POTATO: exPlainable infOrmation exTrAcTion framewOrk

Ádám Kovács
adam.kovacs@tuwien.ac.at
TU Wien
Vienna, Austria
Budapest University of Technology and Economics
Budapest, Hungary

Kinga Gémes
kinga.gemes@tuwien.ac.at
TU Wien
Vienna, Austria
Budapest University of Technology and Economics
Budapest, Hungary

Eszter Iklódi
eszter.iklodi@tuwien.ac.at
TU Wien
Vienna, Austria

Gábor Recski
gabor.recski@tuwien.ac.at
TU Wien
Vienna, Austria

ABSTRACT

We present POTATO, a task- and language-independent framework for human-in-the-loop (HITL) learning of rule-based text classifiers using graph-based features. POTATO handles any type of directed graph and supports parsing text into Abstract Meaning Representations (AMR), Universal Dependencies (UD), and 4lang semantic graphs. A web-based user interface allows users to build rule systems from graph patterns, provides real-time evaluation based on ground truth data, and suggests rules by ranking graph features using interpretable machine learning models. Users can also provide patterns over graphs using regular expressions, and POTATO can recommend refinements of such rules. POTATO is applied in projects across domains and languages, including classification tasks on German legal text and English social media data. All components of our system are written in Python, can be installed via pip, and are released under an MIT License on GitHub.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**; *Rule learning*.

KEYWORDS

exPlainable, explainability, HITL

1 INTRODUCTION

Recent natural language processing (NLP) solutions achieving state-of-the-art results on public benchmarks rely on deep learning models with millions of parameters. Since such models require large amounts of training data, offer little or no explainability of their

decisions, and pose a risk of learning unintended bias, their applicability in real-world scenarios is limited. Rule-based systems may provide accurate and transparent solutions, but can be difficult to build and maintain. POTATO is a framework that supports the semi-automatic creation of rule-based text classifiers. Firstly, rules can be formulated as patterns over graphs representing the syntax and/or semantics of the input text. Secondly, ground truth labels are used to suggest rules to the users which they may choose to accept with or without modifications. POTATO can also recommend refinements of underspecified graph patterns, allowing the user to guide the rule learning process. In addition to a framework for the automatic extraction and ranking of graph-based features, POTATO also provides an intuitive, high-level user interface that allows users without technical expertise to construct graph-based rule systems using automatic suggestions and to evaluate rules against ground truth data in real time. We also provide a REST API to our backend for deploying rule-based solutions built in POTATO. All components of our system are released under an MIT license and are available from GitHub[1] and via pip, by installing the xpotato[2] package. A short video demonstration is available on Youtube[3]. The rest of this paper is structured as follows. Section 2 reviews recent systems for rule learning and HITL learning. Section 3 presents our method for generating and ranking graph-based features from labeled datasets. Section 4 presents our web-based user interface for HITL learning, inspection, editing, and evaluation of rule systems. An overview of the system architecture and workflow is provided in Section 5. In Section 6 we showcase recent applications of our system across domains and languages.

2 RELATED WORK

Our work presents a semi-automatic human-in-the-loop (HITL) method for the learning of rule systems for text classification tasks using patterns over graph representations of natural language input. The graphs we use for representing the syntax and semantics of text include Universal Dependencies [UD, 19], Abstract Meaning Representations [AMR, 1], and 4lang concept graphs [13]. Recent approaches to rule learning for text classification include the learning of first-order logic formulae over semantic representations using neural networks [8, 27] and integer programming [6]. For

---

[1] https://github.com/adaamko/POTATO
[2] https://pypi.org/project/xpotato/
[3] https://youtu.be/79Q7UnLsMFc

suggesting rules to users based on ground truth labels we rely on interpretable machine learning models such as decision trees. Other rule-based approaches to text classification may involve the use of textual patterns [15], semantic structures [28], or a combination of regular expressions and graph patterns [30]. Hybrid approaches to text classification include the incorporation of lexical features into DL architectures [14, 20] and voting between rule-based and ML systems [10, 24]. The systems most similar to POTATO are the HEIDL [28] and GrASP [15, 29] libraries, both of which support pattern-based text classification with automatic suggestions. Our tool differs from these in its use of syntactic and semantic graphs to represent text input, its ability to suggest refinements of rules specified by the user, and functionalities for working with data that has little or no ground truth annotation available. We shall describe all features of our system in detail in Sections 3 and 4.

