From Tax Compliance in Natural Language to Executable Calculations: Combining Lexical-grammar-based Parsing and Machine Learning

Esme Manandise, Conrad de Peuter, Saikat Mukherjee
Intuit, 2700 Coast Ave., Mountain View
CA, 94043-1140. United States
{Esme_Manandise,Conrad_DePeuter,Saikat_Mukherjee}@intuit.com

Abstract

Regulatory agencies publish tax-compliance content written in natural language intended for human consumption. There has been very little work on automated methods for interpreting this content and for generating executable calculations from it. In this paper, we describe a combination of lexical grammar-based parsing with encoder-decoder architectures for automatically bootstrapping executable calculations from natural language. The combination is particularly suitable for domains such as compliance where training data is scarce and accuracy of interpretation is of high importance. We provide an overview of the implementation for North American income-tax forms.

Introduction

Tax compliance is an universal task for individuals and businesses across countries. The complexity of the regulations and the extent of services provided by regulatory agencies vary across jurisdictions. When the responsibility of compliance is on the individual or the business, the regulatory agencies typically publish forms and instructions in natural language. Individuals and businesses are expected to interpret and file the filled-in forms with the agencies.

To simplify the process for tax filers, tax-preparation applications have been created which abstract filers away from the language of forms. Historically, the compliance logic in these applications have been built with the help of tax experts who, manually, interpret the compliance regulations in the domain and write them down in code. This paper gives an overview of an automated hybrid approach for converting natural-language tax calculations to executable calculations. The approach combines rule-based parsing and machine learning.

Our tax domain is the set of tax forms published annually by the Internal Revenue Service (IRS) and the Canada Revenue Agency (CRA). These are the primary tax regulatory agencies in North America. The IRS, for example, publishes more than 5,000 forms (excluding related worksheets, publications and instructions).

Tax forms describe calculations of varying complexity (addition, subtraction, multiplication, division, percentage conversion, rounding) (Table 1).

Recent advances in language processing, especially with pre-trained deep learning models, hold the promise of high accuracy for generating executable calculations from raw text. However, even though certain aspects of these models are pre-trained, we still need to provide training to create the desired calculation output. It is difficult to acquire training examples in low-data domains such as tax compliance. Hence, in our in-house approach we leverage both grammar-based parsing as well as supervised machine learning to generate calculations automatically. The corpus-driven grammar-based approach generates calculations in the absence of any training data and also seeds the training for the learning-based approach.

Related Work

There are few publications in English that detail the language and discourse of the tax-and-regulations domain and none that describes an end-to-end framework to translate textual calculations into executable code.

Glossaries of tax terms are common and relatively small in size. They are made available online and/or are published by government agencies, private outfits and international organizations; some glossaries are integrated in tax and accounting software.

In recent years, there has been an interest in studying tax-domain texts from the linguistic and automation perspectives. Wang et al. (2015) describe SEMPRE, semantic parsing with execution. It has promising functionality to learn to map natural language utterances to denotations via intermediate logical forms. Manandise (2019) describes a preprocessor to annotate automatically raw text in the tax domain with linguistic and domain features extracted using statistical measures based on corpus analysis. Blank and Osofsky (2017) discuss how language simplicity can cause a loss of information and make content less accurate.

Tax NLP Calculation Workflow

The NLP calculation workflow comprises several sequential steps. The source inputs are agency compliance documents, typically in PDF format. The outputs of the workflow are calculations which can be executed by rule engines.

IRS Forms	Raw Input Segment
F8829	Line C times line D divided by 12 times $5.00 times line E
F1041	If line 25 is larger than the total of lines 23 and 26, enter amount overpaid
F8941WKS	If the result is not a multiple of $1,000, round the result down to the next lowest multiple of $1,000

Table 1: Calculations as Raw Text

Content Structuring

Content structuring builds the foundation for the natural-language to executable-calculations workflow. We automatically extract information from compliance forms, which are in PDF format, and represent the information in a structured, machine-readable format. We use machine-learning models which learn to identify and extract structured information from PDF forms such as line numbers, line descriptions, tables, checkboxes, etc.

Content structuring from PDFs can extract 518 form sections with 95% accuracy, 5,573 form line descriptions with 92% accuracy and 7,930 form fields with 85% accuracy.

