arXiv:2104.03958v1 [cs.CL] 8 Apr 2021

GrASP: A Library for Extracting and Exploring Human-Interpretable Textual Patterns

Piyawat Lertvittayakumjorn[1] , **Leshem Choshen**[2] , **Eyal Shnarch**[2] and **Francesca Toni**[1]
[1]Department of Computing, Imperial College London, United Kingdom
[2]IBM Research
pl1515@imperial.ac.uk, {leshem.choshen, eyals}@il.ibm.com, ft@imperial.ac.uk

Abstract

Data exploration is an important step of every data science and machine learning project, including those involving textual data. We provide a Python library for GrASP, an existing algorithm for drawing patterns from textual data.[1] The library is equipped with a web-based interface empowering human users to conveniently explore the data and the extracted patterns. We also demonstrate the use of the library in two settings (spam detection and argument mining) and discuss future deployments of the library, e.g., beyond textual data exploration.

1 Introduction

Pattern-based analysis is a promising approach for exploring textual data in many tasks, such as authorship classification [Houvardas and Stamatatos, 2006], relation extraction [Hearst, 1992; Peng *et al.*, 2014] and argument mining [Madnani *et al.*, 2012]. Patterns uncover prominent characteristics of the data which lead to interesting insights, helping humans proceed to the next steps (e.g., data pre-processing, setting model hyperparameters). In this paper, we implement and publicly release a Python library named GrASP after a supervised algorithm which learns expressive patterns characterizing linguistic phenomena [Shnarch *et al.*, 2017]. To illustrate, examples 1–3 in Table 1 are all SMS spam messages from the dataset by [Almeida *et al.*, 2011]. While there is little word overlap between them, their commonality is apparent, even if hard to name. The GrASP algorithm can reveal an underlying structure which generalizes these three realizations of spams: a positive-sentiment word, closely followed by a determiner, and then by a proper noun.

The input for the algorithm amounts to two sets of texts: in one (the positive set) the target phenomenon appears in all examples, and in the other (the negative set) it does not. GrASP looks for commonalities prominent within the texts of one set but not shared across the sets. To be able to recognize common aspects of texts, beyond their surface form realizations, all input tokens are augmented with a variety of linguistic attributes such as part-of-speech tags, named entity information, or pertinence to a lexicon (e.g., of sentiment words). Attributes are selected to maximize a score (by default their information gain about the label). Then, they are combined by a greedy algorithm to generate patterns which are most indicative either to the positive or the negative set.

```python
from grasp import GrASP
# Step 1: Create the GrASP model
grasp_model = GrASP(num_patterns = 200, gaps_allowed = 2,
    alphabet_size = 200, include_standard = ['TEXT',
    'POS', 'NER', 'SENTIMENT'])
# Step 2: Fit it to the training data
grasp_model.fit_transform(pos_exs, neg_exs)
# Step 3: Export the results
grasp_model.to_csv('results.csv')
grasp_model.to_json('results.json')
```

Listing 1: Basic Usage of GrASP

Since the patterns are a combination of readable attributes, they are human interpretable. So, they can be used to provide insights about the data and contribute to explainability. For instance, example 1 in Table 1 is a spam message because it contains the phrase "awarded a SiPix". Patterns can also be used for classification (e.g., if the pattern mentioned above is matched in a message, we can classify it as spam).

Overall, the contribution of our work is threefold.

- We release GrASP, a Python library for extracting interpretable patterns from text data. The library contains not only the original algorithm but also additional extensions, allowing users to customize the algorithm to their specific needs (§2).
- We provide a web-based tool which displays outputs of GrASP so that they can be conveniently explored. Users can leverage both *pattern-centric* views and *example-centric* views to examine the data and find insights (§3).
- We present two use cases of the library and the exploration tool – (1) for spam detection (see the demo video[2]) and (2) for argument mining (see §4).

2 GrASP Library for Pattern Extraction

Listing 1 shows the basic usage of GrASP for extracting patterns from training data, i.e., two lists of texts containing positive examples (`pos_exs`) and negative examples

[1]The code is available at https://github.com/plkumjorn/GrASP

[2]https://youtu.be/nHOidBrKXE0

Use case	No.	Sentences Matched	Pattern
SMS Spam	1	You are **awarded a SiPix** Digital Camera...	[[SENTIMENT:pos], [POS:DET], [POS:PROPN]] (A positive-sentiment word, closely followed by a determiner, and then by a proper noun)
	2	...to **WIN a** FREE **Bluetooth** Headset...	
	3	...for **Free** ! Call **The Mobile** Update...	
Topic-Dependent Argument Mining	4	**Evidence suggests that** TOPIC is...	[[ARGUMENTATIVE], [POS:VERB], [LEMMA:that, POS:ADP]] (An argumentative word, closely followed by a verb, and then by the word "that" which is also a preposition)
	5	**Poll** in 2002 **found that** 58% of...	
	6	...**data indicate that** TOPIC reduces...	