3 METHOD

In this section we present the core functionalities of POTATO. As a running example throughout the section we shall use the relation extraction task defined for the 2010 Semeval task *Multi-Way Classification of Semantic Relations Between Pairs of Nominals* [12], In this task, sentences containing two entities identified by the labels `Entity1` and `Entity2` must be classified based on the semantic relationship that holds between those entities. In our example we focus on the most frequent label in the dataset, `Entity-Destination` (ED), which is defined as follows: *"An entity is moving towards a destination. Example: the boy went to bed"* [12, p.34]. The training data for this task contains 8000 sentences, 844 of which are labeled as examples of ED. We used 80% of this dataset for developing our rule system and the remaining 20% for validation. The rules we construct using POTATO are patterns over 4lang graph representations of the sentences, we construct these graphs using the implementation of `text_to_4lang` [25] in the `tuw_nlp`[4] library.

For each class label, POTATO enables the construction of a list of rules that are used to label sentences as belonging to that class if and only if at least one rule applies to the graph representation of the sentence. A single rule may be a pattern over a graph or a conjunction of multiple patterns, some of which may be negated. A single pattern is a graph that matches the input graph if and only if the pattern graph is a subgraph of the input graph. Graph nodes and edges may have string labels, and pattern graphs may contain regular expressions (regexes) as both edge and node labels. If a pattern graph contains regex labels, then they match an input graph if and only if the pattern graph is contained in the input graph such that the regex labels of the pattern graph match the string labels of corresponding nodes and edges in the input graph. Since a rule may be a conjunction of patterns, each of which may be negated, and a rule system is a disjunction over rules, a complete rule system can be considered the disjunctive normal form (DNF) of a single boolean formula whose predicates are the individual graph patterns.

POTATO provides rule suggestions by training interpretable machine learning models using graph features. The features extracted from each graph are its connected subgraphs of at most n edges. In most applications we set n to 2. Graphs consisting of a single

node are included for any value of n, since they are connected graphs with 0 edges. A training dataset of labeled texts that are each represented by graphs can then be used to train interpretable ML models and rank subgraphs based on their feature importance. The core assumption of our method is that subgraphs with high feature importance are the ones that should be suggested to the user constructing a rule system. Currently POTATO trains decision trees as implemented by the library `scikit-learn` [21] and ranks subgraphs based on their Gini coefficients. The system also includes the option of ranking subgraphs without training any ML model, by counting the number of positive and negative examples in the training data that contain each subgraph as a feature and calculating $TP - FP$, i.e. the difference between the number of true positive and false positive decisions one would make by classifying input sentences based on the presence of this pattern only. Top-ranked subgraphs are presented to the user, who can choose to incorporate them in the rule system, with or without manual modifications.

