Learning To Parse on Aligned Corpora (Rough Diamond)

Cezary Kaliszyk[1⋆], Josef Urban[⋆⋆2], and Jiří Vyskočil[3]

[1] University of Innsbruck, Austria
[2] Radboud University Nijmegen
[3] Czech Technical University

Abstract. One of the first big hurdles that mathematicians encounter when considering writing formal proofs is the necessity to get acquainted with the formal terminology and the parsing mechanisms used in the large ITP libraries. This includes the large number of formal symbols, the grammar of the formal languages and the advanced mechanisms instrumenting the proof assistants to correctly understand the formal expressions in the presence of ubiquitous overloading.
In this work we start to address this problem by developing approximate probabilistic parsing techniques that autonomously train disambiguation on large corpora. Unlike in standard natural language processing, we can filter the resulting parse trees by strong ITP and AR semantic methods such as typechecking and automated theorem proving, and even let the probabilistic methods self-improve based on such semantic feedback. We describe the general motivation and our first experiments, and build an online system for parsing ambiguous formulas over the Flyspeck library.

1 Introduction

Is it possible to automatically parse informal mathematical texts into formal ones and formally verify them? Four out of five ITP (interactive theorem proving) practitioners say no.[4] Even Andrzej Trybulec – an accomplished linguist by one of his professions and the father of human-like formal mathematical notation, linguistic typing mechanisms and proof style – used to quote the past work (e.g., by Zinn [10]) as discouraging from such efforts. We however believe that it is a good time to try, and in particular to try to automatically *learn* how to formalize ("semanticize") informal texts, based on the knowledge available in existing large formal corpora. There are several reasons [6].

First, statistical machine learning (data-driven algorithm design) has been responsible for several recent AI breakthroughs, including machine translation systems like Google Translate that automatically train on large aligned bilingual corpora. Similar successes are in query answering (IBM Watson) and autonomous car driving, which are arguably much more semantic domains than just "simple" natural language alignment. It seems today that as soon as there are sufficiently

⋆ Supported by the Austrian Science Fund (FWF): P26201.
⋆⋆ Supported by NWO grant nr. 612.001.208.
[4] Approximate results of an opinion poll run by the second author since 2000.

large datasets, data-driven algorithms can automatically learn complicated sets of rules – thus perhaps also the nontrivial mapping of informal to formal – that would be otherwise hard to program and maintain manually.

Second, recent formalization projects have produced large corpora that can – perhaps after additional annotation – be used for such experiments with machine learning of formalization. Further growth of such corpora is only a matter of time, and assisted formalization might help "bootstrap" this process, making it faster and faster due to the positive feedback loop from more data becoming available.

Third, statistical machine learning methods have recently really turned out to be useful in deductive AI domains such as automated reasoning in large theories [1] (ARLT). This shows that in practice, its inherent undecidability does not make mathematics into some special field where statistical techniques cannot apply. Quite the opposite: formal mathematical corpora seem to largely obey similar statistical laws as other texts produced by humans, and statistical and information-retrieval algorithms such as TF-IDF, naive Bayes, k-nearest neighbor, and kernel methods, are indispensable parts of the ARLT methods [4,7].

Finally, we believe that strong ARLT methods are a new useful weapon in auto-formalization, that can complement the statistical translation methods. This could result in hybrid understanding/thinking AI methods that self-improve on large annotated corpora by cycling between (i) statistical prediction of the text disambiguation based on learning from existing annotations and knowledge, and (ii) improving such knowledge by confirming or rejecting the predictions by the semantic ARLT methods. This point is quite unique to the domain of (informal/formal) mathematics, and a good independent reason for this AI research.

2 Contributions

Below we briefly present the first significant effort in statistical learning of parsing ambiguous formulas over a very large formal mathematical corpus – the Flyspeck project. The main result of this effort is a large-scale evaluation of the methods (Section 6), and the first version of an online system[5] (Section 5) that allows HOL Light and Flyspeck users to write ambiguous bracket-free formulas using many common ambiguous symbols, skipping disambiguation mechanisms such as casting functors. Such formulas are probabilistically parsed, using an efficiently implemented parsing system (Section 4) trained on the correct parse trees of all (about 22000) toplevel Flyspeck theorems (Section 3). The trained parsing system produces a required number of most probable parse trees, which are then further filtered by parsing and type checking in HOL Light, presenting the most probable filtered parses in a disambiguated HOL Light notation. Simultaneously, these typechecked formulas are given to the HOL(y)Hammer system which then further marks those that can be automatically proved using the whole Flyspeck library and thus are much more likely to have the intended meaning.

