INLG and SIGDIAL 2014

Proceedings of the INLG and SIGDIAL 2014 Joint Session

Organizers:
Margaret Mitchell, Kathleen McCoy, David McDonald, Aoife Cahill, Ani Nenkova, and Helen Hastie

19 June 2014
Philadelphia, PA, USA

The first joint session of the ACL Special Interest Groups on Natural Language Generation (SIGGEN) and Discourse and Dialogue (SIGDIAL)

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-941643-23-5

Introduction

Welcome to the first joint session of INLG and SIGDIAL! INLG is the biennial meeting of the ACL Special Interest Group on Natural Language Generation (SIGGEN), and SIGDIAL is the annual meeting of the ACL Special Interest Group on Discourse and Dialogue (SIGDIAL). 3 papers (2 long, 1 short) were accepted for a special joint session on generation and dialogue, included in this document. This marks the first year where researchers in SIGDIAL and SIGGEN will come together to discuss issues relevant to both communities.

Margaret Mitchell, Kathleen McCoy, David McDonald, Aoife Cahill, Ani Nenkova, and Helen Hastie
INLG 2014 and SIGDIAL 2014 Program Chairs

Organizers:

Margaret Mitchell, Microsoft Research (MSR)
Kathleen McCoy, University of Delaware
David McDonald, Smart Information Flow Technologies (SIFT)
Aoife Cahill, Educational Testing Service (ETS)
Ani Nenkova, University of Pennsylvania
Helen Hastie, Heriot Watt University

Table of Contents

INLG and SIGDIAL 2014 Joint Session Proceedings

Modeling Blame to Avoid Positive Face Threats in Natural Language Generation

Gordon Briggs
Human-Robot Interaction Laboratory
Tufts University
Medford, MA USA
gbriggs@cs.tufts.edu

Matthias Scheutz
Human-Robot Interaction Laboratory
Tufts University
Medford, MA USA
mscheutz@cs.tufts.edu

Abstract

Prior approaches to politeness modulation in natural language generation (NLG) often focus on manipulating factors such as the directness of requests that pertain to preserving the autonomy of the addressee (negative face threats), but do not have a systematic way of understanding potential impoliteness from inadvertently critical or blame-oriented communications (positive face threats). In this paper, we discuss ongoing work to integrate a computational model of blame to prevent inappropriate threats to positive face.

1 Introduction

When communicating with one another, people often modulate their language based on a variety of social factors. Enabling natural and human-like interactions with virtual and robotic agents may require engineering these agents to be able to demonstrate appropriate social behaviors. For instance, increasing attention is being paid to the effects of utilizing *politeness* strategies in both human-computer and human-robot dialogue interactions (Cassell and Bickmore, 2003; Torrey et al., 2013; Strait et al., 2014). This work has shown that, depending on context, the deployment of politeness strategies by artificial agents can increase human interactants' positive assessments of an agent along multiple dimensions (e.g. likeability).

However, while these studies investigated the human factors aspects of utilizing politeness strategies, they were not concerned with the natural language generation (NLG) mechanisms necessary to appropriately realize and deploy these strategies. Instead, there is a small, but growing, body of work on natural language generation architectures that seek to address this challenge (Gupta et al., 2007; Miller et al., 2008; Briggs and Scheutz, 2013). The common approach taken by these architectures is the operationalization of key factors in Brown and Levinson's seminal work on *politeness theory*, in particular, the degree to which an utterance can be considered a *face-threatening act* (FTA) (Brown and Levinson, 1987).

While this prior work demonstrates the abilities of these NLG architectures to successfully produce polite language, there remain some key challenges. Perhaps the most crucial question is: how does one calculate the degree to which an utterance is a FTA[1]? This is a complex issue, as not only is this value modulated by factors such as social distance, power, and context, but also the multifaceted nature of "face." An utterance may be polite in relation to *negative face* (i.e. the agent's autonomy), but may be quite impolite with regard to *positive face* (i.e. the agent's image and perceived character).

In this paper, we investigate the problem of modeling threats to positive face. First we discuss how prior work that has focused primarily on mitigating threats to negative face, and examine a specific example, taken from the human subject data of (Gupta et al., 2007), to show why accounting for positive face is necessary. Next, we discuss our proposed solution to begin to model threats to positive face– specifically, integrating a computational model of blame. Finally, we discuss the justification behind and limitations of this proposed approach.

2 Motivation

Brown and Levinson (1987) articulated a taxonomy of politeness strategies, distinguishing broadly between the notion of positive and negative politeness (with many distinct strategies for each). These categories of politeness correspond

[1]Less crucially, what is the appropriate notation for this value? It is denoted differently in each paper: Θ, W, and η.

to the concepts of positive and negative face, respectively. An example of a positive politeness strategy is the use of praise ("Great!"), whereas a common negative politeness strategy is the use of an *indirect speech act* (ISA), in particular, an indirect request. An example of an indirect request is the question, "Could you get me a coffee?", which avoids the autonomy-threatening direct imperative, while still potentially being construed as a request. This is an example of a conventionalized form, in which the implied request is more directly associated with the implicit form. Often considered even less of a threat to negative face are unconventionalized ISAs, which often require a deeper chain of inference to derive their implied meaning. It is primarily the modulation of the level of request indirectness that is the focus of (Gupta et al., 2007; Briggs and Scheutz, 2013).

To provide an empirical evaluation of their system, Gupta et al. (2007) asked human subjects to rate the politeness of generated requests on a five-point Likert scale in order of most rude (1) to to most polite (5). The results from (Gupta et al., 2007) for each of their politeness strategy categories are below:

1. Autonomy [3.4] (e.g. "Could you possibly do X for me?")
2. Approval [3.0] (e.g. "Could you please do X mate?")
3. Direct [2.0] (e.g. "Do X.")
4. Indirect [1.8] (e.g. "X is not done yet.")

This finding is, in some sense, counterintuitive, as unconventionalized request forms should be the least face-threatening. However, Gupta et al. (2007) briefly often an explanation, saying that the utterances generated in the indirect category sound a bit like a "complaint or sarcasm." We agree with this assessment. More precisely, while negative face is protected by the use of their unconventionalized ISAs, positive face was not.

To model whether or not utterances may be interpreted as being complaints or criticisms, we seek to determine whether or not they can be interpreted as an act of *blame*[2].

[2]What the precise ontological relationship is between concepts such as complaining, criticizing, and blaming is beyond the scope of this paper.

3 Approach

Like praise, blame (its negative counterpart) is both a cognitive and social phenomenon (Malle et al., 2012). The cognitive component pertains to the internal attitudes of an agent regarding another agent and their actions, while the social component involves the expression of these internal attitudes through communicative acts. To achieve blame-sensitivity in NLG, we need to model both these aspects. In the following sections, we briefly discuss how this could be accomplished.

3.1 Pragmatic and Belief Reasoning

Before a speaker S can determine the high-level perlocutionary effects of an utterance on an addressee (H) vis-á-vis whether or not they feel criticized or blamed, it is first necessary to determine the precise set of beliefs and intentions of the addressee upon hearing an utterance u in context c. We denote this updated set of beliefs and intentions $\Psi_H(u, c)$. Note that this set is a *model* of agent H's beliefs and intentions from the speaker S's perspective, and not necessarily equivalent to the actual belief state of agent H. In order to perform this mental modeling, we utilize a reasoning system similar to that in (Briggs and Scheutz, 2011). This pragmatic reasoning architecture utilizes a set of rules of the form:

$$[[U]]_C := \phi_1 \wedge ... \wedge \phi_n$$

where U denotes an utterance form, C denotes a set of contextual constraints that must hold, and ϕ denotes a belief update predicate. An utterance form is specified by $u = UtteranceType(\alpha, \beta, X, M)$, where $UtteranceType$ denotes the dialogue turn type (e.g. statement, y/n-question), α denotes the speaker of the utterance u, β denotes the addressee of the utterance, X denotes the surface semantics of the utterance, and M denotes a set of sentential modifiers. An example of such a pragmatic rule is found below:

$$[[Stmt(S, H, X, \{\})]]_\emptyset := want(S, bel(H, X))$$

which denotes that a statement by the speaker S to an addressee H that X holds should indicate that, "S wants H to believe X," in all contexts (given the empty set of contextual constraints). If this rule matches a recognized utterance (and the contextual constraints are satisfied, which is trivial in this case), then the mental model of the addressee is updated such that: $want(S, bel(H, X)) \in \Psi_H(u, c)$.

Of particular interest with regard to the Gupta et al. (2007) results, Briggs and Scheutz (2011) describe how they can use their system to understand the semantics of the adverbial modifier "yet," which they describe as being indicative of mutually understood intentionality. More accurately, "yet," is likely indicative of a belief regarding *expectation* of an action being performed or state being achieved. Therefore, a plausible pragmatic rule to interpret, "X is not done yet," could be:

$$[[Stmt(S, H, \neg done(X), \{yet\})]]_\emptyset := want(S, bel(H, \neg done(X))) \wedge expects(S, done(X))$$

Furthermore, in a cooperative, task-driven context, such as that described in (Gupta et al., 2007), it would not be surprising for an interactant to infer that this expectation is further indicative of a belief in a particular intention or a task-based obligation to achieve X.[3]

As such, if we consider an utterance u_d as being a standard direct request form (strategy 3), and an utterance u_y as being an indirect construction with a yet modifier (strategy 4), the following facts may hold:

$$bel(S, promised(H, S, X, t_p)) \notin \Psi_H(u_d, c)$$

$$bel(S, promised(H, S, X, t_p)) \in \Psi_H(u_y, c)$$

If S is making a request to H, there is no believed agreement to achieve X. However, if "yet," is utilized, this may indicate to H a belief that S thinks there is such an agreement.

Having calculated an updated mental model of the addressee's beliefs after hearing a candidate utterance u, we now can attempt to infer the degree to which u is interpreted as an act of criticism or blame.

3.2 Blame Modeling

Attributions of blame are influenced by several factors including, but not limited to, beliefs about an agent's intentionality, capacity, foreknowledge, obligations, and possible justifications (Malle et al., 2012). Given the centrality of intentionality in blame attribution, it is unsurprising that current computational models involve reasoning within a symbolic BDI (belief, desire, intention) framework, utilizing rules to infer an ordinal degree of blame based on the precise set of facts regarding these factors (Mao and Gratch, 2012; Tomai and Forbus, 2007). A rule that is similar to those found in these systems is:

$$bel(S, promised(H, S, X, t_p)) \wedge bel(S, \neg X) \wedge bel(S, (t > t_p)) \wedge bel(S, capable\_of(H, X)) \Rightarrow blames(S, H, high)$$

that is to say, if agent S believes agent H promised to him or her to achieve X by time t_p, and S believes X has not been achieved and the current time t is past t_p, and S believes H is capable of fulfilling this promise, then S will blame H to a high degree. Continuing our discussion regarding the perlocutionary effects of u_d and u_y, it is likely then that: $blames(S, H, high) \notin \Psi_H(u_d, c)$ and $blames(S, H, high) \in \Psi_H(u_y, c)$.

3.3 FTA Modeling

Having determined whether or not an addressee would feel criticized or blamed by a particular candidate utterance, it is then necessary to translate this assessment back into the terms of FTA-degree (the currency of the NLG system). This requires a function $\beta(\Psi)$ that maps the ordinal blame assessment of the speaker toward the hearer based on a set of beliefs Ψ, described in the previous section, to a numerical value than can be utilized to calculate the severity of the FTA (e.g. $blames(S, H, high) = 9.0$, $blames(S, H, medium) = 4.5$). For the purposes of this paper we adopt the theta-notation of Gupta et al. (2007) to denote the degree to which an utterance is a FTA. With the β function, we can then express the blame-related FTA severity of an utterance as:

$$\Theta_{blame}(u, c) = \beta_H(\Psi_H(u, c)) - \alpha(c) \cdot \beta_S(\Psi_S)$$

where β_H denotes the level of blame the speaker believes the hearer has inferred based on the addressee's belief state after hearing utterance u with context c ($\Psi_H(u, c)$). β_S denotes the level of blame the speaker believes is appropriate given his or her current belief state. Finally, $\alpha(c)$ denotes a

[3]How precisely this reasoning is and/or ought to be performed is an important question, but is outside the scope of this paper.

multiplicative factor that models the appropriateness of blame given the current social context. For instance, independent of the objective blameworthiness of a superior, it may be inappropriate for a subordinate to criticize his or her superior in certain contexts.