Using the Semeval relation extraction dataset for training, the top suggestion without manual modifications would be a graph containing the single edge $into \xrightarrow{2} ENTITY2$ (edges labeled 1 and 2 in 4lang graphs mark the first and second arguments of binary relations). This rule in itself achieves 76.2% precision and 62.8% recall on the training data, retrieving 407 true positive and 127 false negative samples. A human expert might extend this pattern by constructing the graph $ENTITY2 \xleftarrow{2} into \xrightarrow{1} (.^*) \xrightarrow{2} ENTITY1$, which requires that *ENTITY1* is the second argument of the concept that is the first argument of the *into* relation. POTATO uses the PENMAN notation [11] to represent graphs on its frontend, the original graph would be displayed as (u_1 / into :2 (u_2 / entity2)) and the modified graph would be entered by the user as (u_1 / into :2 (u_2 / entity2) :1 (u_3 / .* :2 (u_4 / entity1))). This modified graph now achieves 85.3% precision and 41.0% recall for the ED label, by matching 266 true positives (such as the pair of entities in the sentence *We have dumped the **spam** into the **junk folder***) and 46 false positives (such as *The **lungs** are divided into **lobes***). The ranking of subgraphs by feature importance can also be used to suggest refinements of a subgraph containing regular expressions. If the user creates or modifies a rule so that some label contains a regular expressions and asks the system to *refine* this rule, POTATO will evaluate matching subgraphs and replace the regex with a disjunction of all string labels with which the pattern would achieve over 90% precision (and non-zero recall) on the training dataset. Refining the regex node of the previous rule would yield a pattern that would match 138 true positives and only 1 false positive (*A US **aircraft** was dropped into a difficult **landing** in Mali*), achieving 99.3% precision on the training dataset. On the validation data the same pattern achieves 95.6% precision and 21.9% recall[5].

4 USER INTERFACE

Figure 1 illustrates the human-in-the-loop (HITL) workflow enabled by our application. The prerequisite for launching the HITL user interface is to load a dataset as a set of labeled or unlabeled graphs. Any directed graph can be loaded, we provide interfaces for UD

---

[4]https://pypi.org/project/tuw-nlp/

[5]the list of concepts constructed by POTATO for this rule consists of these 16 words: *misplace, invest, drop, import, transport, fetch, pack, insert, pour, implant, remove, leak, add, dump, release, stuff*

parsers in `stanza`, two AMR parsers for English and German, and our own language-agnostic 4lang parser. Suggesting and evaluating rules requires ground truth labels, if these are not available, the UI can be launched in *advanced* mode for bootstrapping labels using rules. Once a dataset is loaded, the HITL frontend can be started and the user is presented with the interface shown in Figure 2, built using the `streamlit` library[6].
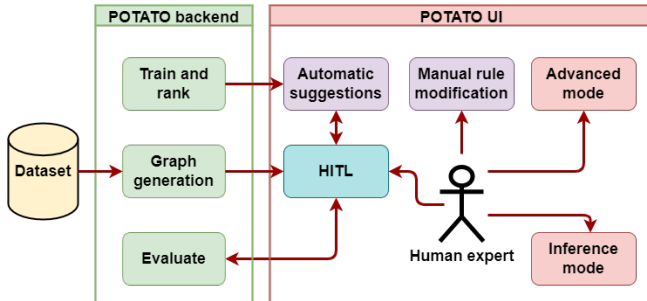

Figure 1: System workflow of POTATO

The dataset browser ① allows the user to view the text, graph, and label for all rows of the dataset. The viewer renders graphs using the `graphviz` library [9], and also provides the PENMAN notation that can be copied by the user for quick editing of rules. Users can choose the class to work on ②, the list of rules constructed for each class are maintained in a list ③. Rules can be viewed and evaluated on the training and validation datasets ④, and users can analyze correct and incorrect predictions of each rule by choosing to view true positive, false positive, or false negative examples ⑤. Functions not visible on Figure 2 due to lack of space include the button *suggest new rules*, which returns a list of suggested graphs together with their performance on the training data, allowing the user to select those that should be added to the rule list. For rules containing regular expressions, the *Refine* button will replace regular expressions with a disjunction of high-precision labels, as described in Section 3. Rules are automatically saved and can be reloaded. Rules then also can be used for inference (from the UI, from the command line, or via a REST API).

