Towards Generating Effective Explanations of Logical Formulas: Challenges and Strategies

Alexandra Mayn and **Kees van Deemter**
Utrecht University
Department of Information and Computing Sciences
{a.mayn, c.j.vandeemter}@uu.nl

Abstract

While the problem of natural language generation from logical formulas has a long tradition, little attention has been paid to ensuring that the generated explanations are optimally helpful to the user. We discuss issues related to deciding what such output should look like and strategies for addressing those issues. We stress the importance of informing generation of NL explanations of logical formulas with reader studies and findings on the comprehension of logic from pragmatics and cognitive science. We illustrate the issues and potential ways of addressing them using a simple demo system's output generated from a propositional logic formula.

1 Introduction

The task of generating natural language text from logical form has a long and diverse tradition (Wang (1980), Appelt (1987), Shieber et al. (1990), to name a few early examples). It has been approached from a variety of perspectives targeting different use cases, including providing feedback to students of logic (Flickinger (2016)), users of logistic software (Kutlak and van Deemter (2015)), and explaining the output of a reasoning engine (Coppock and Baxter (2009)).

However, so far in this domain very little attention has been paid to generating output which is optimally helpful to the user, presumably a non-expert with little knowledge of formal logic, We aim to build a system which, given a logical formula, will produce an effective natural language explanation. To that end, we discuss challenges which might arise when building and scaling up such a generation system and strategies for addressing these challenges with the user in mind. We aim to conduct studies with potential users (students of logic and/or users of software which operates using logic) to determine what kinds of explanations the system should generate.

The rest of this paper is organized as follows: Section 2 summarizes related work. Section 3 discusses challenges and possibilities related to determining and ensuring effectiveness of the generated output for the users. Section 4 illustrates our points by means of a case study on producing explanations of a propositional logic formula. Section 5 concludes.

2 Related work

There have been a number of works aimed at generating text from logical form, using either rule-based (Shieber et al. (1990); Ranta (2011); De Roeck and Lowden (1986)) or statistical (Basile (2015); Zettlemoyer and Collins (2012); Lu and Ng (2011)) methods. However, only a few of them explicitly discuss the issues related to the comprehensibility and effectiveness of the generated output. De Roeck and Lowden (1986) opt for using indentation as opposed to linear text to minimize ambiguity of the generated text, while Ranta (2011)'s solution involves bulleted lists. Flickinger (2016) addresses a related issue, that of generating multiple paraphrases for a logical formula, with a view to subsequently selecting the best one - as many as almost 4500 paraphrases are generated for one formula, but the issue of filtering out ambiguous paraphrases and selecting the best one is left to future work. Kutlak and van Deemter (2015) apply transformations at the logic level with the aim of making the formula more suitable for generation.

Studies with human participants to determine what output of NLG systems is preferable have been conducted in other domains. Eugenio et al. (2005) study the effect of aggregation in the output of an automated tutoring system on the learner and find that what they call functional aggregation,

Proceedings of the 2nd Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NLXAI 2020), pages 39–43,
Dublin, Ireland, 18 December 2020.

which produces summaries of text relating to the same object or function, benefits the user more than purely syntactic aggregation does. Khan et al. (2012) conduct reading studies to investigate comprehension of referring expressions with a view to improving an NLG system's output. They investigate the interaction between brevity and clarity, and find that brevity is preferred when all alternatives are unambiguous, and otherwise a longer but unambiguous alternative is preferred, which goes to show that there is an advantage to non-brief referring expressions in terms of comprehension. Khan et al. (2012)'s NLG algorithms thus incorporate a trade-off between clarity and brevity.

3 User-oriented explanations: Challenges and strategies

There are a number of questions which need to be answered when developing an explanation-producing system aimed at making the explanations maximally helpful to the user.

As Kutlak and van Deemter (2015) point out, it is not always the case that the inputted logical formula is in a form suitable for generation. Some formulas are unnecessarily complex and would therefore tend to produce unnecessarily complex text unless optimised first. To make matters trickier, for expressive logics it is often not decidable whether two logical formulas are equivalent to each other (termed "the problem of *logical form equivalence*" by Shieber (1993)), so heuristics need to be developed to decide what transformations to apply, and how many, and to determine how suitable a formula is for generation. Kutlak and van Deemter (2015) assume as a rule of thumb that transformations which will make a formula shorter are likely to also make it easier to comprehend. However, there are some cases where making the formula longer might be warranted, e.g. if that results in a clearer NL explanation. We believe that it would be beneficial to conduct empirical studies on comprehension and preference between text variants generated based on several equivalent formulas in order to develop such heuristics.