Finally, then, the degree to which an utterance is a FTA is the sum of all the contributions of evaluations of possible threats to positive face and possible threats to negative face:

$$\Theta(u,c) = \sum_{p \in P} \Theta_p(u,c) + \sum_{n \in N} \Theta_n(u,c)$$

where P denotes the set of all possible threats to positive face (e.g. blame) and N denotes the set of all possible threats to negative face (e.g. directness).

We can see how this would account for the human-subject results from (Gupta et al., 2007), as conventionally indirect requests (strategies 1 and 2) would not produce large threat-value contributions from either the positive or negative FTA components. Direct requests (strategy 3) would, however, potentially produce a large Θ_N contribution, while their set of indirect requests (strategy 4) would trigger a large Θ_P contribution.

4 Discussion

Having presented an approach to avoid certain types of positive-FTAs through reasoning about blame, one may be inclined to ask some questions regarding the justification behind this approach. Why should we want to better model one highly complex social phenomenon (politeness) through the inclusion of a model of another highly complex social phenomenon (blame)? Does the integration of a computational model of blame actually add anything that would justify the effort?

At a superficial level, it does not. The criticism/blame-related threat of a specific speech act can be implicitly factored into the base FTA-degree evaluation function supplied to the system, determined by empirical data or designer-consensus as is the case of (Miller et al., 2008). However, this approach is limited in a couple ways. First, this does not account for the fact that, in addition to the set of social factors Brown and Levinson articulated, the appropriateness of an act of criticism or blame is also dependent on whether or not it is *justified*. Reasoning about whether or not an act of blame is justified requires: a computational model of blame.

Second, the inclusion of blame-reasoning within the larger scope of the entire agent architecture may enable useful behaviors both inside and outside the natural language system. There is a growing community of researchers interested in developing ethical-reasoning capabilities for autonomous agents (Wallach and Allen, 2008), and the ability to reason about blame has been proposed as one key competency for such an ethically-sensitive agent (Bello and Bringsjord, 2013). Not only is there interest in utilizing such mechanisms to influence general action-selection in autonomous agents, but there is also interest in the ability to understand and generate valid explanations and justifications for adopted courses of action in ethically-charged scenarios, which is of direct relevance to the design of NLG architectures.

While our proposed solution tackles threats to positive face that arise due to unduly critical/blame-oriented utterances, there are many different ways of threatening positive face aside from criticism/blame. These include phenomena such as the discussion of inappropriate/sensitive topics or non-cooperative behavior (e.g. purposefully ignoring an interlocutor's dialogue contribution). Indeed, empirical results show that referring to an interlocutor in a dyadic interaction using an impersonal pronoun (e.g. "someone") may constitute another such positive face threat (De Jong et al., 2008). Future work will need to be done to develop mechanisms to address these other possible threats to positive face.

5 Conclusion

Enabling politeness in NLG is a challenging problem that requires the modeling of a host of complex, social psychological factors. In this paper, we discuss ongoing work to integrate a computational model of blame to prevent inappropriate threats to positive face that can account for prior human-subject data. As an ongoing project, future work is needed to further test and evaluate this proposed approach.

Acknowledgments

We would like to thank the reviewers for their helpful feedback. This work was supported by NSF grant #111323.

References

Paul Bello and Selmer Bringsjord. 2013. On how to build a moral machine. *Topoi*, 32(2):251–266.

Gordon Briggs and Matthias Scheutz. 2011. Facilitating mental modeling in collaborative human-robot interaction through adverbial cues. In *Proceedings of the SIGDIAL 2011 Conference*, pages 239–247, Portland, Oregon, June. Association for Computational Linguistics.

Gordon Briggs and Matthias Scheutz. 2013. A hybrid architectural approach to understanding and appropriately generating indirect speech acts. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*.

Penelope Brown and Stephen C. Levinson. 1987. *Politeness: Some universals in language usage*. Cambridge University Press.

Justine Cassell and Timothy Bickmore. 2003. Negotiated collusion: Modeling social language and its relationship effects in intelligent agents. *User Modeling and User-Adapted Interaction*, 13(1-2):89–132.

Markus De Jong, Mariët Theune, and Dennis Hofs. 2008. Politeness and alignment in dialogues with a virtual guide. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 207–214. International Foundation for Autonomous Agents and Multiagent Systems.

Swati Gupta, Marilyn A Walker, and Daniela M Romano. 2007. How rude are you?: Evaluating politeness and affect in interaction. In *Affective Computing and Intelligent Interaction*, pages 203–217. Springer.

Bertram F Malle, Steve Guglielmo, and Andrew E Monroe. 2012. Moral, cognitive, and social: The nature of blame. *Social thinking and interpersonal behavior*, 14:313.

Wenji Mao and Jonathan Gratch. 2012. Modeling social causality and responsibility judgment in multi-agent interactions. *Journal of Artificial Intelligence Research*, 44(1):223–273.

Christopher A Miller, Peggy Wu, and Harry B Funk. 2008. A computational approach to etiquette: Operationalizing brown and levinson's politeness model. *Intelligent Systems, IEEE*, 23(4):28–35.

Megan Strait, Cody Canning, and Matthias Scheutz. 2014. Let me tell you! investigating the effects of robot communication strategies in advice-giving situations based on robot appearance, interaction modality and distance. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 479–486. ACM.

Emmett Tomai and Ken Forbus. 2007. Plenty of blame to go around: a qualitative approach to attribution of moral responsibility. Technical report, DTIC Document.

Cristen Torrey, Susan R Fussell, and Sara Kiesler. 2013. How a robot should give advice. In *Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on*, pages 275–282. IEEE.

Wendell Wallach and Colin Allen. 2008. *Moral machines: Teaching robots right from wrong*. Oxford University Press.

Generating effective referring expressions using charts

Nikos Engonopoulos and **Alexander Koller**
University of Potsdam, Germany
{engonopo|akoller}@uni-potsdam.de

Abstract

We present a novel approach for generating effective referring expressions (REs). We define a synchronous grammar formalism that relates surface strings with the sets of objects they describe through an abstract syntactic structure. The grammars may choose to require or not that REs are distinguishing. We then show how to compute a chart that represents, in finite space, the complete (possibly infinite) set of valid REs for a target object. Finally, we propose a probability model that predicts how the listener will understand the RE, and show how to compute the most effective RE according to this model from the chart.

1 Introduction

The fundamental challenge in the generation of referring expressions (REG) is to compute an RE which is effective, i.e. understood as intended by the listener. Throughout the history of REG, we have approximated this as the problem of generating *distinguishing* REs, i.e. REs that are only satisfied by a unique individual in the domain. This has been an eminently successful approach, as documented e.g. in the overview article of Krahmer and van Deemter (2012) and a variety of recent shared tasks involving RE generation (Gatt and Belz, 2010; Belz et al., 2008; Koller et al., 2010).

Nonetheless, reducing effectiveness to uniqueness is limiting in several ways. First, in complex, real-world scenes it may not be feasible to generate fully distinguishing REs, or these may have to be exceedingly complicated. It is also not necessary to generate distinguishing REs in such situations, because listeners are very capable of taking the discourse and task context into account to resolve even ambiguous REs. Conversely, listeners can misunderstand even a distinguishing RE, so uniqueness is no guarantee for success. We propose instead to define and train a probabilistic RE resolution model $P(a|t)$, which directly captures the probability that the listener will resolve a given RE t to some object a in the domain. An RE t will then be "good enough" if $P(a^*|t)$ is very high for the intended target referent a^*.

Second, in an interactive setting like the GIVE Challenge (Koller et al., 2010), the listener may behave in a way that offers further information on how they resolved the generated RE. Engonopoulos et al. (2013) showed how an initial estimate of the distribution $P(a|t)$ can be continuously updated based on the listener's behavior, and that this can improve a system's ability to detect misunderstandings. It seems hard to achieve this in a principled way without an explicit model of $P(a|t)$.

In this paper, we present an algorithm that generates the RE t that maximizes $P(a^*|t)$, i.e. the RE that has the highest chance to be understood correctly by the listener according to the probabilistic RE resolution model. This is a challenging problem, since the algorithm must identify that RE from a potentially infinite set of valid alternatives. We achieve this by using a *chart-based* algorithm, a standard approach in parsing and realization, which has (to our knowledge) never been used in REG.

We start by defining a synchronous grammar formalism that relates surface strings to their interpretations as sets of objects in a given domain (Section 3). This formalism integrates REG with surface realization, and allows us to specify in the grammar whether REs are required to be distinguishing. We then show how to compute a chart for a given grammar and target referent in Section 4. Section 5 defines a log-linear model for $P(a|t)$, and presents a Viterbi-style algorithm for computing the RE t from the chart that maximizes $P(a^*|t)$. Section 6 concludes by discussing how to apply our algorithm to the state-of-the-art approaches of Krahmer et al. (2003) and Golland et al. (2010), and how to address a particular challenge involving cycles that arises when dealing with probabilistic listener models.

2 Related Work

RE generation is the task of generating a natural-language expression that identifies an object to the listener. Since the beginnings of modern REG (Appelt, 1985; Dale and Reiter, 1995), this problem has been approximated as generating a *distinguishing* description, i.e. one which fits only one object in the domain and not any of the others. This perspective has made it possible to apply search-based (Kelleher and Kruijff, 2006), logic-based (Areces et al., 2008) and graph-based (Krahmer et al., 2003) methods to the problem, and overall has been one of the success stories of NLG.

However, in practice, human speakers frequently *overspecify*, i.e. they include information in an RE beyond what is necessary to make it distinguishing (Wardlow Lane and Ferreira, 2008; Koolen et al., 2011). An NLG system, too, might include redundant information in an RE to make it easier to understand for the user. Conversely, an RE that is produced by a human can often be easily resolved by the listener even if it is ambiguous. Here we present an NLG system that directly uses a probabilistic model of RE resolution, and is capable of generating ambiguous REs if it predicts that the listener will understand them.

Most existing REG algorithms focus on generating distinguishing REs, and then select the one that is *best* according to some criterion, e.g. most human-like (Krahmer et al., 2003; FitzGerald et al., 2013) or most likely to be understood (Garoufi and Koller, 2013). By contrast, Mitchell et al. (2013) describe a stochastic algorithm that computes human-like, non-relational REs that may not be distinguishing. Golland et al. (2010) are close to our proposal in spirit, in that they use a log-linear probability model of RE resolution to compute a possibly non-distinguishing RE. However, they use a trivial REG algorithm which is limited to grammars that only permit a (small) finite set of REs for each referent. This is in contrast to general REG, where there is typically an infinite set of valid REs, especially when relational REs ("the button *to the left of* the plant") are permitted.

Engonopoulos et al. (2013) describe how to update an estimate for $P(a|t)$ based on a log-linear model based on observations of the listener's behavior. They use a shallow model based on a string t and not an RE derived from a grammar, and they do not discuss how to generate the best t. The algorithm we develop here fills this gap.

Our formalism for REG can be seen as a *synchronous* grammar formalism; it simultaneously derives strings and their interpretations, connecting the two by an abstract syntactic representation. This allows performing REG and surface realization with a single algorithm, along the lines of SPUD (Stone et al., 2003) and its planning-based implementation, CRISP (Koller and Stone, 2007). Probabilistic synchronous grammars are widely used in statistical machine translation (Chiang, 2007; Graehl et al., 2008; Jones et al., 2012) and semantic parsing (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007). Lu and Ng (2011) have applied such grammars to surface realization. Konstas and Lapata (2012) use related techniques for content selection and surface realization (with simple, non-recursive grammars).

Charts are standard tools for representing a large space of possible linguistic analyses compactly. Next to their use in parsing, they have also been applied to surface realization (Kay, 1996; Carroll et al., 1999; Kaplan and Wedekind, 2000). To our knowledge, ours is the first work using charts for REG. This is challenging because the input to REG is much less structured than in parsing or realization.

3 Grammars for RE generation

We define a new grammar formalism that we use for REG, which we call *semantically intepreted grammar* (SIG). SIG is a synchronous grammar formalism that relates natural language strings with the sets of objects in a given domain which they describe. It uses *regular tree grammars (RTGs)* to describe languages of *derivation trees*, which then project to strings and sets.

3.1 Derivation trees

We describe the abstract syntax of an RE by its *derivation tree*, which is a tree over some ranked signature Σ of symbols representing lexicon entries and grammatical constructions. A *(ranked) signature* is a finite set of symbols $r \in \Sigma$, each of which is assigned an *arity* $\mathsf{ar}(r) \in \mathbb{N}_0$. A tree over the signature Σ is a term $r(t_1, \dots, t_n)$, where $r \in \Sigma$, $n = \mathsf{ar}(r)$, and $t_1, \dots, t_n$ are trees over Σ. We write T_Σ for the set of all trees over Σ.