In some sense we thus implement the first version of "jumping" between probabilistic and semantic parsing used by informal mathematicians, as fittingly described by Dijkstra [2]:

[5] `http://colo12-c703.uibk.ac.at/hh/parse.html`

> The bulk of traditional mathematics is highly informal: formulae are not manipulated in their own right, they are all the time viewed as denoting something, as standing for something else. The bulk of traditional mathematics is characterized by a constant jumping back and forth between the formulae and their interpretation and the latter has to carry the burden of justifying the manipulations. The manipulations of the formulae are not justified by an appeal to explicitly stated rules but by the appeal to the interpretation in which the manipulations are "obviously" OK. By and large, the mathematicians form a much more informal lot than they are aware of.

3 Making Ambiguous Data

While our ultimate goal is to parse the informal LaTeX formulas that have been aligned by Hales with the formal Flyspeck formulas [3,8], our initial research approach is to explore parsing of increasingly ambiguous versions of the formal HOL Light and Flyspeck theorems. Making the formal notation more ambiguous turns out to be relatively easy, allowing us to experiment with different kinds of ambiguities and their amount. We did the initial development and evaluations on a subset of 550 Flyspeck trigonometric theorems.[6] This subset is interesting and suitable, because it contains complex and real versions of trigonometric functions (e.g. `csin` instead of `sin`) and frequently uses casting functions such as e.g. `Cx` which casts a real number to a complex number.

It could be however argued that this subset is a toy domain which does not differ much from manually prepared examples, and where manual tweaking of the algorithms is easy and not particularly useful to interested Flyspeck users. That is why we have tried to scale the parsing methods to the whole Flyspeck, making a large number of ambiguous symbols and sentences, hopefully in a way that makes writing such sentences an interesting experiment for some users. The transformations (*informalizing*) consist of:

- Using 72 overloaded instances defined in HOL Light/Flyspeck, like `("+", "vector_add")` . The result sentence would use `+` instead of `vector_add`.
- Getting the (currently 108) infix operators from HOL Light, and printing them as infix in the informalized sentences. Since `+` is declared as infix, `vector_add u v`, would thus result in `u + v`.
- Getting all "prefixed" symbols from the list of 1000 most frequent symbols by searching for: `real_, int_, vector_, nadd_, treal_, hreal_, matrix_, complex_` and making them ambiguous by forgetting the prefix.
- Similar overloading of various other symbols that disambiguate overloading, for example the "c"-versions of functions such as `ccos cexp clog csin`, similarly for `vsum, rpow, nsum, list_sum`, etc. In the end the above steps yield a list of about 70 overloaded symbols corresponding to some 200 nonambiguous symbols used very frequently throughout HOL Light and Flyspeck.
- Deleting all brackets, type annotations, and the 10 most frequent casting functors such as `Cx` and `real_of_num` (which alone is used 17152 times).

[6] Exactly, the theorems containing substrings sin, cos and tan.

4 Probabilistic Parsing and Its Extensions

Our task is to assign to each informalized sentence (a list of often ambiguous symbols) its most probable HOL parse tree, with all terms annotated by types. For example, the correct parse tree for `REAL_NEGNEG: ! A0 -- -- A0 = A0` is:

```
(Comb (Const "!" (Tyapp "fun" (Tyapp "fun" (Tyapp "real") (Tyapp "bool")) (Tyapp "bool"))) (Abs
"A0" (Tyapp "real") (Comb (Comb (Const "=" (Tyapp "fun" (Tyapp "real") (Tyapp "fun" (Tyapp "real")
(Tyapp "bool")))) (Comb (Const "real_neg" ... (Var "A0" (Tyapp "real")))))
```

For this, after initial tries with the Stanford Statistical Parser,[7] we wrote our custom OCaml implementation of the CYK chart parsing algorithm [9] for probabilistic context-free grammars (PCFG), and a custom tree transformation tool that enables us to create ambiguous sentences and annotated training input ("grammar") trees for the parser from the HOL parse trees. These grammar trees treat each (possibly complicated) type as the resulting nonterminal assigned to parsing each term, and additionally each ambiguous symbol (terminal) such as ``--'' is wrapped in its disambiguating nonterminal, such as `$#real_neg`. This is analogous to annotating with word-sense disambiguations for linguistic PCFG tools, however our "semantic concepts" are not word senses but HOL types and unambiguous symbols. The tool also applies infix notations and replaces casting terminals with their corresponding nonterminals. For example the complex casting terminal `Cx` disappears, since we do not want the user to write it explicitly, and is replaced in the grammar tree by the corresponding "semantic" nonterminal `$#Cx` applied to the corresponding ambiguous subterm.

