POTATO: exPlainable infOrmation exTrAcTion framewOrk

Ádám Kovács^{1,2}, Kinga Gémes^{1,2}, Eszter Iklódi¹, Gábor Recski¹

¹TU Wien, firstname.lastname@tuwien.ac.at

²Dept. of Automation and Applied Informatics, Budapest University of Technology and Economics,

lastname.firstname@aut.bme.hu

Summary



• POTATO is a human-in-the-loop XAI framework

• We provide

- a unified networkx interface for multiple graph libraries (4lang, stanza, AMR)
- a python package for learning and evaluating interpretable graph features as rules
- a human-in-the-loop (HITL) UI framework built in streamlit https://streamlit.io/

Browse dataset:						
Rule chooser and modifier	Graph viewer and evaluator 4					
First, choose class you want to use to build rules	Browse graphs: +					
Entity-Destination(e1,e2)	Choose from the rules					
You can modify any rule you want to	(u_15 / into :2 (u_2 / entity2))					
Remember, we use the <u>PENMAN</u> notation to describe a rule. You can find more information about the rules in the <u>README</u> of our repository.	into					
rules negated_rules						
(u_3 / to :2 (u_2 / entity2))	2					
(u_15 / into :2 (u_2 / entity2))	Y					
(u_264 / place :2 (u_25 / entity1))						
(u_14 / in :2 (u_2 / entity2))	(entity2)					
(u_1200 / give :2 (u_25 / entity1))						
(u_414 / put :2 (u_25 / entity1))						
(u_3 / to :2 (u_2 / entity2) :1 (u_694 / send))	Result of using all the rules: Precision: 0.762, Recall: 0.628, Fscore: 0.689					
(u_966 / add :2 (u_25 / entity1))	The rule's result: Precision: 0.762, Recall: 0.628, Fscore: 0.689, True positives: 407, False					
(u_4 / COORD :2 (u_25 / entity1) :0 (u_414 / put))	positives: 127					
(u_3 / to :1 (u_2628 / donate))	55					
(u_3 / to :1 (u_1200 / give))	Show validation data +					
(u_15 / into :2 (u_2 / entity2) :1 (u_3 / .* :2 (u_4 / entity1)))						
	Select the graphs you want to view					

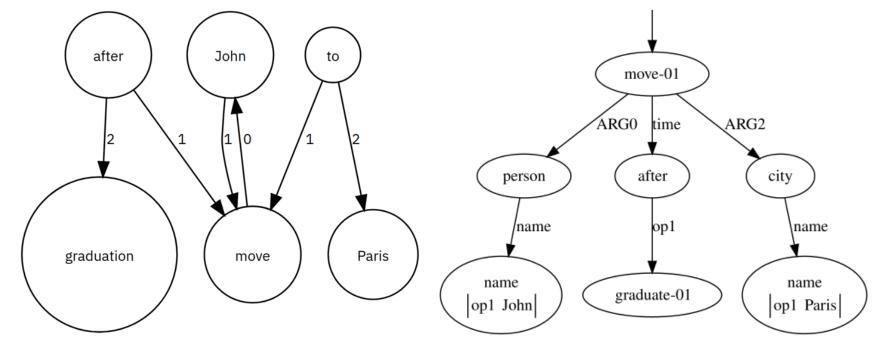
POTATO UI

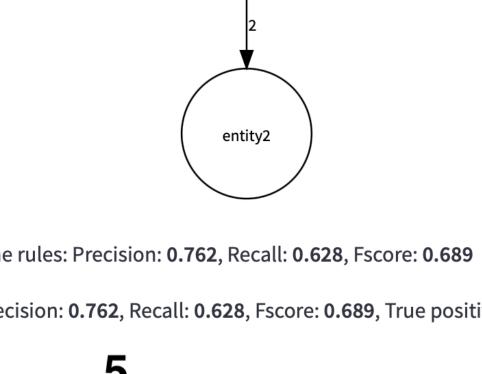
- a REST-API to use extracted features for inference in production mode
- All of our components are open-source under MIT license and can be installed with pip
- Library to build and use graphs: https://github.com/recski/tuw-nlp
- xpotato: https://github.com/adaamko/potato
- Similar libraries: HEIDL [5] and GrASp [3, 6] libraries
- Both of which support pattern-based text classification with automatic suggestions.