Fig. 1b shows an example derivation tree for the RE "the square button" over the signature $\Sigma = \{\mathit{def}|_1, \mathit{square}|_1, \mathit{button}|_0\}$, where $r|_n$ indicates that the symbol r has arity n. In term nota-

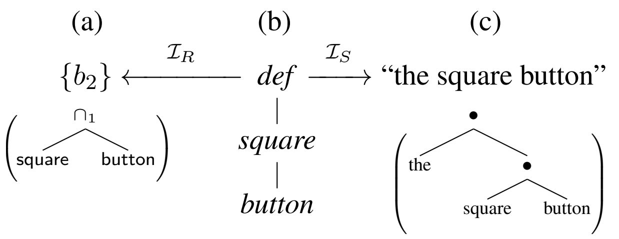


Figure 1: A SIG derivation tree (b) with its interpretations (a, c).

tion, it is *def*(*square*(*button*)).

String interpretation. We *interpret* derivation trees simultaneously as strings and sets. First, let Δ be a finite alphabet, and let Δ^* be the string algebra over Δ. We define a *string interpretation* over Δ as a function $\mathcal{I}_S$ that maps each $r|_n \in \Sigma$ to a function $\mathcal{I}_S(r) : (\Delta^*)^n \to \Delta^*$. For instance, we can assign string interpretations to our example signature Σ as follows; we write $w_1 \bullet w_2$ for the concatenation of the strings w_1 and w_2.

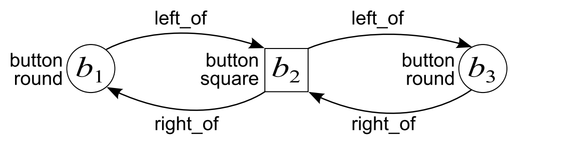
$$\begin{array}{rcl} \mathcal{I}_S(\mathit{def})(w_1) & = & \text{the} \bullet w_1 \\ \mathcal{I}_S(\mathit{square})(w_1) & = & \text{square} \bullet w_1 \\ \mathcal{I}_S(\mathit{button}) & = & \text{button} \end{array}$$

Since the arity of $\mathcal{I}_S(r)$ is the same as the arity of r for any $r \in \Sigma$, we can use $\mathcal{I}_S$ to recursively map derivation trees to strings. Starting at the leaves, we map the tree $r(t_1, \ldots, t_n)$ to the string $\mathcal{I}_S(r)(\mathcal{I}_S(t_1), \ldots, \mathcal{I}_S(t_n))$, where $\mathcal{I}_S(t_i)$ is the string that results from recursively applying $\mathcal{I}_S$ to the subtree t_i. In the example, the subtree *button* is mapped to the string "button". We then get the string for the subtree *square*(*button*) by concatenating this with "square", obtaining the string "square button" and so on, as shown in Fig. 1c.

Relational interpretation. We further define a *relational interpretation* $\mathcal{I}_R$, which maps each $r|_n \in \Sigma$ to a function $\mathcal{I}_R(r) : R(U)^n \to R(U)$, where $R(U)$ is a class of *relations*. We define $\mathcal{I}_R$ over some first-order *model* structure $M = \langle U, L \rangle$, where U is a finite *universe* U of individuals and L interprets a finite set of predicate symbols as relations over U. We let $R(U)$ be the set of all k-place relations over U for all $k \geq 0$. The subsets of U are the special case of $k = 1$. We write $k(R)$ for the arity of a relation $R \in R(U)$.

For the purposes of this paper, we construct $\mathcal{I}_R$ by combining the following operations:

- The denotations of the atomic predicate symbols of M; see Fig. 2 for an example.



$U = \{b_1, b_2, b_3\}$ $\qquad$ $\mathsf{button} = \{b_1, b_2, b_3\}$
$\mathsf{round} = \{b_1, b_3\}$ $\qquad$ $\mathsf{square} = \{b_2\}$
$\mathsf{left\_of} = \{\langle b_1, b_2\rangle, \langle b_2, b_3\rangle\}$
$\mathsf{right\_of} = \{\langle b_2, b_1\rangle, \langle b_3, b_2\rangle\}$

Figure 2: A simple model, illustrated as a graph.

- $\mathsf{proj}_i(R) = \{a_i \mid \langle a_1, \ldots, a_{k(R)}\rangle \in R\}$ is the projection to the i-th component; if $i > k(R)$, it evaluates to $\emptyset$.
- $R_1 \cap_i R_2 = \{\langle a_1, \ldots, a_{k(R_1)}\rangle \in R_1 \mid a_i \in R_2\}$ is the intersection on the i-th component of R_1; if $i > k(R_1)$, it evaluates to $\emptyset$.
- For any $a \in U$, $\mathsf{uniq}_a(R)$ evaluates to $\{a\}$ if $R = \{a\}$, and to $\emptyset$ otherwise.
- For any $a \in U$, $\mathsf{member}_a(R)$ evaluates to $\{a\}$ if $a \in R$, and to $\emptyset$ otherwise.

For the example, we assume that we want to generate REs over the scene shown in Fig. 2; it consists of the universe $U = \{b_1, b_2, b_3\}$ and interprets the atomic predicate symbols button, square, round, left_of, and right_of. Given this, we can assign a relational interpretation to the derivation tree in Fig. 1b using the following mappings:

$$\begin{array}{rcl} \mathcal{I}_R(\mathit{def})(R_1) & = & R_1 \\ \mathcal{I}_R(\mathit{square})(R_1) & = & \mathsf{square} \cap_1 R_1 \\ \mathcal{I}_R(\mathit{button}) & = & \mathsf{button} \end{array}$$

We evaluate a derivation tree to a relation as we did for strings (cf. Fig. 1a). The subtree *button* maps to the denotation of the symbol button, i.e. $\{b_1, b_2, b_3\}$. The subtree *square*(*button*) evaluates to the intersection of this set with the set of square individuals, i.e. $\{b_2\}$; this is also the relational interpretation of the entire derivation tree. We thus see that "the square button" is an RE that describes the individual b_2 uniquely.

3.2 Semantically interpreted grammars

Now we define grammars that describe relations between strings and relations over U. We achieve this by combining a *regular tree grammar* (RTG, (Gécseg and Steinby, 1997; Comon et al., 2007)), describing a language of derivation trees, with a string interpretation and a relational interpretation. An RTG $G = (N, \Sigma, S, P)$ consists of a finite set N of nonterminal symbols, a ranked signature Σ, a start symbol $S \in N$, and a finite set P of production rules $\mathrm{A} \to r(B_1, ..., B_n)$, where

$A, B_1, \ldots, B_n \in N$ and $r|_n \in \Sigma$. We say that a tree $t_2 \in T_\Sigma$ can be *derived in one step* from $t_1 \in T_\Sigma$, $t_1 \Rightarrow t_2$, if it can be obtained by replacing an occurrence of B in t_1 with t and P contains the rule $B \to t$. A tree $t_n \in T_\Sigma$ can be *derived* from t_1, $t_1 \Rightarrow^* t_n$, if there is a sequence $t_1 \Rightarrow \ldots \Rightarrow t_n$ of length $n \geq 0$. For any nonterminal A, we write $L_A(G)$ for the set of trees $t \in T_\Sigma$ with $A \Rightarrow^* t$. We simply write $L(G)$ for $L_S(G)$ and call it the *language* of G.

We define a *semantically interpreted grammar* (SIG) as a triple $\mathcal{G} = (G, \mathcal{I}_S, \mathcal{I}_R)$ of an RTG G over some signature Σ, together with a string interpretation $\mathcal{I}_S$ over some alphabet Δ and a relational interpretation $\mathcal{I}_R$ over some universe U, both of which interpret the symbols in Σ. We assume that every terminal symbol $r \in \Sigma$ occurs in at most one rule, and that the nonterminals of G are pairs A_b of a syntactic category A and a *semantic index* $b = \mathsf{ix}(A_b)$. A semantic index indicates the individual in U to which a given constituent is meant to refer, see e.g. (Kay, 1996; Stone et al., 2003). Note that SIGs can be seen as specific Interpreted Regular Tree Grammars (Koller and Kuhlmann, 2011) with a set and a string interpretation.

We ignore the start symbol of G. Instead, we say that given some individual $b \in U$ and syntactic category A, the set of *referring expressions* for b is $\mathsf{RE}_\mathcal{G}(A, b) = \{t \in L_{A_b}(G) \mid \mathcal{I}_R(t) = \{b\}\}$, i.e. we define an RE as a derivation tree that G can derive from A_b and whose relational interpretation is $\{b\}$. From t, we can read off the string $\mathcal{I}_S(t)$.[1]

3.3 An example grammar

Consider the SIG $\mathcal{G}$ in Fig. 3 for example. The grammar is written in template form. Each rule is instantiated for all semantic indices specified in the line above; e.g. the symbol round denotes the set $\{b_1, b_3\}$, therefore there are rules $N_{b_1} \to \mathit{round}_{b_1}(N_{b_1})$ and $N_{b_3} \to \mathit{round}_{b_3}(N_{b_3})$. The values of $\mathcal{I}_R$ and $\mathcal{I}_S$ for each symbol are specified below the RTG rule for that symbol.

We can use $\mathcal{G}$ to generate NPs that refer to the target referent b_2 given the model shown in Fig. 2 by finding trees in $L_{\mathrm{NP}_{b_2}}(G)$ that refer to $\{b_2\}$. One such tree is $t_1 = \mathit{def}_{b_2}(\mathit{square}_{b_2}(\mathit{button}_{b_2}))$, a more detailed version of the tree in Fig. 1b. It can be derived by $\mathrm{NP}_{b_2} \Rightarrow \mathit{def}_{b_2}(\mathrm{N}_{b_2}) \Rightarrow \mathit{def}_{b_2}(\mathit{square}_{b_2}(\mathrm{N}_{b_2})) \Rightarrow t_1$. Because $\mathcal{I}_R(t_1) = \{b_2\}$, we see that $t_1 \in \mathsf{RE}_\mathcal{G}(\mathrm{NP}, b_2)$; it represents the string $\mathcal{I}_S(t_1) =$ "the square button".

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

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

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

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

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

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

Figure 3: An example SIG grammar.

A second derivation tree for b_2 is $t_2 = \mathit{def}_{b_2}(\mathit{square}_{b_2}(\mathit{square}_{b_2}(\mathit{button}_{b_2})))$, corresponding to $\mathcal{I}_S(t_2) =$ "the square square button". It derives from NP_{b_2} in four steps, and has $\mathcal{I}_R(t_2) = \{b_2\}$. Even the small grammar $\mathcal{G}$ licences an infinite set of REs for b_2, all of which are semantically correct. Avoiding the generation of nonsensical REs like "the square square button" is a technical challenge to which we will return in Section 6. $\mathcal{G}$ can also derive relational REs; for instance, the derivation tree in Fig. 6 for the string "the button to the left of the square button" is in $\mathsf{RE}_\mathcal{G}(\mathrm{NP}, b_1)$.

Finally, $\mathcal{G}$ considers the non-distinguishing $t_3 = \mathit{def}_{b_2}(\mathit{button}_{b_2})$ (for "the button") a valid RE for b_2. This is because member_{b_2} will quietly project the set $\{b_1, b_2, b_3\}$ (to which button_{b_2} refers) to $\{b_2\}$. As discussed in previous sections, we want to allow such non-unique REs and delegate the judgment about their quality to the probability model. It would still be straightforward, however, to impose a hard uniqueness constraint, by simply changing $\mathcal{I}_R(\mathit{def}_a)(R_1)$ to $\mathsf{uniq}_a(R_1)$ in Fig. 3. This would yield $\mathcal{I}_R(t_3) = \emptyset$, i.e. t_3 would no longer be in $\mathsf{RE}_\mathcal{G}(\mathrm{NP}, b_2)$.

4 Chart-based RE generation

We now present a chart-based algorithm for generating REs with SIG grammars. Charts allow us to represent all REs for a target referent compactly, and can be computed efficiently. We show in Section 5 that charts also lend themselves well to computing the most effective RE.