Term Extraction

We use a mixture of collocation scores for noun-phrase identification. On a corpus of 82.9k sentences from IRS income tax forms and 187.6k sentences from associated instruction documents, we extract 13k terms which are subsequently used by our in-house *LeanParsing*.

Data Model Matching

Regulatory agencies provide schemata to describe fields in compliance forms. For instance, the IRS provides an XML data model of some 3000 elements, known as the Modern E-file Format (MeF), according to which tax returns are structured. These data models are nomenclature of fields of forms and not a structured representation of the forms themselves. In order to generate executable calculations from a predicate-argument structure (PAS), we map the predicate structures generated by our parser to the data model of the domain.

Consider raw text in 1 below from a IRS line and its associated PAS:

1. Add your taxable disability income to your spouse's taxable disability income.
2. add(
   taxable_disability_income(taxpayer),
   taxable_disability_income(spouse)
   ))

Converting 2 above into an executable calculation necessitates mapping the predicate structures *taxable_disability_income(taxpayer)* and *taxable_disability_income(spouse)* to the corresponding elements in the MeF data model. We use word embeddings and supervised classification. The word embedding, trained from the complete domain corpus, finds top candidates entities for a given predicate structure. A random forest supervised model, trained on pairs of predicate structures to data model elements, predicts whether a given candidate text-PAS pair is a match. The most confident model prediction is returned.

Parsing

LeanParsing (LP)

LeanParsing is an integrated application combining a goal-oriented rule-based parser and translator to output minimalist structured representations of tax-related calculations expressed in raw text written in English. LP represents the operations (operators and operands) in a simplified logical form (PAS for predicate-argument structure) as executable content for consumption by downstream components (Figure 1).

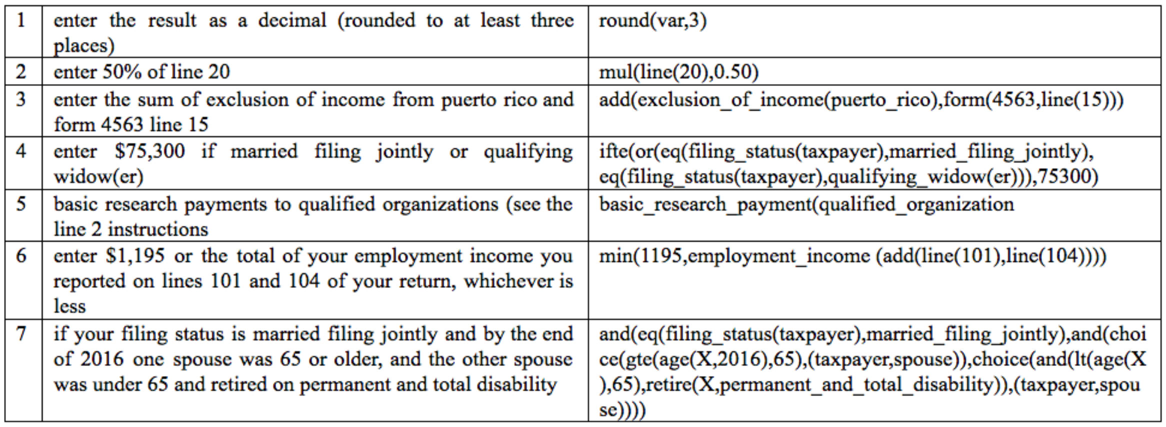
The chart parser includes look-ahead and backtracking features. It combines top-down and bottom-up mechanisms for constituent bracketing and analysis. Parsing and translation are governed by lexicons, grammar-based procedures with discourse mechanisms to allow for interpretation and disambiguation of operators and operands in context beyond sentence boundaries.

The motivations for adopting a rule/procedure-based paradigm for parsing and translation are:

1. narrow deterministic goal
2. small domain corpus (raw-text collection)

LP is tasked with interpreting a specific language within the tax-domain, namely, input that expresses calculations in plain English across agency forms. Consequently, tax-domain text that fulfills other communication functions, once recognized, is ignored. LP simplifies the task by separating text that points to calculations from non-calculations. Tax forms are written for a wide audience—from general public to tax experts. To help with reading comprehension, tax calculations in texts are augmented with descriptive and contextual content. This added content is repetitive, deictic, and often redundant. For instance, in row 5 of Figure 1, the parenthetical material *(see the line 2 instructions)* is not relevant to the interpretation of the calculation which consists of a term operand *basic_research_payment* (to be bound to the actual amounts associated with the term) for the actual organization denoted by the term *qualified_organization*. On the other hand, in row 1, the parenthetical material cannot be discarded as it is relevant to the calculation. In row 6, the expressions *you reported* and *of your return* are redundant (as the tax-filer must be aware of her/his filling activity).