At the NL generation stage, there are important decisions to be made as well. Which phrasings should be used and which ones should be avoided? One of the aspects which can make generation from logic challenging is that the meaning of logical connectives is not always the same as that of their natural language counterparts (Grice (1975), Moeschler (2017)). For instance, Geis and Zwicky (1971) argue that an NL conditional is often used as a logical biconditional (for example, *If you go to the party, I'll go too* is understood to imply that *if you do not go, neither will I*), while Barker-Plummer et al. (2008) show that students of logic particularly struggle with the expression of the biconditional as *just in case* because it has a very different meaning in everyday natural language.

In terms of the form of the generated text, there are a number of alternatives which have been used - linear text, bulleted lists and indentation; these presentation decisions will have an effect on the comprehensibility of the generated output. Related, what is the optimal amount of aggregation in this context? Are there situations where it is preferable not to aggregate? We argue that these questions should be addressed through controlled user experiments where reading comprehension and speed for alternatives is compared along each of these dimensions.

We also believe that such resources as the Grade Grinder Corpus (Barker-Plummer et al., 2011), which contains students' attempts to convert natural language text to FOL, can also inform us about which natural language wordings are effective and which ones should be avoided by the generator. Both number and nature of incorrect attempts by the students can be used in gaining insights as to what realizations of connectives tend to be misunderstood and what they are misunderstood as. For instance (Barker-Plummer et al., 2008) find that many errors are made when formalizing a sentence in FOL requires reordering the antecedent and the consequent.

As has been pointed out in related work (Ranta (2011); Flickinger (2016)), *ambiguity* is a challenging aspect of this generation problem: if not controlled for, bracketing or negation scope ambiguity is likely to emerge. Ranta (2011) proposes using a parser test to determine whether the generated output can have multiple readings, and select an unambiguous one that way. We believe that that is an effective solution to the ambiguity problem. However, we can imagine a case where, for a sufficiently complex formula, the generator might only produce explanations with multiple readings, or the unambiguous variant is too clunky and difficult to read. In that case, it would be beneficial to know about the respective likelihood of the alternative readings for the user. It could be, for instance, that

a reading is identified by the parser which a human is unlikely to consider. With this likelihood information, one could, for instance, select an output which has the fewest possible readings, or only one likely reading. We intend to explore whether such an approximation of likelihood can be obtained using probabilistic parsing (Jurafsky and Martin, 2009, Chapter 14).

4 Case Study: Generation from Propositional Logic

We illustrate the above points by means of a concrete example. As a starting point, we built a simple system, which takes a propositional logic formula as input, parses it into a tree structure, optionally applies transformations to the tree, and realizes the output by reading off the tree, left to right. We chose propositional logic as a base case because it is one of the simplest logics in which many of the important discussed in Section 3 emerge.

Consider the following formula, which involves the block language from Tarski's World, a software component for teaching logic to students (Barwise et al., 2000):

(1) $\neg Cube(x) \wedge \neg(Smaller(x, y) \vee SameShape(x, y))$

At the tree level, we apply a number of meaning-preserving transformations. For any formula, there is an infinite set of formulas equivalent to it, therefore heuristics need to be developed as to what makes a formula a promising candidate for generation. For simplicity, we start with a set of 8 formulas equivalent to (1), obtained by distributing negation, applying De Morgan's laws, or reversing the order of the conjuncts and disjuncts. We pass each of these formulas to a simple generator, obtaining 8 wordings. For example, (1) is realized as:

(a) x is not a cube and it is not the case that x is smaller than y or x is the same shape as y.

Which, if any, of the generated versions should be the final output? At this stage, we run a syntactic parser on the generated text to identify how many and what kind of possible readings the generated text may have. We then determine how many distinct parses each of the texts has by computing which of the parses are equivalent to each other. We find that some generated text variants come out as unambiguous, while others have as many as 11 distinct parses.

We are not quite done yet. It is worth pointing out that the ambiguities which the parser detects might not perfectly predict what ambiguities might arise for people. For example, (a) can be parsed three ways, with *it is not the case* having either narrow or wide scope. However, one could argue that the narrow scope reading is a lot less likely. That could be determined using probabilistic parsing.

Conversely, there could emerge certain mirage ambiguities, where a sentence which is grammatically unambiguous could still be understood multiple ways by the reader, e.g. we can imagine *it is not the case that x is smaller than y or the same shape as y* being misunderstood as $\neg Smaller(x, y) \vee SameShape(x, y)$ (narrow scope of negation). Such cases seem more difficult to foresee. Reader studies could be helpful in gaining insight; complexity heuristics could also be introduced with the hope that less complex sentences would be less likely to give rise to such problems.

Besides ambiguity, there is another dimension along which the effectiveness of the generated text will vary: readability. Text length and naturalness both affect readability. Interestingly, ambiguity also interacts with readability: there is evidence that ambiguous sentences are processed faster than disambiguated ones, but only when the readers do not anticipate the need to answer in-detail questions about the read text (Swets et al., 2008). Methods like aggregation may be employed to improve readability. So, (1) could also be worded as:

(b) x is not a cube, nor is it smaller than or the same shape than y.