[1]Below, we will often write the RE as a string when the derivation tree is clear.

$$\begin{array}{l}
\mathrm{N}_{b_1}/\{b_1,b_2,b_3\} \to \mathit{button}_{b_1} \\
\mathrm{N}_{b_2}/\{b_1,b_2,b_3\} \to \mathit{button}_{b_2} \\
\mathrm{N}_{b_3}/\{b_1,b_2,b_3\} \to \mathit{button}_{b_3} \\
\mathrm{N}_{b_1}/\{b_1,b_3\} \to \mathit{round}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_2,b_3\}) \\
\mathrm{N}_{b_3}/\{b_1,b_3\} \to \mathit{round}_{b_3}(\mathrm{N}_{b_3}/\{b_1,b_2,b_3\}) \\
\mathrm{N}_{b_1}/\{b_1,b_3\} \to \mathit{round}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_3\}) \\
\mathrm{N}_{b_3}/\{b_1,b_3\} \to \mathit{round}_{b_3}(\mathrm{N}_{b_3}/\{b_1,b_3\}) \\
\mathrm{N}_{b_2}/\{b_2\} \to \mathit{square}_{b_2}(\mathrm{N}_{b_2}/\{b_1,b_2,b_3\}) \\
\mathrm{N}_{b_2}/\{b_2\} \to \mathit{square}_{b_2}(\mathrm{N}_{b_2}/\{b_2\}) \\
\mathrm{NP}_{b_2}/\{b_2\} \to \mathit{def}_{b_2}(\mathrm{N}_{b_2}/\{b_1,b_2,b_3\}) \\
\mathrm{NP}_{b_2}/\{b_2\} \to \mathit{def}_{b_2}(\mathrm{N}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_1}/\{b_1\} \to \mathit{leftof}_{b_1,b_2}(\mathrm{N}_{b_1}/\{b_1,b_2,b_3\},\mathrm{NP}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_1}/\{b_1\} \to \mathit{leftof}_{b_1,b_2}(\mathrm{N}_{b_1}/\{b_1,b_3\},\mathrm{NP}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_1}/\{b_1\} \to \mathit{leftof}_{b_1,b_2}(\mathrm{N}_{b_1}/\{b_1\},\mathrm{NP}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_1}/\{b_1\} \to \mathit{round}_{b_1}(\mathrm{N}_{b_1}/\{b_1\}) \\
\mathrm{NP}_{b_1}/\{b_1\} \to \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_2,b_3\}) \\
\mathrm{NP}_{b_1}/\{b_1\} \to \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1,b_3\}) \\
\mathrm{NP}_{b_1}/\{b_1\} \to \mathit{def}_{b_1}(\mathrm{N}_{b_1}/\{b_1\}) \\
\mathrm{N}_{b_3}/\{b_3\} \to \mathit{rightof}_{b_3,b_2}(\mathrm{N}_{b_3}/\{b_1,b_2,b_3\},\mathrm{NP}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_3}/\{b_3\} \to \mathit{rightof}_{b_3,b_2}(\mathrm{N}_{b_3}/\{b_1,b_3\},\mathrm{NP}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_3}/\{b_3\} \to \mathit{rightof}_{b_3,b_2}(\mathrm{N}_{b_3}/\{b_3\},\mathrm{NP}_{b_2}/\{b_2\}) \\
\mathrm{N}_{b_3}/\{b_3\} \to \mathit{round}_{b_3}(\mathrm{N}_{b_3}/\{b_3\}) \\
\mathrm{NP}_{b_3}/\{b_3\} \to \mathit{def}_{b_3}(\mathrm{N}_{b_3}/\{b_1,b_2,b_3\}) \\
\mathrm{NP}_{b_3}/\{b_3\} \to \mathit{def}_{b_3}(\mathrm{N}_{b_3}/\{b_1,b_3\}) \\
\mathrm{NP}_{b_3}/\{b_3\} \to \mathit{def}_{b_3}(\mathrm{N}_{b_3}/\{b_3\}) \\
\mathrm{N}_{b_2}/\{b_2\} \to \mathit{leftof}_{b_2,b_3}(\mathrm{N}_{b_2}/\{b_1,b_2,b_3\},\mathrm{NP}_{b_3}/\{b_3\}) \\
\mathrm{N}_{b_2}/\{b_2\} \to \mathit{rightof}_{b_2,b_1}(\mathrm{N}_{b_2}/\{b_1,b_2,b_3\},\mathrm{NP}_{b_1}/\{b_1\}) \\
\mathrm{N}_{b_2}/\{b_2\} \to \mathit{leftof}_{b_2,b_3}(\mathrm{N}_{b_2}/\{b_2\},\mathrm{NP}_{b_3}/\{b_3\}) \\
\mathrm{N}_{b_2}/\{b_2\} \to \mathit{rightof}_{b_2,b_1}(\mathrm{N}_{b_2}/\{b_2\},\mathrm{NP}_{b_1}/\{b_1\})
\end{array}$$

Figure 4: The chart for the grammar in Fig. 3.

4.1 RE generation charts

Generally speaking, a chart is a packed data structure which describes how larger syntactic representations can be recursively built from smaller ones. In applications such as parsing and surface realization, the creation of a chart is driven by the idea that we consume some input (words or semantic atoms) as we build up larger structures. The parallel to this intuition in REG is that "larger" chart entries are more precise descriptions of the target, which is a weaker constraint than input consumption. Nonetheless, we can define REG charts whose entries are packed representations for large sets of possible REs, and compute them in terms of these entries instead of RE sets.

Technically, we represent charts as RTGs over an extended set of nonterminals. A chart for generating an RE of syntactic category A for an individual $b \in U$ is an RTG $C = (N', \Sigma, S', P')$, where $N' \subseteq N \times R(U)$ and $S' = A_b/\{b\}$. Intuitively, the nonterminal $A_b/\{a_1, \ldots, a_n\}$ expresses that we *intend* to generate an RE for b from A, but each RE that we can derive from the nonterminal *actually* denotes the referent set $\{a_1, \ldots, a_n\}$.

A chart for the grammar in Fig. 3 is shown in Fig. 4. To generate an NP for b_2, we let its start symbol be $S' = \mathrm{NP}_{b_2}/\{b_2\}$. The rule $\mathrm{N}_{b_2}/\{b_1, b_2, b_3\} \to \mathit{button}_{b_2}$ says that we can generate an RE t with $\mathcal{I}_R(t) = \{b_1, b_2, b_3\}$ from the nonterminal symbol N_{b_2} by expanding this symbol with the grammar rule $\mathrm{N}_{b_2} \to \mathit{button}_{b_2}$. Similarly,

$$\frac{\begin{array}{c} A \to r(B_1, ..., B_n) \text{ in } G \\ B'_1 = B_1/R_1, ..., B'_n = B_n/R_n \text{ in } N' \end{array}}{\begin{array}{c} \text{Add } A' = A/\mathcal{I}_R(r)(R_1, ..., R_n) \text{ to } N' \\ \text{Add rule } A' \to r(B'_1, ..., B'_n) \text{ to } P' \end{array}}$$

Figure 5: The chart computation algorithm.

the rule $\mathrm{N}_{b_2}/\{b_2\} \to \mathit{square}_{b_2}(\mathrm{N}_{b_2}/\{b_1, b_2, b_3\})$ expresses that we can generate an RE with $\mathcal{I}_R(t) = \{b_2\}$ by expanding the nonterminal symbol N_{b_2} into $\mathit{square}_{b_2}(t')$, where t' is any tree that the chart can generate from $\mathrm{N}_{b_2}/\{b_1, b_2, b_3\}$.

4.2 Computing a chart

Given a SIG $\mathcal{G}$, a syntactic category A, and a target referent b, we can compute a chart C for $\mathsf{RE}_{\mathcal{G}}(A, b)$ using the parsing schema in Fig. 5. The schema assumes that we have a rule $A \to r(B_1, \ldots, B_n)$ in G; in addition, for each $1 \leq i \leq n$ it assumes that we have already added the nonterminal $B'_i = \mathrm{B}_i/R_i$ to the chart, indicating that there is a tree t_i with $B_i \Rightarrow^* t_i$ and $\mathcal{I}_R(t_i) = R_i$. Then we know that $t = r(t_1, \ldots, t_n)$ can be derived from A and that $R' = \mathcal{I}_R(t) = \mathcal{I}_R(r)(R_1, \ldots, R_n)$. We can therefore add the nonterminal $A' = A/R'$ and the production rule $A' \to r(B'_1, \ldots, B'_n)$ to the chart; this rule can be used as the first step in a derivation of t from A'. We can optimize the algorithm by adding A' and the rule only if $R' \neq \emptyset$.

The algorithm terminates when it can add no more rules to the chart. Because U is finite, this always happens after a finite number of steps, even if there is an infinite set of REs. For instance, the chart in Fig. 4 describes an infinite language of REs, including "the square button", "the button to the left of the round button", "the button to the left of the button to the right of the square button", etc. Thus it represents relational REs that are nested arbitrarily deeply through a finite number of rules.

After termination, the chart contains all rules by which a nonterminal can be decomposed into other (productive) nonterminals. As a result, $L(C)$ contains exactly the REs for b of category A:

Theorem 1 *If C is a chart for the SIG $\mathcal{G}$, the syntactic category A, and the target referent b, then $L(C) = \mathsf{RE}_{\mathcal{G}}(A, b)$.*

5 Computing best referring expressions

The chart algorithm allows us to compactly represent *all* REs for the target referent. We now show how to compute the *best* RE from the chart. We present a novel probability model $P(b|t)$ for RE resolution, and take the "best" RE to be the

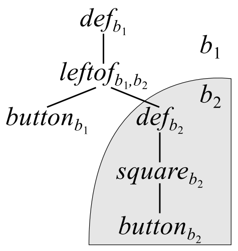


Figure 6: The derivation tree for "the button to the left of the square button".

one with the highest chance to be understood as intended. Next to the best RE itself, the algorithm also computes the entire distribution $P(b|t)$, to support later updates in an interactive setting.

Nothing in our algorithm hinges on this particular model; it can also be used with any other scoring model that satisfies a certain monotonicity condition which we spell out in Section 5.2.

5.1 A log-linear model for effective REs

We model the probability $P(b|t)$ that the listener will resolve the RE t to the object b using a log-linear model with a set of *feature functions* $f(a, t, M)$, where a is an object, t is a derivation tree, and M is the relational interpretation model.

We focus on features that only look at information that is local to a specific subtree of the RE, such as the label at the root. For instance, a feature $f_{round}(a, t', M)$ might return 1 if the root label of t' is *round*$_a$ and a is round in M, and 0 otherwise. Another feature $f_{def}(a, t', M)$ might return $1/k$ if t' is of the form $def_b(t'')$, $R = \mathcal{I}_R(t'')$ has k elements, and $a \in R$; and 0 otherwise. This feature counterbalances the ability of the grammar in Fig. 3 to say "the w" even when w is a non-unique description by penalizing descriptions with many possible referents through lower feature values.

When generating a relational RE, the derivation tree naturally splits into separate *regions*, each of which is meant to identify a specific object. These regions are distinguished by the semantic indices in the nonterminals that derive them; e.g., in Fig. 6, the subtree for "the square button" is an attempt to refer to b_2, whereas the RE as a whole is meant to refer to b_1. To find out how effective the RE is as a description of b_1, we evaluate the features at all nodes in the region $\mathsf{top}(t)$ containing the root of t.

Each feature function f_i is associated with a *weight* w_i. We obtain a *score tuple* $\mathsf{sc}(t')$ for some subtree t' of an RE as follows:

$$\mathsf{sc}(t') = \langle s(a_1, t', M), \ldots, s(a_m, t', M) \rangle,$$

t	b_1	b_2	b_3
"the button"	0.33	0.33	0.33
"the round button"	0.45	0.10	0.45
"the button to the left of the square button"	0.74	0.14	0.12

Figure 7: Probability distributions for some REs t.

where $U = \{a_1, \ldots, a_m\}$ and $s(a, t', M) = \sum_{i=1}^{n} w_i \cdot f_i(a, t', M)$. We then combine these into a score tuple $\mathsf{score}(t) = \sum_{u \in \mathsf{top}(t)} \mathsf{sc}(t.u)$ for the whole RE t, where $t.u$ is the subtree of t below the node u. Finally, given a score tuple $\mathsf{s} = \langle s_1, \ldots, s_m \rangle$ for t, we define the usual log-linear probability distribution as

$$P(a_i|t) = \mathsf{prob}(a_i, \mathsf{s}) = \frac{e^{s_i}}{\sum_{j=1}^{m} e^{s_j}}.$$

The *best* RE for the target referent b is then

$$\mathsf{best}_{\mathcal{G}}(A, b) = \underset{t \in \mathsf{RE}_{\mathcal{G}}(A,b)}{\arg\max}\ \mathsf{prob}(b, \mathsf{sc}(t)).$$

For illustration, we consider a number of REs for b_1 in our running example. We use f_{round} and f_{def} and let $w_{round} = w_{def} = 1$. In this case, the RE "the button" has a score tuple $\langle 1/3, 1/3, 1/3 \rangle$, which is the sum of the tuple $\langle 0, 0, 0 \rangle$ for f_{round} (since the RE does not use the "round" rule) and the tuple $\langle 1/3, 1/3, 1/3 \rangle$ for f_{def} (since "button" is three-way ambiguous in M). This yields a uniform probability distribution over U (see Fig. 7). By contrast, "the round button" gets $\langle 3/2, 0, 3/2 \rangle$, resulting in the distribution in the second line of Fig. 7. This RE is judged better than "the button" because it assigns a higher probability to b_1.