Table 1: Examples of GrASP patterns capturing the common structure in a variety of surface forms – the sentences matched – from our two use cases. Matched words are in bold. A description of each pattern is provided below it, in parentheses.

(neg_exs). In the first step, the user specifies hyperparameters such as the desired number of patterns and the set of linguistic attributes for augmenting input tokens. The following general-purpose attributes are built-in: the token text itself (in lower case), its lemma, wordnet hypernyms[3], part-of-speech, named-entity, dependency, and sentiment tags[4]. The user can also add custom attributes (e.g., a task-specific lexicon, see §4). After that, the user trains the GrASP model and exports the results to a csv file or a json file. Both file types summarize all the extracted patterns and relevant statistics, while the json file also contains the configuration, the alphabet, and the training examples annotated with the patterns matched.

Note that in our version of GrASP, we add several parameters which are not found in the original algorithm. While the original GrASP uses information gain as the criterion to rank and select patterns, we allow the users to implement their own criteria, tailored to their use cases. For example, if they want patterns with high precision, they may use $F_{0.05}$ as a criterion. Other new parameters for customizing GrASP include the number of gaps allowed between the matched tokens (which overrides the window size parameter) and the minimum coverage for a pattern to be selected.

2.1 Supporting Features

Besides the basic usage, we provide the following features which are deemed useful in many situations.

Translating patterns into natural language explanations. It could be difficult for lay users to read and understand linguistic attributes in patterns. Therefore, using templates, we provide a function `pattern2text` for translating a pattern (e.g., [[HYPERNYM:communication.n.02], [POS:NUM]] with a window size of 4) into an English explanation (e.g., "A type of communication (n), closely followed by a number").

Using custom attributes. The library allows users to create custom attributes by inheriting the `CustomAttribute` class and implementing two functions. One extracts attributes, and the other explains the attributes for the translation feature.

Removing redundant patterns. We provide a function to remove specific patterns subsumed by more general ones. For instance, all [[TEXT:.], [POS:NUM]] matches are also matches of [[POS:NUM]]. Hence, this function removes the former (specific) pattern. A flag further allows to condition removal on the fact that the score of the removed pattern must be lower than the score of the remaining one.

Vectorizing texts using patterns. Given an input text t and n GrASP patterns, we provide a function to create a ternary vector of length n indicating whether t is matched by a positive, negative, or no pattern. This vector can be used as additional features of t for downstream tasks.

3 The Web-Based Exploration Tool

We provide a web-based exploration tool implemented using Flask[5]. Taking as input the json file exported by the GrASP library, this tool provides four types of reports: two are pattern-centric and the other two are example-centric. All reports for the spam use case and for the argument mining use case (see §4) are available online for further examination.[6] Here, due to space limits, we display only the pattern-centric views for the spam use case (i.e., Figure 1–2).

I. Pattern-centric level 1. (Figure 1) This report lists all the patterns together with the configuration of GrASP used for extracting them and the statistics about the training data. For each pattern (row), the following statistics are reported: the number of positive and negative examples matched, coverage, the metric score, precision, recall, and F1 of the pattern. Users can click a header to sort the table based on the column values. Moreover, users can click the header of the pattern column to translate the patterns into their natural language meanings. Once users click a pattern in the table, they will be redirected to that pattern's level 2 report.

II. Pattern-centric level 2. (Figure 2) This report can be generated for each pattern individually. In addition to the pattern's statistics, it shows the positive and negative examples matched by this pattern, with the corresponding tokens highlighted. Users can click the link icon at the end of each example to go to its example-centric level 2 report.

[3]We consider only three levels of synsets above the token of interest in order to exclude synsets that are too abstract to comprehend (e.g., *psychological feature*, *group action*, and *entity*).

[4]We use spacy for tokenization and tagging, nltk and lesk [Lesk, 1986] for word sense disambiguation and finding hypernyms, and a lexicon by [Hu and Liu, 2004] for sentiment tagging.