Lexical Resources A base lexicon for single tokens and a terminology (term lexicon) for multi-word expressions corresponding to tax concepts and entities are populated automatically by mining IRS and CRA income-tax forms, schedules, worksheets, and publications. The lexical expressions



1	enter the result as a decimal (rounded to at least three places)	round(var,3)
2	enter 50% of line 20	mul(line(20),0.50)
3	enter the sum of exclusion of income from puerto rico and form 4563 line 15	add(exclusion_of_income(puerto_rico),form(4563,line(15)))
4	enter $75,300 if married filing jointly or qualifying widow(er)	ifte(or(eq(filing_status(taxpayer),married_filing_jointly), eq(filing_status(taxpayer),qualifying_widow(er))),75300)
5	basic research payments to qualified organizations (see the line 2 instructions	basic_research_payment(qualified_organization
6	enter $1,195 or the total of your employment income you reported on lines 101 and 104 of your return, whichever is less	min(1195,employment_income (add(line(101),line(104))))
7	if your filing status is married filing jointly and by the end of 2016 one spouse was 65 or older, and the other spouse was under 65 and retired on permanent and total disability	and(eq(filing_status(taxpayer),married_filing_jointly),and(choice(gte(age(X,2016),65),(taxpayer,spouse)),choice(and(lt(age(X),65),retire(X,permanent_and_total_disability)),(taxpayer,spouse))))

Figure 1: Raw Segment and PAS Pairs

result from co-occurrence/collocation-based surface statistical measures. In addition, linguistic filters exclude terms that are ill-formed.

The base lexicon is a repository of granular multimodal knowledge about single tokens in the domain. These tokens correspond to the head of terms. For instance, the single-token concept *expense* is the head of the terms *daycare expense* or *research and experimental expense*.

Lexical entries have been designed as *dict* pairs *key:values*. The values themselves can be of type *key:values*. The *values* fields are augmented with Wordnet and in-house-Wordnet-like features to describe granular morphological, semantic, syntactic, and domain-idiosyncratic properties of the keys.

Parsing and Translation Overview LP combines both shallow parsing and chart deep parsing (wherein each token is accounted for). However, if calculations can be built before all the tokens in input are consumed, LP closes early. Early closure is enabled by look-ahead and backtracking mechanisms.

The main algorithmic tasks for LP are:

1. tokenization
2. morphological analysis (base forms)
3. single-token matching against base-lexicon with creation of temporary *dict* for the input tokens; populate *dict* with all available knowledge about tokens from base lexicon
4. term matching against terminology with creation of temporary *dict*; populate *dict* with all available knowledge about terms from terminology and ontology
5. on-the-fly creation of *dict* for unknown tokens with default predicted values
6. pre-processing tasks (text normalization, language-variant unification, abbreviations, acronyms)
7. detection of candidate calculations (partial or complete)
8. noise tagging of expressions (single tokens, multi-tokens, phrasal expressions) estimated to be unrelated to calculations
9. calculation classification (arithmetic, amount, etc)
10. first try shallow parsing (possible early closure) of labeled calculations
11. if early closure fails, do deep parsing of detected calculations.
12. translation of intermediate parse charts into intermediate PAS representations to final representations

Notes Rule-based LP does not include a separate declarative grammar; instead, the grammar consists of a set of rules and recursive procedures of type *if X then Y*, which are written either as regular expressions and/or encoded in Python. In its simplest form, LP includes rules that span the input and test for possible patterns expressed as co-occurrences of parts of speech and/or semantic, morphological and syntactic features. This rule type is the most declarative in nature. If shallow parsing does not produce a PAS, LP runs in its full deep-parsing mode (chart parsing).

Evaluation of Raw Segment and PAS Pairs Out of 7,589 calculations, LP generates 5,584 accurate calculations in PAS format, thus achieving 73% coverage with no-human in the loop.

Error Checks

Architecture Our in-house *ColBERT* is a supervised machine learning framework to extract tax calculations from agency content. The framework takes pairs of English-language tax laws, and calculation graphs describing those laws, and learns to output calculation graphs given PAS or English-language sentences. *ColBERT* uses an encoder-decoder architecture and is trained off of both human-written examples and LP-created examples.

