Risko, E. F., and Kingstone, A., editors, *The Handbook of Attention*, chapter 1, pages 3–25. MIT Press, Cambridge, Massachusetts.

Fraundorf, S. H., Benjamin, A. S., and Watson, D. G. (2013). What happened (and what didn't): Discourse constraints on encoding of plausible alternatives. *Journal of Memory and Language*, 69(3):196 – 227.

Frege, G. (1884). *Die Grundlagen der Arithmetik – Eine logisch mathematische Untersuchung über den Begriff der Zahl.(GLA)*. Verlag von Wilhelm Koebner, Breslau.

Freundschuh, S. M. and Egenhofer, M. J. (1997). Human conceptions of spaces: Implications for geographic information systems. *Transactions in GIS*, 2(4):361–375.

Gadiraju, U., Kawase, R., Dietze, S., and Demartini, G. (2015). Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1631–1640, New York, NY, USA. ACM.

Gadiraju, U., Möller, S., Nöllenburg, M., Saupe, D., Egger-Lampl, S., Archambault, D., and Fisher, B. (2017). *Crowdsourcing Versus the Laboratory: Towards Human-Centered Experiments Using the Crowd*, pages 6–26. Springer International Publishing, Cham.

Gardent, C. (2002). Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 96–103. Association for Computational Linguistics.

Gargett, A., Garoufi, K., Koller, A., and tina Striegnitz, K. (2010). The give-2 corpus of giving instructions in virtual environments. In Chair), N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Garoufi, K. and Koller, A. (2011a). Combining symbolic and corpus-based approaches for the generation of successful referring expressions. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*, Nancy.

Garoufi, K. and Koller, A. (2011b). The Potsdam NLG systems at the GIVE-2.5 Challenge. In *GIVE-2.5 Challenge: System descriptions*, Nancy.

Garrod, S. C. and Sanford, A. J. (1982). The mental representation of discourse in a focussed memory system: Implications for the interpretation of anaphoric noun phrases. *Journal of semantics*, 1(1):21–41.

Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Gilovich, T., Keltner, D., Chen, S., and Nisbett, R. (2015). *Social Psychology*. W.W. Norton.

Glantz, K., Durlach, N. I., Barnett, R. C., and Aviles, W. A. (1997). Virtual reality (VR) and psychotherapy: Opportunities and challenges. *Presence: Teleoperators and Virtual Environments*, 6(1):87–105.

Gorniak, P. and Roy, D. (2004). Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research*, 21:429–470.

Gotzner, Wartenburger, and Spalek (2016). The impact of focus particles on the recognition and rejection of contrastive alternatives. *Language and Cognition*, 8:59–95.

Gotzner, N. (2017). *What's Included in the Set of Alternatives?*, pages 103–122. Springer International Publishing, Cham.

Grice, H. P. (1975). Logic and conversation. In Cole, P. and Morgan, J. L., editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 41–58. Academic Press, New York.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Hallinan, J. T. (2009). *Why we make mistakes*. Broadway Books.

Haponchyk, I., Uva, A., Yu, S., Uryupina, O., and Moschitti, A. (2018). Supervised clustering of questions into intents for dialog system applications. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 2310–2321.

Harris, L. R. and Jenkin, M. (2001). Vision and attention. In Jenkin, M. and Harris, L., editors, *Vision and Attention*, pages 1–17. Springer New York, New York, NY.

Harris, M. (2015). How a lone hacker shredded the myth of crowdsourcing. *Wired*. [Online; posted 09-February-2015].

Hoffmann, J. and Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston, MA, USA.

Horacek, H. (2004). On referring to sets of objects naturally. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG)*, pages 70–79, Brokenhurst.

Ilinykh, N., Zarrieß, S., and Schlangen, D. (2018). The task matters: Comparing image captioning and task-based dialogical image descriptions. In *Proceedings of the 11th International Conference on Natural Language Generation (INLG 2018)*, pages 397–402.