That phrasing is also no longer ambiguous, which illustrates that aggregation can have an impact on clarity as well as readability.

Of course, a yet more natural phrasing of (1) is as follows:

(c) x is not a cube. it is at least as large as y and has a different shape.

It would be challenging to generate (c) automatically given an input formula like (1) since the system would need to have information about what *not smaller* means. Kutlak and van Deemter (2015) allow the user to enter background axioms, which partially addresses the problem: the user would

need to explicitly indicate the equivalence between *not smaller* and *at least as large*, and between *not the same* and *different shape*, in order for such output to be generated.

In the above, we have experimented with an approach that computes many possible interpretations of each candidate NL text. An interesting avenue for further research is to investigate how such a brute-force approach may be approximated by a set of heuristics, which could then be used in an approach similar to the *revision process* for NLG (Inui et al., 1992) to avoid unnecessary computation: generating output which is estimated to be the best (i.e., most clear and natural), checking these constraints and repeating the process for the next best output if one of the constraints is violated. An open and challenging question is that of generality: if we identify a set of heuristics for a certain class of formulas, how well will they generalize to a different class of formulas or set of predicates? We aim to explore that through controlled experiments.

5 Conclusion

In this paper, we addressed an aspect of the design of an NLG system for explaining the meaning of logical formulas which has often been overlooked: the needs of the user. We discussed questions which we aim to answer when building such a system, such as logical simplifications, paraphrasing and ambiguity, and considered ways in which they can be informed: reading studies with potential users, work with corpora, and insights from cognitive science and pragmatics. We illustrated these questions and potential solutions by means of an example of generating text from a propositional logic formula.

6 Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 860621.

References

Douglas Appelt. 1987. Bidirectional grammars and the design of natural language generation systems. In *Theoretical Issues in Natural Language Processing 3*.

Dave Barker-Plummer, Richard Cox, and Robert Dale. 2011. Student translations of natural language into logic: The grade grinder corpus release 1.0. In *Proceedings of the 4th international conference on educational data mining*, pages 51–60.

Dave Barker-Plummer, Richard Cox, Robert Dale, and John Etchemendy. 2008. An empirical study of errors in translating natural language into logic. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 30.

Jon Barwise, John Etchemendy, Gerard Allwein, Dave Barker-Plummer, and Albert Liu. 2000. *Language, proof and logic*. CSLI publications.

Valerio Basile. 2015. From logic to language: Natural language generation from logical forms.

Elizabeth Coppock and David Baxter. 2009. A translation from logic to english with dynamic semantics. In *JSAI International Symposium on Artificial Intelligence*, pages 197–216. Springer.

Anne De Roeck and Barry GT Lowden. 1986. Generating english paraphrases from formal relational calculus expressions. In *Coling 1986 Volume 1: The 11th International Conference on Computational Linguistics*.

Barbara Di Eugenio, Davide Fossati, Dan Yu, Susan Haller, and Michael Glass. 2005. Aggregation improves learning: experiments in natural language generation for intelligent tutoring systems. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 50–57. Association for Computational Linguistics.

Dan Flickinger. 2016. Generating english paraphrases from logic. *From Semantics to Dialectometry*, pages 99–107.

Michael L Geis and Arnold M Zwicky. 1971. On invited inferences. *Linguistic inquiry*, 2(4):561–566.

Herbert P Grice. 1975. Logic and conversation. In *Speech acts*, pages 41–58. Brill.

Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1992. Text revision: A model and its implementation. In *International Workshop on Natural Language Generation*, pages 215–230. Springer.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Imtiaz H Khan, Kees van Deemter, and Graeme Ritchie. 2012. Managing ambiguity in reference generation: the role of surface structure. *Topics in Cognitive science*, 4(2):211–231.

Roman Kutlak and Kees van Deemter. 2015. Generating succinct english text from fol formulae. In *Procs. of First Scottish Workshop on Data-to-Text Generation*.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622.

Jacques Moeschler. 2017. What logic tells us about natural language1. *The Routledge handbook of pragmatics*, page 241.

Aarne Ranta. 2011. Translating between language and logic: what is easy and what is difficult. In *International Conference on Automated Deduction*, pages 5–25. Springer.

Stuart Shieber. 1993. The problem of logical-form equivalence. *Computational Linguistics*.

Stuart Shieber, Gertjan Van Noord, Fernando CN Pereira, and Robert C Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*.

Benjamin Swets, Timothy Desmet, Charles Clifton, and Fernanda Ferreira. 2008. Underspecification of syntactic ambiguities: Evidence from self-paced reading. *Memory & Cognition*, 36(1):201–216.

Juen-tin Wang. 1980. On computational sentence generation from logical form. In *COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics*.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.