Relational REs involve derivation trees with multiple regions, only the top one of which is directly counted for $P(b|t)$ (see Fig. 6). We incorporate the quality of the other regions through appropriate features. In the example, we use a feature $f_{leftof}(a, t', M) = \sum_{b: \langle a,b \rangle \in \mathsf{left\_of}} P(b|t'')$, where t'' is the second subtree of t'. This feature computes the probability that the referent to which the listener resolves t'' is actually to the right of a, and will thus take a high value if t'' is a good RE for b_2. Assuming a probability distribution of $P(b_2|t') = 0.78$ and $P(b_1|t') = P(b_3|t') = 0.11$ for t' ="the square button", we get the tuple $\langle 0.78, 0.11, 0 \rangle$ for f_{leftof}, yielding the third line of Fig. 7 for $w_{leftof} = 1$.

5.2 Computing the best RE

We compute $\mathsf{best}_{\mathcal{G}}(A, b)$ from the chart by adapting the Viterbi algorithm. Our key data structure assigns a score tuple $\mathbf{is}(A')$ to each nonterminal A' in the chart. Intuitively, if the semantic index of A' is b, then $\mathbf{is}(A')$ is the score tuple $\mathsf{sc}(t)$ for the tree $t \in L_{A'}(C)$ which maximizes $P(b|t)$. We also record this best tree as $\mathsf{bt}(A')$. Thus the algorithm is correct if, after running it, we obtain $\mathsf{best}_{\mathcal{G}}(A, b) = \mathsf{bt}(\mathrm{A}_b/\{b\})$.

As is standard in chart algorithms, we limit our attention to features whose values can be computed bottom-up by local operations. Specifically, we assume that if $A' \to r(B'_1, \dots, B'_n)$ is a rule in the chart and t_i is the best RE for B'_i for all i, then the best RE for A' that can be built using this rule is $r(t_1, \dots, t_n)$. This means that features must be *monotonic*, i.e. that the RE that seemed locally best for B'_i leads to the best RE overall.

Under this assumption, we can compute $\mathbf{is}(A')$ and $\mathsf{bt}(A')$ bottom-up as shown in Fig. 8. We iterate over all nonterminals A' in the chart in a fixed linear order, which we call the *evaluation order*. Then we compute $\mathbf{is}(A')$ and $\mathsf{bt}(A')$ by maximizing over the rules for A'. Assume that the best RE for A' can be constructed using the rule $A' \to r(B'_1, \dots, B'_n)$. Then if, at the time we evaluate A', we have fully evaluated all the B'_i in the sense that $\mathsf{bt}(B'_i)$ is actually the best RE for B'_i, the algorithm will assign the best RE for A' to $\mathsf{bt}(A')$, and its score tuple to $\mathbf{is}(A')$. Thus, if we call an evaluation order *exact* if the nonterminals on the right-hand side of each rule in the chart come before the nonterminal on the left-hand side, we can inductively prove the following theorem:

Theorem 2 *If the evaluation order is exact, then for every nonterminal A' in the chart, we obtain* $\mathsf{bt}(A') = \arg\max_{t \in L_{A'}(C)} P(\mathsf{ix}(A')|t)$ *and* $\mathbf{is}(A') = \mathsf{sc}(\mathsf{bt}(A'))$.

In other words, the algorithm is correct if the evaluation order is exact. If it is not, we might compute a sub-optimal RE as $\mathsf{bt}(A')$, which underestimates $\mathbf{is}(A')$. The choice of evaluation order is thus crucial.

```
1: for nonterminals $A'$ in evaluation order do
2:     for rules r of the form $A' \to r(B'_1, \dots, B'_n)$ do
3:         $a = \mathsf{ix}(A')$
4:         $t' = r(\mathsf{bt}(B'_1), \dots, \mathsf{bt}(B'_n))$
5:         $\mathsf{s} = \mathsf{sc}(t') + \sum_{i=1, \mathsf{ix}(B'_i)=a}^{n} \mathbf{is}(B'_i)$
6:         if $\mathsf{prob}(a, \mathsf{s}) > \mathsf{prob}(a, \mathbf{is}(A'))$ then
7:             $\mathbf{is}(A') = \mathsf{s}$
8:             $\mathsf{bt}(A') = t'$
```

Figure 8: Computing the best RE.

6 Evaluating charts with cycles

It remains to show how we can determine an exact evaluation order for a given chart. One way to think about the problem is to consider the *ordering graph* $O(C)$ of the chart C (see Fig. 9 for an example). This is a directed graph whose nodes are the nonterminals of the chart; for each rule $A' \to r(B'_1, \dots, B'_n)$ in C, it has an edge from B'_i to A' for each i. If this graph is acyclic, we can simply compute a topological sort of $O(C)$ to bring the nodes into a linear order in which each B'_i precedes A'. This is enough to evaluate charts using certain simpler models. For instance, we can apply our REG algorithm to the log-linear model of Golland et al. (2010). Because they only generate REs with a bounded number of relations, their grammars effectively only describe finite languages. In such a case, our charts are always acyclic, and therefore a topological sort of $O(C)$ yields an exact evaluation order.

This simple approach will not work with grammars that allow arbitrary recursion, as they can lead to charts with cycles (indicating an infinite set of valid REs). E.g. the chart in Fig. 4 contains a rule $\mathrm{N}_{b_2}/\{b_2\} \to \mathit{square}_{b_2}(\mathrm{N}_{b_2}/\{b_2\})$ (shown in Fig. 9), which can be used to construct the RE t' = "the square square button" in addition to the RE t = "the square button". Such cycles can be *increasing* with respect to a log-linear probability model, i.e. the model considers t' a better RE than t. Indeed, t has a score tuple of $\langle 0, 2, 0 \rangle$, giving $P(b_2|t) = 0.78$. By contrast, t' has a score tuple of $\langle 0, 3, 0 \rangle$, thus $P(b_2|t') = 0.91$. This can be continued indefinitely, with each addition of "square" increasing the probability of being resolved to b_2. Thus, there is no best RE for b_2; every RE can be improved by adding another copy of "square".

In such a situation, it is a challenge to even compute *any* score for every nonterminal without running into infinite loops. We can achieve this by decomposing $O(C)$ into its strongly connected components (SCCs), i.e. the maximal subgraphs in which each node is reachable from any other node. We then consider the *component graph* $O'(C)$; its nodes are the SCCs of $O(C)$, and it has an edge from c_1 to c_2 if $O(C)$ has an edge from some node in c_1 to some node in c_2. $O'(C)$ is acyclic by construction, so we can compute a topological

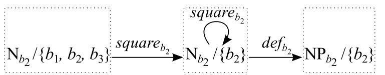


Figure 9: A fragment of the ordering graph for the chart in Fig. 4. Dotted boxes mark SCCs.

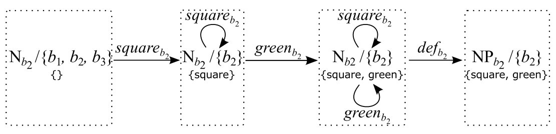


Figure 10: A fragment of a chart ordering graph for a grammar with enriched nonterminals.

sort and order all nonterminals from earlier SCCs before all nonterminals from later SCCs. Within each SCC, we order the nonterminals in the order in which they were discovered by the algorithm in Fig. 5. This yields a linear order on nonterminals, which at least ensures that by the time we evaluate a nonterminal A', there is at least one rule for A' whose right-hand nonterminals have all been evaluated; so $\mathbf{is}(A')$ gets at least *some* value.

In our example, we obtain the order $\mathrm{N}_{b_2}/\{b_1, b_2, b_3\}$, $\mathrm{N}_{b_2}/\{b_2\}$, $\mathrm{NP}_{b_2}/\{b_2\}$. The rule $\mathrm{N}_{b_2}/\{b_2\} \rightarrow$ *square*$_{b_2}(\mathrm{N}_{b_2}/\{b_2\})$ will thus not be considered in the evaluation of $\mathrm{N}_{b_2}/\{b_2\}$, and the algorithm returns "the square button". The algorithm computes optimal REs for acyclic charts, and also for charts where all cycles are *decreasing*, i.e. using the rules in the cycle make the RE worse. This enables us, for instance, to encode the REG problem of Krahmer et al. (2003) into ours by using a feature that evaluates the rule for each attribute to its (negative) cost according to the Krahmer model. Krahmer et al. assume that every attribute has positive cost, and is only used if it is necessary to make the RE distinguishing. Thus all cycles in the chart are decreasing.

One limitation of the algorithm is that it does not overspecify. Suppose that we extend the example model in Fig. 2 with a color predicate green $= \{b_2\}$. We might then want to prefer "the green square button" over "the square button" because it is easier to understand. But since all square objects (i.e. $\{b_2\}$) are also green, using "green" does not change the denotation of the RE, i.e. it is represented by a loop from $\mathrm{N}_{b_2}/\{b_2\}$ to $\mathrm{N}_{b_2}/\{b_2\}$, which is skipped by the algorithm. One idea could be to *break* such cycles by the careful use of a richer set of nonterminals in the grammar; e.g., they might record the set of all attributes that were used in the RE. Our example rule would then become $\mathrm{N}_{b_2}/\{b_2\}/\{square, green\} \rightarrow$ *green*$_{b_2}(\mathrm{N}_{b_2}/\{b_2\}/\{square\})$, which the algorithm can make use of (see Fig. 10).

7 Conclusion

We have shown how to generate REs using charts. Based on an algorithm for computing a chart of all valid REs, we showed how to compute the RE that maximizes the probability of being understood as the target referent. Our algorithm integrates REG with surface realization. It generates distinguishing REs if this is specified in the grammar; otherwise, it computes the best RE without regard to uniqueness, using features that prefer unambiguous REs as part of the probability model.

Our algorithm can be applied to earlier models of REG, and in these cases is guaranteed to compute optimal REs. The probability model we introduced here is more powerful, and may not admit "best" REs. We have shown how the algorithm can still do something reasonable in such cases, but this point deserves attention in future research, especially with respect to overspecification.

We evaluated the performance of our chart algorithm on a number of randomly sampled input scenes from the GIVE Challenge, which contained 24 objects on average. Our implementation is based on the IRTG tool available at `irtg.googlecode.com`. While in the worst case the chart computation is exponential in the input size, in practice runtimes did not exceed 60 ms for the grammar shown in Fig. 3.

We have focused here on *computing* best REs given a probability model. We have left *training* the model and evaluating it on real-world data for future work. Because our probability model focuses on effectiveness for the listener, rather than human-likeness, our immediate next step is to train it on an interaction corpus which records the reactions of human listeners to system-generated REs. A further avenue of research is to deliberately generate succinct but ambiguous REs when the model predicts them to be easily understood. We will explore ways of achieving this by combining the effectiveness model presented here with a language model that prefers succinct REs.

Acknowledgments. We thank Emiel Krahmer, Stephan Oepen, Konstantina Garoufi, Martín Villalba and the anonymous reviewers for their useful comments and discussions. The authors were supported by the SFB 632 "Information Structure".

References

Douglas E. Appelt. 1985. *Planning English sentences*. Cambridge University Press.

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

Anja Belz, Eric Kow, Jette Viethen, and Albert Gatt. 2008. The GREC challenge 2008: Overview and evaluation results. In *Proceedings of the 5th International Conference on Natural Language Generation (INLG)*.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 2007. Tree automata techniques and applications. Available on `http://tata.gforge.inria.fr/`.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean Maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Nikos Engonopoulos, Martin Villalba, Ivan Titov, and Alexander Koller. 2013. Predicting the resolution of referring expressions from user behavior. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Seattle.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Konstantina Garoufi and Alexander Koller. 2013. Generation of effective referring expressions in situated context. *Language and Cognitive Processes*.