We produce two versions of the parser, a standard one and a HOL-specialized one that prunes the parse space by additional fast lightweight semantic restrictions, such as compatibility constraints on types of free variables in parsed subtrees. Both versions have a training and testing phase. In the training phase all the grammar trees (we use all 22000 trees for Flyspeck formulas by default, but this can be further limited) are used to generate grammar rules about the terminals and nonterminals and their probabilities, generating a binarized PCFG. In the testing (evaluation) phase the PCFG is used to parse a given ambiguous sentence with a required number of best parses. Efficient indexing is used to prune the search space, and the parse limit is used to prune improbable parsing subtrees, making it reasonably fast (on average 4 seconds for a Flyspeck theorem) to get the 20 most probable parses. The resulting grammar trees are again transformed back into a HOL parse tree, to which HOL parsing and typechecking is applied as an additional filter. Since all these three parts (CYK, transformations, and HOL Light routines) are written in OCaml, their tight integration is possible, offering further future options such as full HOL-based pruning of untypable subtrees during the CYK parsing, etc. The so far implemented HOL-based extensions of CYK, include the variable typing constraints, special treatment of lambda abstractions, and allowing all unknown symbols to have small nonzero probability of being a variable.

[7] `http://nlp.stanford.edu/software/lex-parser.shtml`

5 Online Parsing System

Since we are very interested in seeing the probabilistic parsing in action, we deploy the whole parsing toolchain as an online service[8] that further uses the HOL(y)Hammer AI/ATP system [5] for even stronger semantic filtering. The service allows HOL Light and Flyspeck users to write ambiguous formulas using many common ambiguous symbols and omitting brackets and casting functors. For example, the top two parses out of allowed 16 for
‘‘`sin 0 * x = cos pi / 2`’’ are

```
sin (&0) * A0 = cos (pi / &2) where A0:real
sin (&0) * A0 = cos pi / &2 where A0:real
```

where only the first one can be automatically proved by HOL(y)Hammer. The user can add brackets to limit the parses, and then for example
‘‘`sin ( 0 * x ) = cos pi / 2`’’ produces 16 parses of which 11 get type-checked by HOL Light as follows, with all but three being proved by HOL(y)Hammer:

```
sin (&0 * A0) = cos (pi / &2) where A0:real
sin (&0 * A0) = cos pi / &2 where A0:real
sin (&0 * &A0) = cos (pi / &2) where A0:num
sin (&0 * &A0) = cos pi / &2 where A0:num
sin (&(0 * A0)) = cos (pi / &2) where A0:num
sin (&(0 * A0)) = cos pi / &2 where A0:num
csin (Cx (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0) * A0) = ccos (Cx (pi / &2)) where A0:real^2
Cx (sin (&0 * A0)) = ccos (Cx (pi / &2)) where A0:real
csin (Cx (&0 * A0)) = Cx (cos (pi / &2)) where A0:real
csin (Cx (&0) * A0) = Cx (cos (pi / &2)) where A0:real^2
```

The HOL-specialized probabilistic parsing and HOL typechecking phases are fast (given that the input sentence is not too long), because we limit the number of required parses to 16, and preselect only the 1024 closest grammar trees for the grammar training by running a k-nearest neighbor (k-NN) filter using n-gram (unigram, bigram and trigram) representations of all Flyspeck theorems in their ambiguous form. Thus the four first phases – k-NN filtering, grammar induction, probabilistic parsing, and HOL typechecking – typically take several seconds, giving real-time feedback to the user. The AI/ATP phase is slow, because for maximal semantic performance it typically runs parallelized (14-CPU) AI/ATP methods selecting relevant premises from the whole Flyspeck, and this needs to be done for a dozen of the most probable and typechecked parse trees. This is however a “mere hardware” issue: If we had a 200-core server rather than the current one, the 16 best parses could be attacked by HOL(y)Hammer in parallel, and the AI/ATP phase would feel much more real-time too. Some screenshots of the service in action are available on our web page.[9]

6 Evaluation on Flyspeck

Once the methods were reasonably scaled up to the whole Flyspeck, we have done a large-scale training/testing evaluation (100-fold cross-validation) on the whole corpus of 22000 theorems. It proceeds as follows:

[8] `http://colo12-c703.uibk.ac.at/hh/parse.html`
[9] `http://colo12-c703.uibk.ac.at/hh/parseimg.html`