Itti, L. and Borji, A. (2014). Computational modelsbottom-up and top-down aspects. In Nobre, A. C. K. and Kastner, S., editors, *The Oxford Handbook of Attention*, chapter 38, pages 1122–1158. Oxford University Press, Oxford.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.

Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *Physical Review*, 106(4):620–630.

Kelleher, J. and Kruijff, G.-J. (2006). Incremental generation of spatial referring expressions in situated dialogue. In *Proceedings of COLING-ACL 2006*, Sydney, Australia.

Kelleher, J. and van Genabith, J. (2004). Visual salience and reference resolution in simulated 3-d environments. *Artificial Intelligence Review*, 21(3):253–267.

Klein, W. (1982). Local deixis in route directions. In Klein, R. J. . W., editor, *Speech, Place, and Action: Studies in Deixis and Related Topics*, pages 161–182. Wiley, Chichester.

Koch, C. and Ullman, S. (1985). Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227.

Koleva, N., Villalba, M., Staudte, M., and Koller, A. (2015). The impact of listener gaze on predicting reference resolution. In *ACL (2)*, pages 812–817.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010). Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, HRI '10, pages 259–266, Piscataway, NJ, USA. IEEE Press.

Koller, A. and Engonopoulos, N. (2017). Integrated sentence generation with charts. In *Proceedings of the 10th International Conference on Natural Language Generation (INLG)*, Santiago de Compostela.

Koller, A., Garoufi, K., Staudte, M., and Crocker, M. (2012). Enhancing referential success by tracking hearer gaze. In *Proceedings of the 13th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, Seoul.

Koller, A. and Stone, M. (2007). Sentence generation as a planning problem. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 336–343. Association for Computational Linguistics.

Koller, A., Striegnitz, K., Gargett, A., Byron, D., Cassell, J., Dale, R., Moore, J., and Oberlander, J. (2010). Report on the Second NLG Challenge on Generating Instructions in Virtual Environments (GIVE-2). In *Proceedings of the Sixth International Natural Language Generation Conference (Special session on Generation Challenges)*.

Koolen, R., Gatt, A., Goudbeek, M., and Krahmer, E. (2009). Need i say more? on factors causing referential overspecification. In *Proceedings of the CogSci Workshop on the Production of Referring Expressions (PRE-CogSci 2009)*, Amsterdam.

Krahmer, E. and Theune, M. (1999). Efficient generation of descriptions in context. In van Deemter, K. and Kibble, R., editors, *Proceedings of the ESSLI Workshop on the Generation of Nominals*, volume 99, Utrecht, The Netherlands.

Krahmer, E. and Theune, M. (2002). Efficient context-sensitive generation of referring expressions. In van Deemter, K. and Kibble, R., editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, volume 143, pages 223–263. CSLI Publications.

Krahmer, E. and van Deemter, K. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Krahmer, E., van Erk, S., and Verleg, A. (2003). Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Krahmer, E. J. and Theune, M. (1998). Context sensitive generation of descriptions. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*, pages 1151–1154, Sydney, Australia. International Speech Communication Association (ISCA).

Krifka, M. (2008). Basic notions of information structure. *Acta Linguistica Hungarica*, 55:243–276.

Land, M. F. and Hayhoe, M. (2001). In what ways do eye movements contribute to everyday activities? *Vision Research*, 41(25-26):3559–3565.

Lau, T., Drews, C., and Nichols, J. (2009). Interpreting written how-to instructions. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence*, pages 1433–1438.

Lavie, N. and Dalton, P. (2014). Load theory of attention and cognitive control. In Nobre, A. C. K. and Kastner, S., editors, *The Oxford Handbook of Attention*, chapter 3, pages 56–75. Oxford University Press, Oxford.

Levelt, W. J. (1993). *Speaking: From Intention to Articulation*. ACL-MIT Press series in natural-language processing. Bradford Books, U.S.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 8(10):707–710.

Levit, M. and Roy, D. (2007). Interpretation of spatial language in a map navigation task. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(3):667–679.