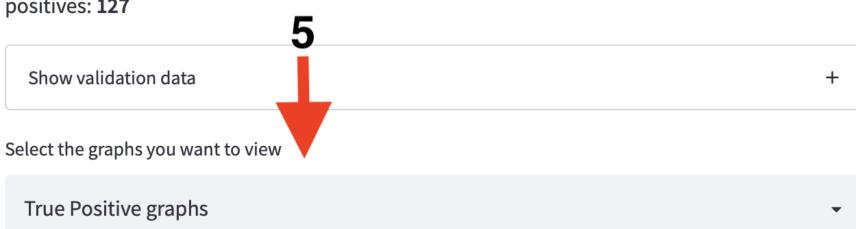
Syntactic, Semantic graphs

nsubj case case IN NN NNP VBD IN NNP After graduation John moved to Paris

Universal dependency graph (UD)







After you modified any rule, click on save updates button to save your changes

Figure 3: The main page of POTATO allows the user to (1) browse the dataset and view the processed graphs (2) choose the class you want to build rule-based systems on (3) modify, delete, add new rules and get suggestions (4) view the results of the selected rules (5) view example predictions for each rule

Human-in-the-loop learning of rules	HASOC
 Idea → use subgraphs as features for training simple classifiers (LogReg, Random Forest, etc.) Generate subgraphs only up to a certain edge number (to avoid large number of features) 	HASOC 2020 - English Precision Recall F1 Rules 95.3 74.6 83.7 BERT 90.2 90.5 90.3
 Suggest rules to users based on feature importance User can accept, reject, edit, combine patterns Subgraphs may have regexes as node or edge labels 	HASOC 2020 - German Precision Recall F1 Rules 92.4 28.3 43.4 BERT 66.6 81.7 73.4

4lang [2]

AMR [1]

Graph rules

- Rules on graphs could utilieze the underlying graph structure of texts
- SpaCy's DependencyMatcher module
 - Can be used to match rules on dependency trees.
 - But only works on UD structures
 - Complex structure
- Our own solution
 - Works with networkx
 - Can be used with arbitrary graph structures
 - Currently works with AMR [1], 41ang [2], Stanza [4]

Patterns with AMR in our system

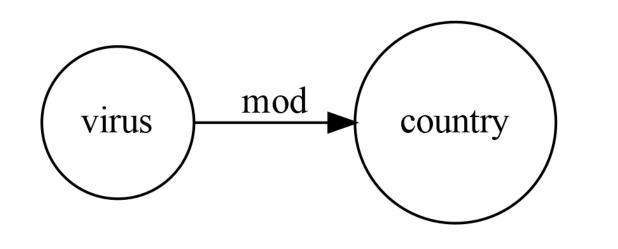
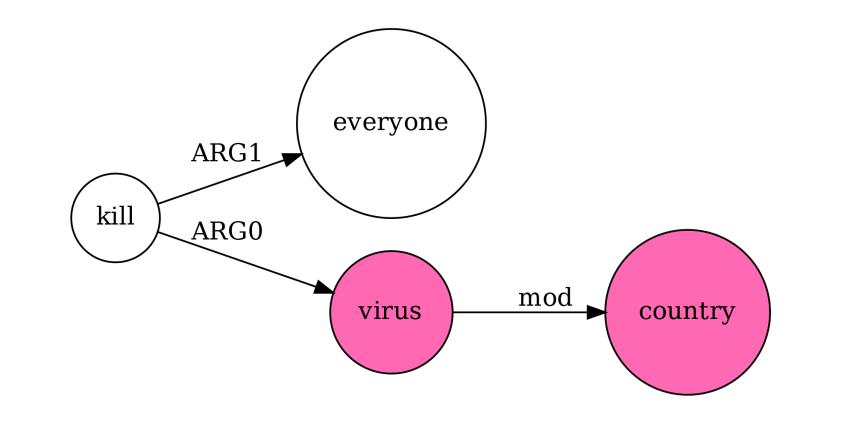