5 SYSTEM ARCHITECTURE

All software is released via two python packages (`xpotato`, `tuw-nlp`), both installable via pip. The frontend of the system is included as a separate module of the `xpotato` library. In this section we describe each component of our system.

tuw-nlp. The tuw-nlp package contains a suite of generic NLP tools and includes the `graph` module for building and manipulating linguistic graphs. The module uses networkx[7] for its graph data structures, and implements the graph operations used by POTATO's backend, including the generation of subgraphs and the pattern matching on graphs for the evaluation and ranking of graph patterns. The pattern matcher

class customizes and wraps the `DiGraphMatcher` class from `networkx.algorithms.isomorphism`, which implements the vf2 algorithm [4]. The `tuw_nlp.graph` module also provides interfaces for working with AMR graphs (English or German), UD graphs, and 4lang semantic graphs. For English AMR-parsing, the module interfaces with a pretrained Transformer-based AMR parser [23] via the `amrlib`[8] library, German AMRs are constructed using a multilingual, transition-based system [5] implemented by the `amr-eager-multilingual`[9] library. For Universal Dependency (UD) parsing, `tuw-nlp` relies on the `stanza` package [22], and 4lang semantic graphs are built from UD trees using a reimplementation of the `text_to_4lang` tool [25].

xpotato. The backend of POTATO in the main module of the `xpotato` package implements core functionalities by interfacing with the `tuw_nlp.graph` module described above, the `scikit-learn` library [21] for training and inspecting decision trees and the `scikit-criteria` package [2] for feature ranking. The module also implements the algorithm behind the *refine* mechanism (see Section 3) for suggesting refinements of graph patterns containing regular expressions. The frontend of POTATO is a web application implemented using `streamlit` that exposes all features of the main `xpotato` module and enables the worflow described in Section 4. The web application can be started in three modes. The *simple* mode corresponds to the standard scenario when sufficient annotated training data is available for the rule suggestion mechanism. The *advanced* mode contains an additional frontend module for annotating unlabeled data supported by using graph patterns. Launching the application in *inference* mode loads a saved list of rules and allows the user to apply the rule set to raw text. Inference can also be run as a REST API using a command-line tool, the service uses the FastAPI[10] library.

6 APPLICATIONS

POTATO is language- and domain-agnostic. Besides the English relation extraction task it has been used for the classification of German legal documents and for offensive text detection in English and German social media. In this section we describe these applications briefly. For the task of relation extraction we have developed a simple rule-based solution not only for the Semeval dataset introduced in Section 3, but also a larger corpus for medical relation extraction called CrowdTruth[11] [7]. This benchmark contains two relations, *treat* and *cause*. For the *treat* relation, state-of-the-art deep learning systems achieve F1-scores up to 0.9 [3]. For a simple rule-based approach we used POTATO to build patterns over UD graphs, and using only 12 rules we achieved precision and recall values of 0.91 and 0.32, respectively, an example of a rudimentary but high-precision and transparent solution. The rule system is available on GitHub[12].

We also used POTATO to develop rule systems for the 2019-2021 HASOC datasets on offensive text detection, [16–18]. The rules for English were part of our submission to the 2021 HASOC shared task

---

Figure 2: The main page of POTATO allows the user to ⓵ browse the dataset and view the processed graphs ⓶ choose the class you want to build rule-based systems on ⓷ modify, delete, add new rules and get suggestions ⓸ view the results of the selected rules ⓹ view example predictions for each rule

[10], where they were used to increase the recall and F1 score of our top-performing deep learning models in an ensemble solution. On the German HASOC dataset we achieve precision and recall values of 0.92 and 0.28 with just 8 graph patterns, these results are currently under review, the rule-based solutions are available on GitHub[13].

Finally, POTATO is applied in the BRISE[14] project for digitalizing the building permit process of the City of Vienna and involves extracting formal rules from text documents of the city's Zoning Plan. A crucial first step of the implemented pipeline [26] is the multi-label classification of sentences based on the formal attributes of the domain that they mention. For example, the sentence *Flachdächer bis zu einer Dachneigung von fünf Grad sind (…) zu begrünen.* 'Flat roofs with a pitch not exceeding 5 degrees must be greened (…)' must be classified as mentioning three attributes, `BegruenungDach` 'greening of roofs', `Dachart` 'roof type', and `DachneigungMax` 'maximum roof pitch'. While the total number of attributes is close to a hundred, the 20 most frequent ones cover nearly 75% of all labels and the top 10 cover more than 50%. The latest rule-based solution

to this task, available publicly on GitHub[15] and to be published in a forthcoming paper, has been implemented using `text_to_4lang` and POTATO and on the 10 most frequent attributes achieves class-level precision values above 0.8 and up to 1.0, class-level recall values between 0.5 and 0.95, and overall precision and recall scores of 0.9 and 0.45, respectively. Early experiments show that similar results can be obtained using standard machine learning algorithms, but the rule system offers a fully transparent, configurable, and auditable solution.