[5]https://flask.palletsprojects.com/en/1.1.x/

[6]https://plkumjorn.pythonanywhere.com/

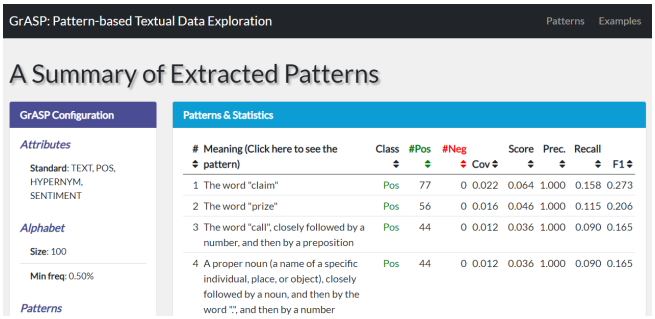

Figure 1: The pattern-centric level 1 report listing the patterns and their metrics, the GrASP configuration, and the training set statistics.

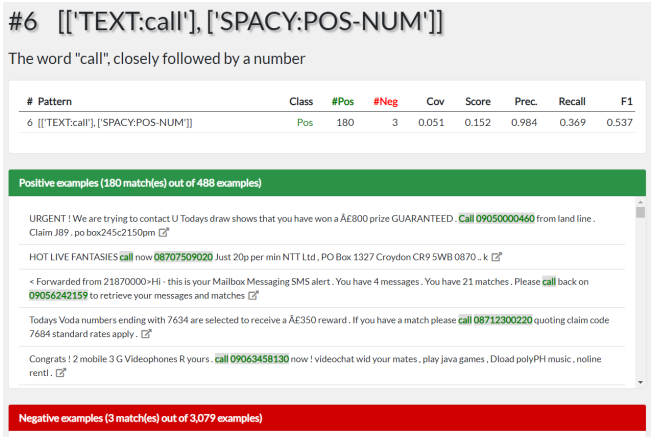

Figure 2: The pattern-centric level 2 report listing positive and negative examples matched by a specific pattern.

III. Example-centric level 1. This report lists all positive and negative training examples (with pagination). For each example, words matched by only positive pattern(s) and only negative pattern(s) are highlighted in green and red, respectively, while words matched by both types of patterns are in purple. By clicking a highlighted word, users can see the list of patterns matching this word. Clicking the link icon of an example directs to its example-centric level 2 report.

IV. Example-centric level 2. This report can be generated for each training example, showing all the positive and negative patterns matching it. Clicking a pattern links to its pattern-centric level 2 report.

Overall, these reports provide a convenient way for quickly exploring the data and gaining insights from the extracted patterns, as we show next.

4 Use Case: Argument Mining

We present a use case of our tools targeting the corpus of topic-dependent argument mining [Shnarch *et al.*, 2018]. This dataset includes 4,065 training and 1,720 test pairs of topics and sentences, each labeled with whether the sentence is a piece of valid evidence for the topic.

We augment each token with the built-in attributes as well as a custom binary attribute indicating whether a token was found in a lexicon of argumentative words. We use information gain to select patterns, set the number of patterns to 100, and allow up to 2 gaps in each pattern. Next, we train and examine the patterns with the pattern-centric reports.

We find that the most indicative pattern is the word *that* in its *preposition* meaning (as opposed to other meanings of that word, e.g., a determiner, as in this sentence). This is a very nice outcome since GrASP automatically revealed what experts have found – "that" is an indicator of argumentative content and can be used as an effective source of weak supervision [Levy *et al.*, 2017]. Other informative patterns are about studies: presenting them (e.g., [[`POS:NUM`],[`LEMMA:study`]] which matches phrases like ***one study***, ***Two*** *major* ***studies***, *A* ***1999*** *meta* ***study***, and *a* ***25*** *year longitudinal* ***study***), declaring their findings (e.g., examples 4–6 in Table 1), and reporting their experimental results (e.g., [[`POS:ADP`],[`POS:VERB`],[`HYPERNYM:risk`]] which covers expressions such as ***in reducing*** *the* ***risk***, *more* ***than tripled*** *the* ***risk***, and ***as controlling*** *the* ***risks***). These patterns, too, have rediscovered what is known in the domain – texts describing studies are a good source for evidence [Rinott *et al.*, 2015; Ein-Dor *et al.*, 2020].

The exploration tool enables having a human in the loop by making it easy to compare patterns (e.g., by their recall-precision trade off), examine their realizations in the positive and negative sets, check patterns combination (e.g., click on a purple word in the example-centric report to see all patterns that matched it), and evaluate the contribution of adding custom attributes (such as the lexicon of argumentative words). After that, users can improve the list of patterns by filtering, correcting, and merging patterns. The selected patterns can then contribute to the downstream pipeline (e.g., by using the most precise/high-coverage patterns to improve precision/recall). Alternatively, the patterns can be used to obtain weak supervision, combined with labeled data, to improve performance as in [Shnarch *et al.*, 2018]. Finally, similar to [Sen *et al.*, 2020], the human-readable patterns matching an example can provide explainability for the model decision for it.