Albert Gatt and Anja Belz. 2010. Introducing shared task evaluation to NLG: The TUNA shared task evaluation challenges. In E. Krahmer and M. Theune, editors, *Empirical Methods in Natural Language Generation*, number 5790 in LNCS, pages 264–293. Springer.

Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer-Verlag.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3).

B. Jones, J. Andreas, D. Bauer, K.-M. Hermann, and K. Knight. 2012. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING*.

Ron Kaplan and Jürgen Wedekind. 2000. LFG generation produces context-free languages. In *Proceedings of the 18th COLING*.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th ACL*.

John Kelleher and Geert-Jan Kruijff. 2006. Incremental generation of spatial referring expressions in situated dialogue. In *In Proceedings of Coling-ACL '06, Sydney Australia*.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 2–13. Association for Computational Linguistics.

Alexander Koller and Matthew Stone. 2007. Sentence generation as a planning problem. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*.

Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. The First Challenge on Generating Instructions in Virtual Environments. In E. Krahmer and M. Theune, editors, *Empirical Methods in Natural Language Generation*, number 5790 in LNAI, pages 337–361. Springer.

Yannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th ACL*.

Ruud Koolen, Albert Gatt, Martijn Goudbeek, and Emiel Krahmer. 2011. Factors causing overspecification in definite descriptions. *Journal of Pragmatics*, 43:3231–3250.

Emiel Krahmer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of EMNLP*.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2013. Generating expressions that refer to visible objects. In *Proceedings of NAACL-HLT*, pages 1174–1184.

Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. 2003. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19(4):311–381.

Liane Wardlow Lane and Victor Ferreira. 2008. Speaker-external versus speaker-internal forces on utterance form: Do cognitive demands override threats to referential success? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 34:1466–1481.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th ACL*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*.

Crowdsourcing Language Generation Templates for Dialogue Systems

Margaret Mitchell
Microsoft Research
Redmond, WA USA
memitc@microsoft.com

Dan Bohus
Microsoft Research
Redmond, WA USA
dbohus@microsoft.com

Ece Kamar
Microsoft Research
Redmond, WA USA
eckamar@microsoft.com

Abstract

We explore the use of crowdsourcing to generate natural language in spoken dialogue systems. We introduce a methodology to elicit novel templates from the crowd based on a dialogue seed corpus, and investigate the effect that the amount of surrounding dialogue context has on the generation task. Evaluation is performed both with a crowd and with a system developer to assess the naturalness and suitability of the elicited phrases. Results indicate that the crowd is able to provide reasonable and diverse templates within this methodology. More work is necessary before elicited templates can be automatically plugged into the system.

1 Introduction

A common approach for natural language generation in task-oriented spoken dialogue systems is template-based generation: a set of templates is manually constructed by system developers, and instantiated with slot values at runtime. When the set of templates is limited, frequent interactions with the system can quickly become repetitive, and the naturalness of the interaction is lost.

In this work, we propose and investigate a methodology for developing a corpus of natural language generation templates for a spoken dialogue system via crowdsourcing. We use an existing dialogue system that generates utterances from templates, and explore how well a crowd can generate reliable paraphrases given snippets from the system's original dialogues. By utilizing dialogue data collected from interactions with an existing system, we can begin to learn different ways to converse while controlling the crowd to stay within the scope of the original system. The proposed approach aims to leverage the system's existing capabilities together with the power of the crowd to expand the system's natural language repertoire and create richer interactions.

Our methodology begins with an existing corpus of dialogues, extracted from a spoken dialogue system that gives directions in a building. Further details on this system are given in §4.1. The extracted dialogue corpus contains phrases the system has generated, and crowd-workers construct alternates for these phrases, which can be plugged back into the system as *crowd templates*. We investigate via crowdsourcing the effect of the amount of surrounding context provided to workers on the perceived meaning, naturalness, and diversity of the alternates they produce, and study the acceptability of these alternates from a system developer viewpoint. Our results indicate that the crowd provides reasonable and diverse templates with this methodology. The developer evaluation suggests that additional work is necessary before we can automatically plug crowdsourced templates directly into the system.

We begin by discussing related work in §2. In §3, we detail the proposed methodology. In §4, we describe the experimental setup and results. Directions for future work are discussed in §5.

2 Related Work

Online crowdsourcing has gained popularity in recent years because it provides easy and cheap programmatic access to human intelligence. Researchers have proposed using crowdsourcing for a diverse set of natural language processing tasks, including paired data collection for training machine translation systems (Zaidan and Callison-Burch, 2011), evaluation of NLP systems (Callison-Burch and Dredze, 2010) and speech transcriptions (Parent and Eskenazi, 2010). A popular task targeting language diversity is paraphrase generation, which aims at collecting diverse phrases while preserving the original meaning. Crowdsourcing paraphrase generation has been studied for the purposes of plagiarism detection (Burrows and Stein, 2013), machine translation (Buzek et al., 2010), and expanding language models used in mobile applications (Han and Ju, 2013). Automated and crowd-based methods have been proposed for evaluating paraphrases generated by the crowd (Denkowski and Lavie, 2010; Tschirsich and Hintz, 2013). Researchers have proposed workflows to increase the diversity of language collected with crowd-based paraphrase generation (Negri et al., 2012) and for reducing the language bias in generation by initiating generation with visual input (Chen and Dolan, 2011). While paraphrase generation typically aims to preserve the meaning of a phrase without considering its use beyond the sentence level, we focus on collecting diverse language to be used directly in a dialogue system in a way that agrees with the full dialogue context.

Manually authoring dialogue systems has been identified as a challenging and time-consuming task (Ward and Pellom, 1999), motivating researchers to explore opportunities to use the crowd to improve and evaluate dialogue systems. Wang et al. (2012) proposed methods to acquire corpora for NLP systems using semantic forms as seeds, and for analyzing the quality of the collected corpora. Liu et al. (2010) used crowdsourcing for free-form language generation and for semantic labeling, with the goal of generating language corpora for new domains. Crowd-workers contribute to dialogue generation in real-time in the Chorus system by providing input about what the system should say next (Lasecki et al., 2013). Crowdsourcing has also been used with some success for dialogue system evaluation (Jurčíček et al., 2011).

Previous work on increasing language diversity in dialogue systems with crowdsourcing has focused on learning about diversity in user input to improve components such as speech recognition and language understanding (e.g., Wang et al. (2012)). Instead, our work focuses on adding diversity to system outputs. Mairesse et al. (2010) followed a similar approach to the work reported here, using crowdsourcing to collect paraphrases for a dialogue system in the restaurant domain. However, the focus of the Mairesse et al. work was on training an NLG module using this data. Our work focuses on crowdsourcing techniques to extract relevant paraphrases, examining the effect of context on their suitability and generalizability.

3 Methodology

Our methodology for developing natural language generation templates is illustrated by the pipeline in Figure 1. This pipeline is designed for dialogue systems that use a template-based natural language generation component. It assumes that the given system has an initial set of language generation templates that have been manually authored, and expands from there. The initial system is used to collect a corpus of dialogues, which we will refer to as the **dialogue seed corpus**, through interactions with users. Based on the dialogue seed corpus, we automatically construct a set of **generation HITs**, web-based crowdsourcing tasks that are used to elicit paraphrases from crowd-workers for instantiated system templates. A generation HIT displays one of the system turns extracted from a system dialogue, with a phrase highlighted, and different amounts of surrounding context in different conditions. The worker is asked to replace the phrase with another one that keeps the same meaning and the coherence of the interaction. If slots are marked in the original, they must be preserved by the worker, which allows us to easily convert the elicited paraphrases to crowd templates. Once a corpus of crowd templates are collected in this fashion, a system developer may filter and decide which to add as viable alternatives to the system's existing list of language generation templates (top path in the pipeline from Figure 1).

We also construct a set of **evaluation HITs** and post them to the crowd to assess the suitability and relative naturalness of the crowd templates (bottom path in the pipeline from Figure 1.) We study how the scores obtained in this crowd-evaluation may be used to help filter the set of new templates that are presented as candidates to the system developer. In the following subsections, we describe each of the pipeline components in detail.

3.1 Dialogue Seed Corpus

We assume as a starting point an existing dialogue system that uses a template-based language generation component. The system uses a set of templates T, which are instantiated with slots filled to generate system phrases. A system turn may contain one or more such phrases connected together. For instance, in the dialogue fragments shown in Figure 2, the template *"Sorry, that was [Place] you wanted, right?"* generates at runtime *"Sorry, that was Ernestine Patrick's office you wanted,*

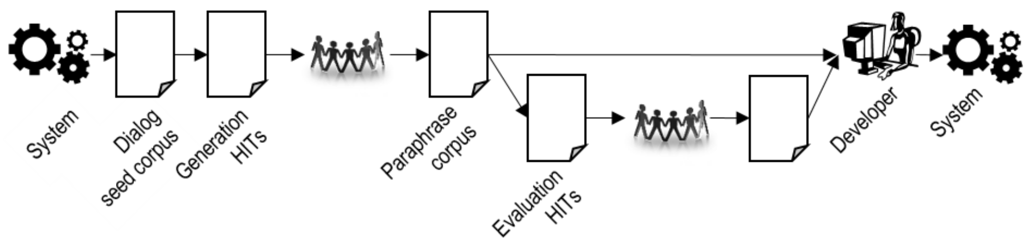
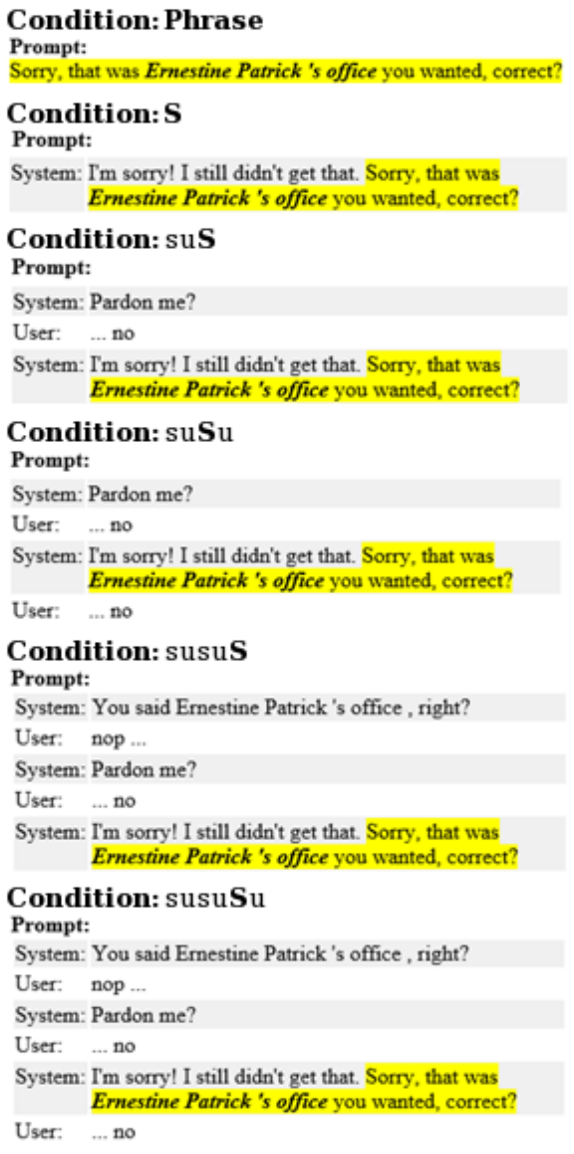


Figure 1: Pipeline for crowd-based development of natural language generation templates.



Condition: Phrase
Prompt:
Sorry, that was ***Ernestine Patrick 's office*** you wanted, correct?

Condition: S
Prompt:
System: I'm sorry! I still didn't get that. Sorry, that was ***Ernestine Patrick 's office*** you wanted, correct?

Condition: suS
Prompt:
System: Pardon me?
User: ... no
System: I'm sorry! I still didn't get that. Sorry, that was ***Ernestine Patrick 's office*** you wanted, correct?