1. We create the ambiguous sentences and the disambiguated grammar trees from all 22k Flyspeck theorems as described in Section 3. These sets are permuted randomly and split into 100 equally sized chunks of about 220 trees or sentences. The trees serve for training and the sentences for evaluation.
2. For each testing chunk C_i ($i \in 1..100$) of 220 sentences we take the union of the 99 chunks of grammar trees (altogether about 21800 trees) that correspond to the remaining sentences and build the probabilistic grammar on them - this is fast, taking several seconds. This way we avoid training on the parse trees of the testing sentences.
3. Then we try to get the best 20 parse trees for all the 220 sentences in C using that grammar. This takes on average 4 seconds for each sentence, i.e. the whole parsing takes about 90000 CPU seconds = 25 CPU hours.
4. The parse trees are again transformed into HOL syntax trees, typechecked in HOL, and a single AI/ATP method is run on each typechecked tree for 30 seconds (using Vampire and 128 most relevant Flyspeck premises). This is weaker than using the full HOL(y)Hammer online system, but we cannot afford the 14-fold AI/ATP parallelization due to the number of parse trees.
5. 698549 of the parse trees typecheck (221145 do not), resulting in 302329 distinct (modulo alpha) HOL formulas. These are subjected to ATP, i.e., we run for ca 9000000 CPU seconds = 2500 CPU hours. This is done on a large server with 100-fold parallelization, taking about one day of real time.

We can automatically prove about 70957 (23.5%) of the 302329 typechecked formulas.[10] However, first analysis shows that many of them are provable only because they are parsed incorrectly, for example when the antecedent of an implication becomes trivially false. In this first experiment we do not recognize such cases, however it should not be too difficult to remove such cases with another ATP round that checks for the unsatisfiability of antecedents. Such additional semantic checks could also eventually become a part of the (more tightly integrated) semantic-parsing toolchain.

In 39.4% of the 22000 cases, the HOL formula resulting from one of the sentence's 20 parse trees is alpha-equal to the correct (training) original HOL formula, and its average rank there is 9.34. This is quite encouraging statistics, given that this runs efficiently over whole Flyspeck with quite a high number of introduced ambiguities, and many more sophisticated probabilistic parsing tricks (such as full-scale lexicalization) have not been used yet.

Interestingly, 0.2% of the 22000 cases produce a parse tree that is the same as an existing training tree, but of a different theorem. This means – as can already be seen from the online system parses above – that thanks to the probabilistic behavior the system also (quite necessarily) functions as a conjecture maker. Given a seed of symbols, the system tries to figure out the most probable ways how to give meaning to them, a bit like the Dijkstra's "informal mathematical lot" does. Quite likely one of the many interesting future directions is to evolve one version of the system in such a way that the conjectures are as interesting as possible, using our probabilistic setting to avoid today's brute-force methods.

[10] The exact list is at `http://mizar.cs.ualberta.ca/~mptp/i2f/00proved2`.

References

1. J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. Accepted to Journal of Formalized Reasoning, preprint at `http://www4.in.tum.de/~blanchet/h4qed.pdf`, 2015.
2. E. W. Dijkstra. The fruits of misunderstanding. *Elektronische Rechenanlagen*, 25(6):10–13, 1983.
3. T. Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2012.
4. C. Kaliszyk and J. Urban. Stronger automation for Flyspeck by feature weighting and strategy evolution. In J. C. Blanchette and J. Urban, editors, *PxTP 2013*, volume 14 of *EPiC Series*, pages 87–95. EasyChair, 2013.
5. C. Kaliszyk and J. Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.
6. C. Kaliszyk, J. Urban, J. Vyskocil, and H. Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*, volume 8543 of *Lecture Notes in Computer Science*, pages 435–439. Springer, 2014.
7. D. Kühlwein, T. van Laarhoven, E. Tsivtsivadze, J. Urban, and T. Heskes. Overview and evaluation of premise selection techniques for large theory mathematics. In B. Gramlich, D. Miller, and U. Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 378–392. Springer, 2012.
8. C. Tankink, C. Kaliszyk, J. Urban, and H. Geuvers. Formal mathematics on display: A wiki for Flyspeck. In J. Carette, D. Aspinall, C. Lange, P. Sojka, and W. Windsteiger, editors, *MKM/Calculemus/DML*, volume 7961 of *Lecture Notes in Computer Science*, pages 152–167. Springer, 2013.
9. D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.
10. C. Zinn. *Understanding informal mathematical discourse*. PhD thesis, University of Erlangen-Nuremberg, 2004.