5 Discussion

Apart from extracting and exploring patterns from textual datasets, GrASP has potential to be applied to several other problems such as explainable text classification (where patterns provide explanations) [Ribeiro *et al.*, 2018], human-AI model co-creation (with patterns enabling communication) [Yang *et al.*, 2019], and analyzing deep learning models (by studying patterns that the models or the neurons capture) [Lertvittayakumjorn *et al.*, 2020; Albini *et al.*, 2020]. Furthermore, the GrASP algorithm and the provided exploration tool are language-agnostic. To apply GrASP to a language besides English, users need to create custom attributes that augment input tokens with linguistic features of the target language. More generally, GrASP is applicable to other types of sequence data beyond text as long as we can extract suitable attributes for the sequence data [Agrawal and Srikant, 1995]. Lastly, it is possible and interesting to extend our library with the functionalities of GrASP^{lite} [Shnarch *et al.*, 2020] — extracting patterns in an unsupervised way.

References

[Agrawal and Srikant, 1995] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

[Albini *et al.*, 2020] Emanuele Albini, Piyawat Lertvittayakumjorn, Antonio Rago, and Francesca Toni. Dax: Deep argumentative explanation for neural networks. *arXiv preprint arXiv:2012.05766*, 2020.

[Almeida *et al.*, 2011] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262, 2011.

[Ein-Dor *et al.*, 2020] Liat Ein-Dor, Eyal Shnarch, Lena Dankin, Alon Halfon, Benjamin Sznajder, Ariel Gera, Carlos Alzate, Martin Gleize, Leshem Choshen, Yufang Hou, Yonatan Bilu, Ranit Aharonov, and Noam Slonim. Corpus wide argument mining—a working solution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7683–7691, Apr. 2020.

[Hearst, 1992] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, 1992.

[Houvardas and Stamatatos, 2006] John Houvardas and Efstathios Stamatatos. N-gram feature selection for authorship identification. In *International conference on artificial intelligence: Methodology, systems, and applications*, pages 77–86. Springer, 2006.

[Hu and Liu, 2004] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[Lertvittayakumjorn *et al.*, 2020] Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–348, Online, November 2020. Association for Computational Linguistics.

[Lesk, 1986] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, 1986.

[Levy *et al.*, 2017] Ran Levy, Shai Gretz, Benjamin Sznajder, Shay Hummel, Ranit Aharonov, and Noam Slonim. Unsupervised corpus–wide claim detection. In *Proceedings of the 4th Workshop on Argument Mining*, pages 79–84, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[Madnani *et al.*, 2012] Nitin Madnani, Michael Heilman, Joel Tetreault, and Martin Chodorow. Identifying high-level organizational elements in argumentative discourse. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 20–28, Montréal, Canada, June 2012. Association for Computational Linguistics.

[Peng *et al.*, 2014] Yifan Peng, Manabu Torii, Cathy H Wu, and K Vijay-Shanker. A generalizable nlp framework for fast development of pattern-based biomedical relation extraction systems. *BMC bioinformatics*, 15(1):1–18, 2014.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Rinott *et al.*, 2015] Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[Sen *et al.*, 2020] Prithviraj Sen, Marina Danilevsky, Yunyao Li, Siddhartha Brahma, Matthias Boehm, Laura Chiticariu, and Rajasekar Krishnamurthy. 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)*, pages 4211–4221, Online, November 2020. Association for Computational Linguistics.

[Shnarch *et al.*, 2017] 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, September 2017. Association for Computational Linguistics.

[Shnarch *et al.*, 2018] Eyal Shnarch, Carlos Alzate, Lena Dankin, Martin Gleize, Yufang Hou, Leshem Choshen, Ranit Aharonov, and Noam Slonim. Will it blend? blending weak and strong labeled data in a neural network for argumentation mining. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–605, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Shnarch *et al.*, 2020] Eyal Shnarch, Leshem Choshen, Guy Moshkowich, Ranit Aharonov, and Noam Slonim. Unsupervised expressive rules provide explainability and assist human experts grasping new domains. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2678–2697, Online, November 2020. Association for Computational Linguistics.

[Yang *et al.*, 2019] Yiwei Yang, Eser Kandogan, Yunyao Li, Prithviraj Sen, and Walter S Lasecki. A study on interaction in human-in-the-loop machine learning for text analytics. In *IUI Workshops*, 2019.