Condition: suSu
Prompt:
System: Pardon me?
User: ... no
System: I'm sorry! I still didn't get that. Sorry, that was ***Ernestine Patrick 's office*** you wanted, correct?
User: ... no

Condition: susuS
Prompt:
System: You said Ernestine Patrick 's office , right?
User: nop ...
System: Pardon me?
User: ... no
System: I'm sorry! I still didn't get that. Sorry, that was ***Ernestine Patrick 's office*** you wanted, correct?

Condition: susuSu
Prompt:
System: You said Ernestine Patrick 's office , right?
User: nop ...
System: Pardon me?
User: ... no
System: I'm sorry! I still didn't get that. Sorry, that was ***Ernestine Patrick 's office*** you wanted, correct?
User: ... no

Figure 2: Generation HIT excerpts in six different context conditions (w/o instructions, examples).

right?”. Statistics on the dialogue seed corpus used in this study are provided in §4.2.

The proposed methodology does not require transcriptions of user utterances in the dialogue seed corpus; instead, it utilizes the recognition results for each user turn. The primary reason behind this choice is that a dialogue that contains recognized user turns may be more coherent than one that contains transcripts and can be generated automatically, as the dialogue manager generates system responses based on the recognition results. However, turn-overtaking issues and recognition problems sometimes resulted in incoherent dialogue interactions. Improving speech recognition remains an area for future work.

3.2 Generation HITs

We use the dialogue seed corpus to produce generation HITs to elicit paraphrases for system phrases from crowd-workers. In the simplest form, a generation HIT might present a single system phrase to the worker. We hypothesize that the surrounding context may be an important factor in facilitating the construction of appropriate paraphrases, affecting their diversity, naturalness, generalizability, etc.; we therefore investigate the effect of presenting varying amounts of dialogue context to the worker.

Specifically, given a system phrase corresponding to a template t instantiated in a dialogue, we investigate six different dialogue context conditions. A phrase in a condition presented to a crowd-worker will be referred to as a **seed**, p. Examples of seeds in each condition are illustrated in Figure 2. In the first condition, denoted $Phrase$, a seed is presented to the worker in isolation. In the second condition, denoted $\mathbf{S}$, the entire system turn containing p is presented to the worker, with p highlighted. In the next 4 conditions, denoted $su\mathbf{S}$, $su\mathbf{S}u$, $susu\mathbf{S}$, $susu\mathbf{S}u$, seeds are presented in increasingly larger contexts including one or two previous system and user turns (denoted with lowercase ‘s’ and ‘u’ in the encoding above), followed by the system turn $\mathbf{S}$ that contains the highlighted seed p, followed in two conditions ($susu\mathbf{S}u$ and $su\mathbf{S}u$) by another user turn. Not all context conditions are applicable for each instantiated template, e.g., conditions that require previous context, such as $su\mathbf{S}$, cannot be constructed for phrases appearing in the first system turn. We follow a between-subjects design, such that each worker works on only a single condition.

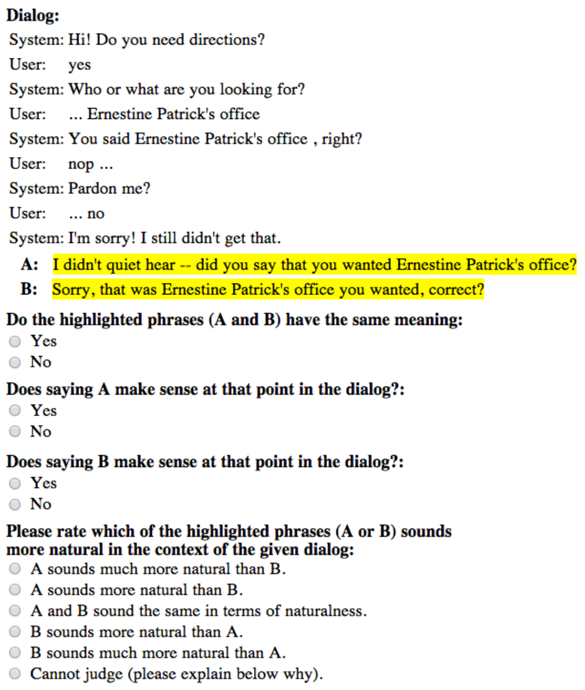
Each generation HIT elicits a paraphrase for a seed. The HIT additionally contains instructions and examples of what workers are expected to do and not to do.[1] We instruct workers to read the dialogue presented and rephrase the highlighted phrase (seed) so as to preserve the meaning and the cohesion of the interaction. To identify slots accurately in the crowd-generated paraphrases, we mark slot values in the given seed with bold italics and instruct workers to keep this portion exactly the same in their paraphrases (see Figure 2). These paraphrases are then turned into **crowd templates** following 3 basic steps: (1) Spelling error correction; (2) Normalization;[2] and (3) Replacing filled slots in the worker's paraphrase with the slot name. We ask workers to provide paraphrases (in English) that differ from the original phrase more substantially than by punctuation changes, and implement controls to ensure that workers enter slot values.

In completing the generation tasks, the crowd produces a corpus of paraphrases, one paraphrase for each seed. For example, *"I apologize, are you looking for Ernestine Patrick's office?"*, is a paraphrase for the highlighted seed shown in Figure 2. As we have asked the workers not to alter slot values, crowd templates can easily be recovered, e.g., *"I apologize, are you looking for [Place]?"*

3.3 Evaluation HITs

A good crowd template must minimally satisfy two criteria: (1) It should maintain the meaning of the original template; and (2) It should sound natural in *any* dialogue context where the original template was used by the dialogue manager, i.e., it should generalize well, beyond the specifics of the dialogue from which it was elicited.

To assess crowd template quality, we construct evaluation HITs for each crowd template. Instantiated versions of the original template and the crowd template are displayed as options A and B (with randomized assignment) and highlighted as part of the entire dialogue in which the original template was used (see Figure 3). In this **in-context (IC)** evaluation HIT, the worker is asked whether the instantiated crowd template has the same meaning as the original, and which is more natural. In addition, because the original dialogues



Dialog:
System: Hi! Do you need directions?
User: yes
System: Who or what are you looking for?
User: ... Ernestine Patrick's office
System: You said Ernestine Patrick's office , right?
User: nop ...
System: Pardon me?
User: ... no
System: I'm sorry! I still didn't get that.
A: I didn't quiet hear -- did you say that you wanted Ernestine Patrick's office?
B: Sorry, that was Ernestine Patrick's office you wanted, correct?

Do the highlighted phrases (A and B) have the same meaning:
○ Yes
○ No

Does saying A make sense at that point in the dialog?:
○ Yes
○ No

Does saying B make sense at that point in the dialog?:
○ Yes
○ No

Please rate which of the highlighted phrases (A or B) sounds more natural in the context of the given dialog:
○ A sounds much more natural than B.
○ A sounds more natural than B.
○ A and B sound the same in terms of naturalness.
○ B sounds more natural than A.
○ B sounds much more natural than A.
○ Cannot judge (please explain below why).

Figure 3: Example evaluation HIT excerpt.

were sometimes incoherent (see §3.1), we also asked the evaluation workers to judge whether the given phrases made sense in the given context.

Finally, in order to assess how well the crowd template generalizes across different dialogues, we use a second, **out-of-context (OOC)** evaluation HIT. For each crowd template, we randomly selected a new dialogue where the template t appeared. The out-of-context evaluation HIT presents the instantiated original template and crowd template in this new dialogue. The crowdworkers thus assess the crowd template in a dialogue context different from the one in which it was collected. We describe the evaluation HITs in further detail in §4.

3.4 Developer Filtering

While a crowd-based evaluation can provide insights into the quality of the crowd templates, ultimately, whether or not a template is appropriate for use in the dialogue system depends on many other factors (e.g., register, style, expectations, system goals, etc.). The last step in the proposed methodology is therefore a manual inspection of the crowd templates by a system developer, who assesses which are acceptable for use in the system without changes.

[1] Instructions available at m-mitchell.com/corpora.html.

[2] We normalize capitalization, and add punctuation identical to the seed when no punctuation was provided.

Figure 4: Directions Robot system.

4 Experiments and Results

We now describe our experiments and results. We aim to discover whether *there is an effect of the amount of surrounding context on perceived crowd template naturalness*. We additionally explore whether the crowd template retains the meaning of the original template, whether they both make sense in the given context, and the diversity of the templates that the crowd produced for each template type. We report results when the templates are instantiated *in-context*, in the original dialogue; and *out-of-context*, in a new dialogue. We first describe the experimental test-bed and the corpora used and collected below.

4.1 Experimental Platform

The test-bed for our experiments is Directions Robot, a situated dialogue system that provides directions to peoples' offices, conference rooms, and other locations in our building (Bohus et al., 2014). The system couples a Nao humanoid robot with a software infrastructure for multi-modal, physically situated dialogue (Bohus and Horvitz, 2009) and has been deployed for several months in an open space, in front of the elevator bank on the 3rd floor of our building (see Figure 4). While some of the interactions are need-based, e.g., visitors coming to the building for meetings, many are also driven by curiosity about the robot.

The Directions Robot utilizes rule-based natural language generation, with one component for giving directions based on computed paths, and another component with 38 templates for the rest of the dialogue. Our experimentation focuses on these 38 templates. As the example shown in Figure 2 illustrates, slots are dynamically filled in at run-time, based on the dialogue history.

We conducted our experiments on a general-

Cond.	Crowd Generation				Crowd Eval.	
	# Gen HITs (× 3)	# *w*	Time/ HIT (sec)	# Uniq. Para.	# Eval HITs (× 5)	Time/ HIT (sec)
Phrase	767	26	34.7	1181	1126	29.4
S	860	28	30.8	1330	1260	39.2
suS	541	26	33.3	1019	772	30.5
suSu	265	24	38.8	531	392	32.6
susuS	360	24	41.0	745	572	32.3
susuSu	296	28	42.9	602	440	34.4
Total	3089	-	-	5408	4562	-
Average	-	26	36.9	-	-	33.1

Table 1: Statistics for the crowd-based generation and evaluation processes. Each generation HIT was seen by 3 unique workers and each evaluation HIT was seen by 5 unique workers. *#w* represents number of workers. For evaluation, *#w* = 231.

purpose crowdsourcing marketplace, the Universal Human Relevance System (UHRS).[3] The marketplace connects human intelligence tasks with a large population of workers across the globe. It provides controls for selecting the country of residence and native languages for workers, and for limiting the maximum number of tasks that can be done by a single worker.

4.2 Crowd-based Generation

Dialogue seed corpus We used 167 dialogues collected with the robot over a period of one week (5 business days) as the dialogue seed corpus. The number of turns in these dialogues (including system and user) ranges from 1 to 41, with a mean of 10 turns. 30 of the 38 templates (79%) appeared in this corpus.

Generation HITs We used the dialogue seed corpus to construct generation HITs, as described in §3.2. In a pilot study, we found that for every 10 instances of a template submitted to the crowd, we received approximately 6 unique paraphrases in return, with slightly different ratios for each of the six conditions. We used the ratios observed for each condition in the pilot study to down-sample the number of instances we created for each template seen more than 10 times in the corpus. The total number of generation HITs resulting for each condition is shown in Table 1.

Crowd generation process Statistics on crowd generation are shown in Table 1. Each worker could complete at most 1/6 of the total HITs for that condition. We paid 3 cents for each genera-

[3]This is a Microsoft-internal crowdsourcing platform.

tion HIT, and each HIT was completed by 3 unique workers. From this set, we removed corrupt responses, and all paraphrases for a generation HIT where at least one of the 3 workers did not correctly write the slot values. This yielded a total of 9123 paraphrases, with 5408 unique paraphrases.

4.3 Crowd-based Evaluation

Evaluation HITs To keep the crowd evaluation tractable, we randomly sampled 25% of the paraphrases generated for all conditions to produce evaluation HITs. We excluded paraphrases from seeds that did not receive paraphrases from all 3 workers or were missing required slots. As discussed in §3, paraphrases were converted to crowd templates, and each crowd template was instantiated in the original dialogue, *in-context* (IC) and in a randomly selected *out-of-context* (OOC) dialogue. The OOC templates were instantiated with slots relevant to the chosen dialogue. This process yielded 2281 paraphrases, placed into each of the two contexts.

Crowd evaluation process As discussed in §3.3, instantiated templates (crowd and original) were displayed as options A and B, with randomized assignment (see Figure 3). Workers were asked to judge whether the original and the crowd template had the same meaning, and whether they made sense in the dialogue context. Workers then rated which was more natural on a 5-point ordinal scale ranging from -2 to 2, where a -2 rating marked that the original was much more natural than the crowd template. Statistics on the judgments collected in the evaluation HITs are shown in Table 1. Workers were paid 7 cents for each HIT. Each worker could complete at most 5% of all HITs, and each HIT was completed by 5 unique workers.