The encoder component converts text in a source language to vectors, and the decoder converts these vectors into tokens in a target language. Our ColBERT's source language is English, and its target language is the complete set of operations and operands possible within our internal tax compliance language. For an encoder, we used BERT, a pre-trained encoder released by Google. As our target language was internal, the decoder was trained from scratch.

In our training examples, we introduced a token to represent *the result of the previous calculation*: DATATEMP. When processing outputted sequences, we split at known operators, and put the full operations associated with these operands into a stack. When the DATATEMP token appeared, we popped the last operation off the stack. This created edges in the calculation graph from one example to another. When the output of the network is multiple boolean statements, we assumed a logical conjunction of the Boolean statements.

Training Data and Training To train ColBERT, we required pairs of English-language tax laws and verified calculations graphs representing those laws. From the LP pipeline and some manually written examples from internal tax analysts, we had roughly 3000 examples.

Error Calculations A major use case of *ColBERT* was in coding the calculations required for CRA error specifications. These specifications outline conditions which invalidate a tax return. Although they are not included within individual forms, it is essential that tax software perform these calculations. The calculations on these documents are more complex than typical form-based calculations, with many operations in individual sentences. As LP was built to parse lines of forms, which typically have at most two operations in an individual text input, it was not well suited to parse the error specifications.

Within the error specifications, there were many examples of the same operation sequence with different operands. Our synthetic data creation schema allowed for a workflow where an analyst would write a ground truth example for one of these sequences and the network would learn this sequence and output the correct calculation graph for all sequences of this structure.

Evaluation As the output of the network was an internal compliance language, we were not able to evaluate this network on external datasets. Internally, the network was successful in saving a significant amount of human effort. By generating calculations through the supervised framework and then manually correcting errors, as opposed to authoring all content from scratch, analysts were able to speed up their work by 16x on average. A key discovery to this speedup was that the network output did not need to be perfect. The process of reviewing automatically produced calculations and their associated metadata, and manually correcting when necessary was significantly faster than authoring from scratch. When repeated identical errors were discovered, the analyst would create a single correction and the network would be retrained. In our final iteration, 40% of error calculations (which comprise of many individual calculations) were correctly generated in their automated form, and 60% required some level of manual intervention.

Conclusion

In this paper, we provided an overview for combining lexical grammar and deep learning architectures for bootstrapping executable calculations from compliance forms written in natural language. The system leverages output from grammar-based techniques to train the neural network, thereby achieving coverage over descriptions in form lines as well as more error check calculation descriptions. Future directions include exploring semantic parsing techniques as well as interpreting set operations, which are fairly common in tax forms. Also, we want to evaluate the feasibility of the approaches on non-English jurisdictions (e.g. Quebec in Canada) and non-tax compliance domains.

Acknowledgments

For contributions, insight and discussions, we thank Anu Sreepathy, Mritunjay Kumar and Anu Singh.

References

An, Y.; and Wilson, N. 2016. Tax Knowledge Adventure: Ontologies that Analyze Corporate Tax Transactions. In *Proceedings of the 17th International Digital Government Research Conference on Digital Government Research*, 6:303–311

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *arXiv:1810.04805 [cs.CL]*

Blank, J.; and Osofsky, L. Simplexity: Plain Language and the Tax Law. 2017. In *Emory Law Journal*, 66:189

Cohen, S. 2006. Words! Words! Words!: Teaching the Language of Tax. In *Journal of Legal Education*, 55

Curtotti, M. and McCreath, E. 2011. A corpus of Australian contract language: Description, profiling and analysis. in *Proceedings of the International Conference on Artificial Intelligence and Law*. 6:199-208

Distinto, I.; Guarino, N.; and Masolo, C. 2013. A well-founded ontological framework for modeling personal income tax. In *Proceedings of the International Conference on Artificial Intelligence and Law*. 6:33-42

Manandise, E. 2019. Towards Unlocking the Narrative of the United States Income Tax Forms. In *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)*, 33-41. Association for Computational Linguistics

Manandise, E.; and de Peuter, C. 2020. Mitigating Silence in Compliance Terminology during Parsing of Utterances. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation.*, 204-212. Association for Computational Linguistics

Wang, Y.; and Berant, J.; and Liang, P. 2015. Building a Semantic Parser Overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing. China, 1332-1342. Association for Computational Linguistics