Lisovskaya, A. (2015). Designing an infrastructure for crowdsourcing experiments with a dialog system. Bachelor's thesis, Potsdam University.

Luck, S. J. and Vogel, E. K. (1997). The capacity of visual working memory for features and conjunctions. *Nature*, 390:279–281.

MacMahon, M. and Stankiewicz, B. (2006). Human and automated indoor route instruction following. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, Vancouver, BC.

Mast, V. and Wolter, D. (2013). A probabilistic framework for object descriptions in indoor route instructions. In Tenbrink, T., Stell, J., Galton, A., and Wood, Z., editors, *Spatial Information Theory*, pages 185–204, Cham. Springer International Publishing.

Matuszek, C., Fox, D., and Koscher, K. (2010). Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, HRI '10, pages 251–258, Piscataway, NJ, USA. IEEE Press.

Mazar, N., Amir, O., and Ariely, D. (2008). The dishonesty of honest people: A theory of self-concept maintenance. *Journal of Marketing Research*, 45(6):633–644.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Milosavljevic, M. and Dale, R. (1996). Strategies for comparison in encyclopædia descriptions. In *Proceedings of the 8th International Natural Language Generation Workshop (INLG 1996)*.

Mohri, M. (2003). Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389.

Nikravesh, M. and Bensafi, S. (2005). *Soft Computing for Perception-Based Decision Processing and Analysis: Web-Based BISC-DSS*, pages 93–188. Springer Berlin Heidelberg, Berlin, Heidelberg.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Partalas, I., Vrakas, D., and Vlahavas, I. (2008). Reinforcement learning and automated planning: A survey. In Vrakas, D. and Vlahavas, I., editors, *Advanced Problem Solving Techniques*. IGI Global.

Pednault, E. P. (1987). Formulating multiagent, dynamic-world problems in the classical planning framework. In *Reasoning about actions & plans*, pages 47–82. Elsevier.

Pierrehumbert, J. B. and Hirschberg, J. (1990). The meaning of intonational contours in the interp retation of discourse. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, *Intentions in Communication*, chapter 14. MIT University Press Group.

Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., and Sykes, C. (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173:789–816.

Posner, M. I. (1980). Orienting of attention. *Quarterly Journal of Experimental Psychology*, 32(1):3–25.

Purver, M. (2004). *The Theory and Use of Clarification Requests in Dialogue*. PhD thesis, KingâĂŹs College, University of London.

Ratnaparkhi, A. (1997). A simple introduction to maximum entropy models for natural language processing. Technical Report IRCS-97-08, University of Pennsylvania.

Ratnaparkhi, A. (1998). Maximum entropy models for natural language ambiguity resolution. Technical Report IRCS-98-15, University of Pennsylvania.

Reiter, E. and Dale, R. (1997). Building natural-language generation systems. *Natural Language Engineering*, 3:57–87.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press.

Rooth, M. (1992). A theory of focus interpretation. *Natural Language Semantics*, 1:75–116.

Rooth, M. (1997). Focus. In Lappin, S., editor, *The Handbook of Contemporary Semantic Theory*, chapter 10, pages 271–298. Blackwell Publishing.

Ross, J., Irani, L., Silberman, M. S., Zaldivar, A., and Tomlinson, B. (2010). Who are the crowdworkers?: Shifting demographics in mechanical turk. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 2863–2872, New York, NY, USA. ACM.

Rüetschi, U. (2007). *Wayfinding in Scene Space: Modelling Transfers in Public Transport*. PhD thesis, University of Zürich.

Russell, S. J. and Norvig, P. (2004). *Inteligencia Artificial: Un enfoque moderno*. Prentice Hall, 2 edition.

Sanford, A. J. and Garrod, S. C. (1981). *Understanding written language : explorations of comprehension beyond the sentence*. Chichester: John Wiley.

Schneier, B. (2008). Inside the twisted mind of the security professional. *Wired*. [Online; posted 20-March-2008].

Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cam: Verschiedene Aufl. Cambridge University Press.