Outlier elimination One challenge with crowdsourced evaluations is noise introduced by spammers. While questions with known answers may be used to detect spammers in objective tasks, the subjective nature of our evaluation tasks makes this difficult: a worker who does not agree with the majority may simply have different opinions about the paraphrase meaning or naturalness. Instead of spam detection, we therefore seek to identify and eliminate outliers; in addition, as previously discussed, each HIT was performed by 5 workers, in an effort to increase robustness.

We focused attention on workers who performed at least 20 HITs (151 of 230 workers, covering 98% of the total number of HITs). Since we randomized the A/B assignment of instantiated original templates and crowd templates, we expect to see a symmetric distribution over the relative naturalness scores of all judgments produced by a worker. To identify workers violating this expectation, we computed a score that reflected the symmetry of the histogram of the naturalness votes for each worker. We considered as outliers 6 workers that were more than z=1.96 standard deviations away from the mean on this metric (corresponding to a 95% confidence interval). Secondly, we computed a score that reflected the percentage of tasks where a worker was in a minority, i.e., had the single opposing vote to the other workers on the *same meaning* question. We eliminated 4 workers, who fell in the top 97.5 percentile of this distribution. We corroborated these analyses with a visual inspection of scatterplots showing these two metrics against the number of tasks performed by each judge.[4] As one worker failed on both criteria, overall, 9 workers (covering 9% of all judgements) were considered outliers and their responses were excluded.

4.4 Crowd Evaluation Results

Meaning and Sense Across conditions, we find that most crowd templates are evaluated as having the *same meaning* as the original and *making sense* by the majority of workers. Evaluation percentages are shown in Table 2, and are around 90% across the board. This suggests that in most cases, the generation task yields crowd templates that meet the goal of preserving the meaning of the original template.

Naturalness To evaluate whether the amount of surrounding context has an effect on the perceived naturalness of a paraphrase relative to the original phrase, we use a Kruskal-Wallis (KW) test on the mean scores for each of the paraphrases, setting our significance level to .05. A Kruskal-Wallis test is a non-parametric test useful for significance testing when the independent variable is categorical and the data is not assumed to be normally distributed. We find that there is an effect of condition on the relative naturalness score (KW chi-squared = 15.9156, df = 5, p = 0.007) when crowd

[4]Scatterplots available at m-mitchell.com/corpora.html.

Cond.	Crowd Evaluation								Developer Evaluation		
	% Same Meaning		% Makes Sense		Avg. Relative Naturalness		Avg. D-score		% Dev. Accepted		Avg. D-score
	IC	**OOC**	**IC**	**OOC**	**IC**	**OOC**	**IC**	**OOC**	**All**	**Seen>1**	
Phrase	92	91	90	90	-.54 (.66)	-.50 (.61)	.67	.67	37	67	.30
S	91	89	88	88	-.50 (.65)	-.47 (.66)	.68	.64	35	53	.29
suS	84	87	85	87	**-.37** (.65)	**-.37** (.61)	.70	.70	40	63	.41
suSu	88	85	**95**	88	-.48 (.62)	-.43 (.61)	.76	.71	38	50	.39
susuS	**94**	**94**	91	**94**	-.43 (.70)	-.39 (.67)	**.81**	**.80**	38	**78**	.34
susuSu	91	89	92	86	-.40 (.61)	-.38 (.66)	.73	.74	**45**	67	**.42**

Table 2: % same meaning, % makes sense, and average relative naturalness (standard deviation in parentheses), measured in-context (IC) and out-of-context (OOC); crowd-based and developer-based diversity score (D-score); developer acceptance rate computed over all templates, and those seen more than once. The $susuS$ condition yields the most diverse templates using crowd-based metrics; removing templates seen once in the evaluation corpus, this condition has the highest acceptance in the developer evaluation.

templates are evaluated in-context, but not out-of-context (KW chi-squared = 9.4102, df = 5, p-value = 0.09378). Average relative naturalness scores in each condition are shown in Table 2.

Diversity We also assess the diversity of the templates elicited from the crowd, based on the evaluation set. Specifically, we calculate a diversity score (D-score) for each template type t. We calculate this score as the number of unique crowd template types for t voted to make sense and have the same meaning as the original by the majority, divided by the total number of seeds for t with evaluated crowd templates. More formally, let P be the original template instantiations that have evaluated crowd templates, M the set of unique crowd template types voted as having the *same meaning* as the original template by the majority of workers, and S the set of unique crowd template types voted as *making sense* in the dialogue by the majority of workers. Then:

$$\text{D-score}(t) = \frac{|M \cap S|}{|P|}$$

The average diversity scores across all templates for each condition are shown in Table 2. We find the templates that yield the most diverse crowd templates include `WL_Retry` *“Where are you trying to get to in this building?”* and `OK_Help`, *“Okay, I think I can help you with that”*, which have a diversity rating of 1.0 in several conditions: for each template instance we instantiate (i.e., each generation HIT), we get a new, unique crowd template back. Example crowd templates for the `OK_Help` category include *“I believe I can help you find that”* and *“I can help you ok”*. The templates with the least diversity are those for `Hi`, which has a D-score around 0.2 in the S and $Phrase$ conditions.

4.5 Developer Acceptability Results

For the set of crowd templates used in the crowd-based evaluation process, one of the system developers[5] provided binary judgments on whether each template could be added (without making any changes) to the system or not. The developer had access to the original template, extensive knowledge about the system and domain, and the way in which each of these templates are used.

Results indicate that the developer retained 487 of the 1493 unique crowd templates that were used in crowd-evaluation (33%). A breakdown of this acceptance rate by condition is shown in Table 2. When we eliminate templates seen only once in the evaluation corpus, acceptability increases, at the expense of recall. We additionally calculate a diversity score from those templates accepted by the developer, which is simply the number of crowd template types accepted by the developer, divided by the total number of seeds used to elicit the crowd templates in the developer's evaluation, for each template type t.

The developer evaluation revealed a wide range of reasons for excluding crowd templates. Some of the most common were lack of grammaticality, length (some paraphrases were too long/short), stylistic mismatch with the system, and incorrect punctuation. Other reasons included register issues, e.g., too casual/presumptive/impolite, issues of specificity, e.g., template was too general, and issues of incompatibility with the dialogue state and turn construction process. Overall, the developer interview highlighted very specific system

[5]The developer was not an author of this paper.

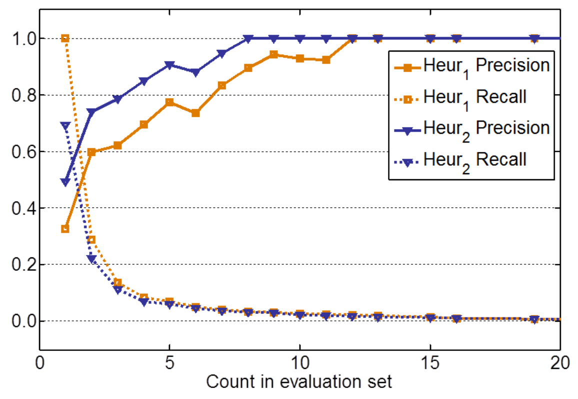


Figure 5: Precision and recall for heuristics.

and domain knowledge in the selection process.

4.6 Crowd-based Evaluation and Developer Acceptability

We now turn to an investigation of whether statistics from the crowd-based generation and evaluation processes can be used to automatically filter crowd templates. Specifically, we look at two heuristics, with results plotted in Figure 5. These heuristics are applied across the evaluation corpus, collating data from all conditions. The first heuristic, Heur$_1$, uses a simple threshold on the number of times a crowd template occurred in the evaluation corpus.[6] We hypothesize that more frequent paraphrases are more likely to be acceptable to the developer, and in fact, as we increase the frequency threshold, precision increases and recall decreases.

The second heuristic, Heur$_2$, combines the threshold on counts with additional scores collected in the out-of-context crowd-evaluation: It only considers templates with an aggregated judgment on the *same meaning* question greater than 50% (i.e., the majority of the crowd thought the paraphrase had the same meaning as the original), and with an aggregated relative naturalness score above the overall mean. As Figure 5 illustrates, different tradeoffs between precision and recall can be achieved via these heuristics, and by varying the count threshold.

These results indicate that developer filtering remains a necessary step for adding new dialogue system templates, as the filtering process cannot yet be replaced by the crowd-evaluation. This is not surprising since the evaluation HITs did not express all the different factors that we found the developer took into account when selecting templates, such as style decisions and how phrases are combined in the system to form a dialogue. Future work may consider expanding evaluation HITs to reflect some of these aspects. By using signals acquired through crowd generation and evaluation, we should be able to reduce the load for the developer by presenting a smaller and more precise candidate list at the expense of reductions in recall.

5 Discussion

We proposed and investigated a methodology for developing a corpus of natural language generation templates for a spoken dialogue system via crowdsourcing. We investigated the effect of the context we provided to the workers on the perceived meaning, naturalness, and diversity of the alternates obtained, and evaluated the acceptability of these alternates from a system developer viewpoint.

Our results show that the crowd is able to provide suitable and diverse paraphrases within this methodology, which can then be converted into crowd templates. However, more work is necessary before elicited crowd templates can be plugged directly into a system.

In future work, we hope to continue this process and investigate using features from the crowd and judgments from system developers in a machine learning paradigm to automatically identify crowd templates that can be directly added to the dialogue system. We would also like to extend beyond paraphrasing single templates to entire system turns. With appropriate controls and feature weighting, we may be able to further expand dialogue capabilities using the combined knowledge of the crowd. We expect that by eliciting language templates from multiple people, as opposed to a few developers, the approach may help converge towards a more natural distribution of alternative phrasings in a dialogue. Finally, future work should also investigate the end-to-end effects of introducing crowd elicited templates on the interactions with the user.

Acknowledgments

Thanks to members of the ASI group, Chit W. Saw, Jason Williams, and anonymous reviewers for help and feedback with this research.

[6]Since the evaluation corpus randomly sampled 25% of the generation HITs output, this is a proxy for the frequency with which that template was generated by the crowd.

References

D. Bohus and E. Horvitz. 2009. Dialog in the open world: Platform and applications. *Proceedings of ICMI'2009*.

Dan Bohus, C. W. Saw, and Eric Horvitz. 2014. Directions robot: In-the-wild experiences and lessons learned. *Proceedings of AAMAS'2014*.

Martin Potthast Burrows, Steven and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 43.

Olivia Buzek, Philip Resnik, and Benjamin B. Bederson. 2010. Error driven paraphrase annotation using mechanical turk. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Michael Denkowski and Alon Lavie. 2010. Exploring normalization techniques for human judgments of machine translation adequacy collected using amazon mechanical turk. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Matthai Philipose Han, Seungyeop and Yun-Cheng Ju. 2013. Nlify: lightweight spoken natural language interfaces via exhaustive paraphrasing. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*.

Filip Jurčíček, Simon Keizer, Milica Gašić, Frančois Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2011. Real user evaluation of spoken dialogue systems using amazon mechanical turk. *Proceedings of INTERSPEECH*, 11.

Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. 2013. Chorus: a crowd-powered conversational assistant. *Proceedings of the 26th annual ACM symposium on User interface software and technology*.

Sean Liu, Stephanie Seneff, and James Glass. 2010. A collective data generation method for speech language models. *Spoken Language Technology Workshop (SLT), IEEE*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Matteo Negri, Yashar Mehdad, Alessandro Marchetti, Danilo Giampiccolo, and Luisa Bentivogli. 2012. Chinese whispers: Cooperative paraphrase acquisition. *Proceedings of LREC*.

Gabriel Parent and Maxine Eskenazi. 2010. Toward better crowdsourced transcription: Transcription of a year of the let's go bus information system data. *Spoken Language Technology Workshop (SLT), IEEE*.

Martin Tschirsich and Gerold Hintz. 2013. Leveraging crowdsourcing for paraphrase recognition. *LAW VII & ID*, 205.

William Yang Wang, Dan Bohus, Ece Kamar, and Eric Horvitz. 2012. Crowdsourcing the acquisition of natural language corpora: Methods and observations. *Spoken Language Technology Workshop (SLT), IEEE*.

W. Ward and B. Pellom. 1999. The cu communicator system. *Proceedings of IEEE ASRU*.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Author Index