ACKNOWLEDGMENTS

REFERENCES

[1] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, 178–186. https://www.aclweb.org/anthology/W13-2322

---

[13]https://github.com/adaamko/POTATO/tree/main/features/hasoc
[14]https://smartcity.wien.gv.at/en/brise/

---

[15]https://github.com/recski/brise-plandok

[2] Juan B Cabral, Nadia Ayelen Luczywo, and José Luis Zanazzi. 2016. Scikit-Criteria: Colección de métodos de análisis multi-criterio integrado al stack científico de Python. In *XLV Jornadas Argentinas de Informática e Investigación Operativa (45JAIIO)- XIV Simposio Argentino de Investigación Operativa (SIO) (Buenos Aires, 2016)*. Sociedad Argentina de Informática e Investigación Operativa (SADIO), Buenos Aires, Argentina, 59–66. http://45jaiio.sadio.org.ar/sites/default/files/Sio-23.pdf

[3] Gjorgjina Cenikj, Tome Eftimov, and Barbara Koroušić Seljak. 2021. SAFFRON: tranSfer leArning For Food-disease RelatiOn extractioN. In *Proceedings of the 20th Workshop on Biomedical Language Processing*. Association for Computational Linguistics, Online, 30–40. https://doi.org/10.18653/v1/2021.bionlp-1.4

[4] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. 2001. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*. Springer-Verlag, Ischia, Italy, 149–159.

[5] Marco Damonte and Shay B. Cohen. 2018. Cross-Lingual Abstract Meaning Representation Parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1146–1155. https://doi.org/10.18653/v1/N18-1104

[6] Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. 2018. Boolean Decision Rules via Column Generation. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., Montréal, Canada. https://proceedings.neurips.cc/paper/2018/file/743394beff4b1282ba735e5e3723ed74-Paper.pdf

[7] Anca Dumitrache, Lora Aroyo, and Chris Welty. 2017. Crowdsourcing Ground Truth for Medical Relation Extraction. *ACM Transactions on Interactive Intelligent Systems* 8 (01 2017). https://doi.org/10.1145/3152889

[8] Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61 (2018), 1–64. https://dl.acm.org/doi/10.5555/3241691.3241692

[9] Emden R. Gansner and Stephen C. North. 2000. An open graph visualization system and its applications to software engineering. *SOFTWARE - PRACTICE AND EXPERIENCE* 30, 11 (2000), 1203–1233.

[10] Kinga Gémes, Ádám Kovács, Marcus Reichel, and Gábor Recski. 2021. Offensive text detection on English Twitter with deep learning models and rule-based systems. (Dec. 2021). https://hlt.bme.hu/media/pdf/hasoc2021.pdf to appear in FIRE 2021: Forum for Information Retrieval Evaluation.

[11] Michael Wayne Goodman. 2020. Penman: An Open-Source Library and Tool for AMR Graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Online, 312–319. https://www.aclweb.org/anthology/2020.acl-demos.35

[12] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Uppsala, Sweden, 33–38. https://aclanthology.org/S10-1006

[13] András Kornai. 2019. *Semantics*. Springer Verlag, Cham, Switzerland. http://kornai.com/Drafts/sem.pdf

[14] Anna Koufakou, Endang Wahyu Pamungkas, Valerio Basile, and Viviana Patti. 2020. HurtBERT: Incorporating Lexical Features with BERT for the Detection of Abusive Language. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*. Association for Computational Linguistics, Online, 34–43. https://doi.org/10.18653/v1/2020.alw-1.5