Smith, D. A. and Lieberman, H. (2013). Interpreting vague and ambiguous referring expressions by dynamically binding to properties of the context set. In *Modeling and Using Context: Proceedings of the 8th International and Interdisciplinary Conference (CONTEXT 2013)*, pages 15–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

Staff, U.C.P. (2017). *The Chicago Manual of Style*. Chicago Manual of Style. University of Chicago Press.

Staudte, M., Koller, A., Garoufi, K., and Crocker, M. (2012). Using listener gaze to augment speech generation in a virtual 3D environment. In *Proceedings of the 34th Annual Meeting of the Cognitive Science Society (CogSci)*, Sapporo.

Stefanovitch, N., Alshamsi, A., Cebrian, M., and Rahwan, I. (2014). Error and attack tolerance of collective problem solving: The darpa shredder challenge. *EPJ Data Science*, 3(1):13.

Steube, A. (2001). Correction by contrastive focus. *Theoretical Linguistics*, 27(2-3):215–250.

Stoia, L., Shockley, D. M., Byron, D. K., and Fosler-Lussier, E. (2008). Scare: A situated corpus with annotated referring expressions. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakesh, Morocco.

Striegnitz, K., Denis, A., Gargett, A., Garoufi, K., Koller, A., and Theune, M. (2011). Report on the Second Second Challenge on Generating Instructions in Virtual Environments (GIVE-2.5). In *Proceedings of the 13th European Workshop on Natural Language Generation (Special session on Generation Challenges)*, Nancy.

Sulem, E., Abend, O., and Rappoport, A. (2018). Bleu is not suitable for the evaluation of text simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744. Association for Computational Linguistics.

Suri, S., Goldstein, D. G., and Mason, W. A. (2011). Honesty in an online labor market. In *Human Computation*, pages 61–66, San Francisco, California, USA. Association for the Advancement of Artificial Intelligence.

Tanenbaum, A. S. (1988). *Computer networks*. Prentice-Hall software series. Prentice-Hall.

Theeuwes, J. (2014). Spatial orienting and attentional capture. In Nobre, A. C. K. and Kastner, S., editors, *The Oxford Handbook of Attention*, chapter 8, pages 231–252. Oxford University Press, Oxford.

Thompson, H. S., Anderson, A., Bard, E. G., Doherty-Sneddon, G., Newlands, A., and Sotillo, C. (1993). The hcrc map task corpus: Natural dialogue for speech recognition. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.

van Deemter, K. (2016). *Computational Models of Referring: a Study in Cognitive Science*. The MIT Press, Cambridge, Mass.

van der Sluis, I., Gatt, A., and van Deemter, K. (2007). Evaluating algorithms for the generation of referring expressions: Going beyond toy domains. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*.

Vaughan, J. W. (2018). Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18(193):1–46.

Vecera, S. P. and Behrmann, M. (2001). 6 - attention and unit formation: A biased competition account of object-based attention. In Shipley, T. F. and Kellman, P. J., editors, *From Fragments to Objects*, volume 130 of *Advances in Psychology*, pages 145 – 180. North-Holland.

Viethen, J. and Dale, R. (2008). The use of spatial relations in referring expression generation. In *Proceedings of the Fifth International Natural Language Generation Conference*, INGL '08, pages 59–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

Villalba, M., Teichmann, C., and Koller, A. (2017). Generating contrastive referring expressions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 678–687. Association for Computational Linguistics.

Vogel, A. and Jurafsky, D. (2010). Learning to follow navigational directions. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 806–814.

Winograd, T. (1972). *Understanding Natural Language*. Academic Press.

Wu, Y., Wu, Y., Gkioxari, G., and Tian, Y. (2018). Building generalizable agents with a realistic and rich 3d environment. arXiv:1801.02209.

Ylonen, T. and Lonvick, C. (2006). The secure shell (ssh) connection protocol. RFC 4254, RFC Editor.

Yu, D., Deng, L., and Acero, A. (2009). Using continuous features in the maximum entropy model. *Pattern Recognition Letters*, 30(14):1295 – 1300.