Figure 1: The written rule



- User
- Subgr
- Underspecified subgraphs can be refined

The architecture

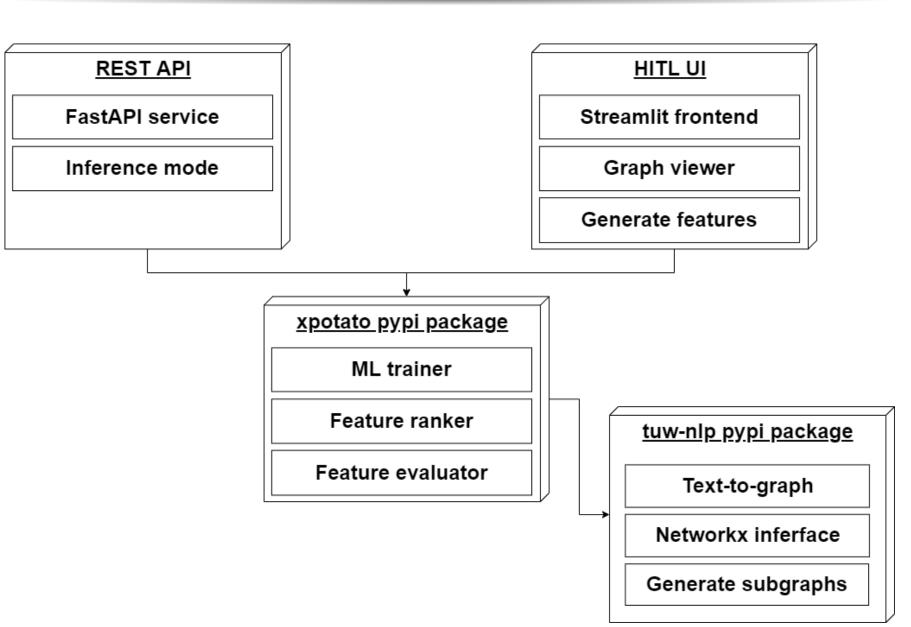


Figure 4: The system architecture of POTATO consists of 4 components

The workflow

POTATO backend	POTATO UI					
Train and	Automatic	Manual rule	Advanced			

BRISE

	BERT		R			
	Precision%	Recall%	F1%	Precision%	Recall%	F1%
Planzeichen	83	90	86	96	85	90
Dachart	88	84	86	95	84	89
BegruenungDach	90	78	84	87	91	89
AnFluchtlinie	81	71	76	89	70	79
VorkehrungBepflanzung	100	95	98	100	90	95
GebaeudeBautyp	100	52	69	100	66	80
• • •						

References

[1] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider.

Abstract Meaning Representation for sembanking.

In Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, pages 178–186, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

- [2] András Kornai.
 - Semantics.

Springer Verlag, Cham, Switzerland, 2019.

- [3] Piyawat Lertvittayakumjorn, Leshem Choshen, Eyal Shnarch, and Francesca Toni.
 - GrASP: A library for extracting and exploring human-interpretable textual patterns.
 - https://arxiv.org/abs/2104.03958, 2021.

Figure 2: Input from HASOC (*Hate Speech and Offensive Content* Identification): The Chinese virus kills everyone

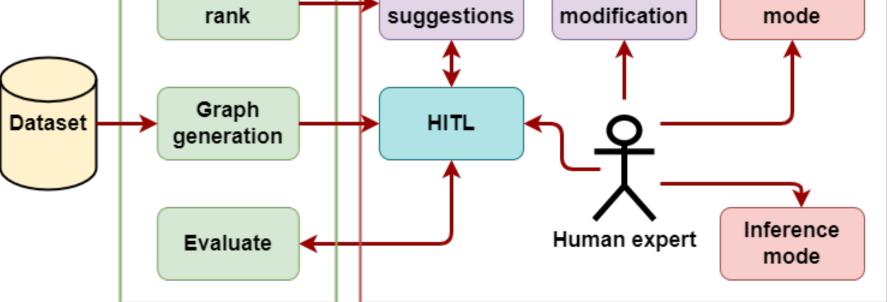


Figure 5: POTATO can be started in 3 modes: (1) simple mode, (2)advanced mode, (3) inference mode

[4] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning.

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, pages 101–108, Online, 2020. Association for Computational Linguistics.

[5] Prithviraj Sen, Yunyao Li, Eser Kandogan, Yiwei Yang, and Walter Lasecki.

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, pages 135–140, Florence, Italy, 2019. Association for Computational Linguistics.

[6] Eyal Shnarch, Ran Levy, Vikas Raykar, and Noam Slonim. GRASP: Rich patterns for argumentation mining. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1345–1350, Copenhagen, Denmark, 2017.

Association for Computational Linguistics.