[15] Piyawat Lertvittayakumjorn, Leshem Choshen, Eyal Shnarch, and Francesca Toni. 2021. GrASP: A Library for Extracting and Exploring Human-Interpretable Textual Patterns. https://arxiv.org/abs/2104.03958. arXiv:2104.03958 [cs.CL]

[16] Thomas Mandl, Sandip Modha, Anand Kumar M, and Bharathi Raja Chakravarthi. 2020. Overview of the HASOC Track at FIRE 2020: Hate Speech and Offensive Language Identification in Tamil, Malayalam, Hindi, English and German. In *Forum for Information Retrieval Evaluation* (Hyderabad, India) *(FIRE 2020)*. Association for Computing Machinery, New York, NY, USA, 29–32. https://doi.org/10.1145/3441501.3441517

[17] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Mandlia Chintak, and Aditya Patel. 2019. Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation* (Kolkata, India) *(FIRE '19)*. Association for Computing Machinery, New York, NY, USA, 14–17. https://doi.org/10.1145/3368567.3368584

[18] Thomas Mandl, Sandip Modha, Gautam Kishore Shahi, Hiren Madhu, Shrey Satapara, Prasenjit Majumder, Johannes Schäfer, Tharindu Ranasinghe, Marcos Zampieri, Durgesh Nandini, and Amit Kumar Jaiswal. 2021. Overview of the HASOC subtrack at FIRE 2021: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages. In *Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation (FIRE 2021)*. Association for Computing Machinery, New York, NY, USA, 1–3. https://doi.org/10.1145/3503162.3503176

[19] Joakim Nivre, Mitchell Abrams, Željko Agić, et al. 2018. Universal Dependencies 2.3. http://hdl.handle.net/11234/1-2895 LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[20] Endang Wahyu Pamungkas and Viviana Patti. 2019. Cross-domain and Cross-lingual Abusive Language Detection: A Hybrid Approach with Deep Learning and a Multilingual Lexicon. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Association for Computational Linguistics, Florence, Italy, 363–370. https://doi.org/10.18653/v1/P19-2051

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Online, 101–108. https://doi.org/10.18653/v1/2020.acl-demos.14

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs.LG]

[24] Amir Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive Language Detection Using Multi-level Classification. In *Advances in Artificial Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 16–27. https://doi.org/10.1007/978-3-642-13059-5_5

[25] Gábor Recski. 2018. Building concept definitions from explanatory dictionaries. *International Journal of Lexicography* 31 (2018), 274–311. Issue 3. https://doi.org/10.1093/ijl/ecx007

[26] Gabor Recski, Björn Lellmann, Adam Kovács, and Allan Hanbury. 2021. Explainable Rule Extraction via Semantic Graphs. In *Proceedings of the Fifth Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2021)*. CEUR Workshop Proceedings, São Paulo, Brazil, 24–35. http://ceur-ws.org/Vol-2888/paper3.pdf

[27] Prithviraj Sen, Marina Danilevsky, Yunyao Li, Siddhartha Brahma, Matthias Boehm, Laura Chiticariu, and Rajasekar Krishnamurthy. 2020. Learning Explainable Linguistic Expressions with Neural Inductive Logic Programming for Sentence Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4211–4221. https://doi.org/10.18653/v1/2020.emnlp-main.345

[28] Prithviraj Sen, Yunyao Li, Eser Kandogan, Yiwei Yang, and Walter Lasecki. 2019. HEIDL: Learning Linguistic Expressions with Deep Learning and Human-in-the-Loop. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Florence, Italy, 135–140. https://doi.org/10.18653/v1/P19-3023

[29] Eyal Shnarch, Ran Levy, Vikas Raykar, and Noam Slonim. 2017. GRASP: Rich Patterns for Argumentation Mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1345–1350. https://doi.org/10.18653/v1/D17-1140

[30] Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Dane Bell. 2020. Odinson: A Fast Rule-based Information Extraction Framework. In *Proceedings of the 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 2183–2191. https://aclanthology.org/2020.lrec-1.267