Journal of Machine Learning Research 13 (2012) 2063-2067 Submitted 8/11; Revised 3/12; Published 6/12

Pattern for Python

Tom De Smedt TOM.DESMEDT@UA.AC.BE
Walter Daelemans WALTER.DAELEMANS@UA.AC.BE
CLiPS Computational Linguistics Group
University of Antwerp
2000 Antwerp, Belgium

Editor: Cheng Soon Ong

Abstract

Pattern is a package for Python 2.4+ with functionality for web mining (Google + Twitter + Wikipedia, web spider, HTML DOM parser), natural language processing (tagger/chunker, n-gram search, sentiment analysis, WordNet), machine learning (vector space model, *k*-means clustering, Naive Bayes + *k*-NN + SVM classifiers) and network analysis (graph centrality and visualization). It is well documented and bundled with 30+ examples and 350+ unit tests. The source code is licensed under BSD and available from `http://www.clips.ua.ac.be/pages/pattern`.

Keywords: Python, data mining, natural language processing, machine learning, graph networks

1. Introduction

The World Wide Web is an immense collection of linguistic information that has in the last decade gathered attention as a valuable resource for tasks such as machine translation, opinion mining and trend detection, that is, "Web as Corpus" (Kilgarriff and Grefenstette, 2003). This use of the WWW poses a challenge since the Web is interspersed with code (HTML markup) and lacks metadata (language identification, part-of-speech tags, semantic labels).

"Pattern" (BSD license) is a Python package for web mining, natural language processing, machine learning and network analysis, with a focus on ease-of-use. It offers a mash-up of tools often used when harnessing the Web as a corpus, which usually requires several independent toolkits chained together in a practical application. Several such toolkits with a user interface exist in the scientific community, for example ORANGE (Demšar et al., 2004) for machine learning and GEPHI (Bastian et al., 2009) for graph visualization. By contrast, PATTERN is more related to toolkits such as NLTK (Bird et al., 2009), PYBRAIN (Schaul et al., 2010) and NETWORKX (Hagberg et al., 2008), in that it is geared towards integration in the user's own programs. Also, it does not specialize in one domain but provides general cross-domain functionality.

The package aims to be useful to both a scientific and a non-scientific audience. The syntax is straightforward. Function names and parameters were so chosen as to make the commands self-explanatory. The documentation assumes no prior knowledge. We believe that PATTERN is valuable as a learning environment for students, as a rapid development framework for web developers, and in research projects with a short development cycle.

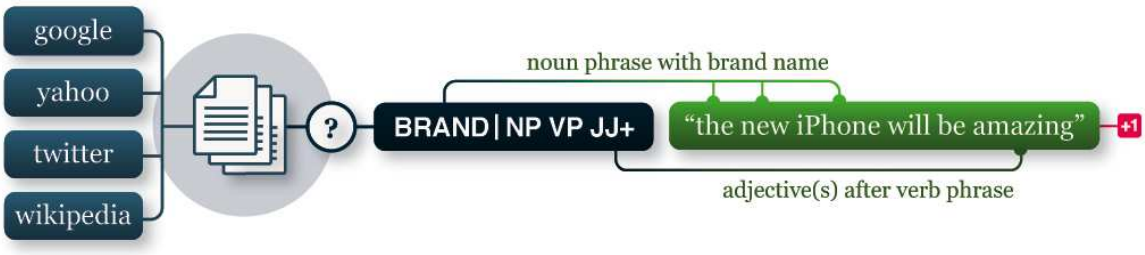

Figure 1: Example workflow. Text is mined from the web and searched by syntax and semantics. Sentiment analysis (positive/negative) is performed on matching phrases.

2. Package Overview

PATTERN is organized in separate modules that can be chained together, as shown in Figure 1. For example, text from Wikipedia (pattern.web) can be parsed for part-of-speech tags (pattern.en), queried by syntax and semantics (pattern.search), and used to train a classifier (pattern.vector).

pattern.web Tools for web data mining, using a download mechanism that supports caching, proxies, asynchronous requests and redirection. A `SearchEngine` class provides a uniform API to multiple web services: Google, Bing, Yahoo!, Twitter, Wikipedia, Flickr and news feeds using FEED PARSER (packages.python.org/feedparser). The module includes an HTML parser based on BEAUTIFUL SOUP (crummy.com/software/beautifulsoup), a PDF parser based on PDFMINER (unixuser.org/ euske/python/pdfminer), a web crawler, and a webmail interface.

pattern.en Fast, regular expressions-based shallow parser for English (identifies sentence constituents, e.g., nouns, verbs), using a finite state part-of-speech tagger (Brill, 1992) extended with a tokenizer, lemmatizer and chunker. Accuracy for Brill's tagger is 95% and up. A parser with higher accuracy (MBSP) can be plugged in. The module has a `Sentence` class for parse tree traversal, functions for singularization/pluralization (Conway, 1998), conjugation, modality and sentiment analysis. It comes bundled with WORDNET3 (Fellbaum, 1998) and PYWORDNET.

pattern.nl Lightweight implementation of pattern.en for Dutch, using the BRILL-NL language model (Geertzen, 2010). Contributors are encouraged to read the developer documentation on how to add support for other languages.

pattern.search N-gram pattern matching algorithm for `Sentence` objects. The algorithm uses an approach similar to regular expressions. Search queries can include a mixture of words, phrases, part-of-speech-tags, taxonomy terms (e.g., `pet` = dog, cat or goldfish) and control characters (e.g., + = multiple, * = any, ( ) = optional) to extract relevant information.

pattern.vector Vector space model using a `Document` and a `Corpus` class. Documents are lemmatized bag-of-words that can be grouped in a sparse corpus to compute TF-IDF, distance metrics (cosine, Euclidean, Manhattan, Hamming) and dimension reduction (Latent Semantic Analysis). The module includes a hierarchical and a *k*-means clustering algorithm, optimized with the *k*-means++ initialization algorithm (Arthur and Vassilvitskii, 2007) and triangle inequality (Elkan, 2003). A Naive Bayes, a *k*-NN, and a SVM classifier using LIBSVM (Chang and Li, 2011) are included, with tools for feature selection (information gain) and K-fold cross validation.

pattern.graph Graph data structure using `Node`, `Edge` and `Graph` classes, useful (for example) for modeling semantic networks. The module has algorithms for shortest path finding, subgraph partitioning, eigenvector centrality and betweenness centrality (Brandes, 2001). Centrality algorithms were ported from NETWORKX. The module has a force-based layout algorithm that positions nodes in 2D space. Visualizations can be exported to HTML and manipulated in a browser (using our `canvas.js` helper module for the HTML5 Canvas2D element).

pattern.metrics Descriptive statistics functions. Evaluation metrics including a code profiler, functions for accuracy, precision and recall, confusion matrix, inter-rater agreement (Fleiss' kappa), string similarity (Levenshtein, Dice) and readability (Flesch).

pattern.db Wrappers for CSV files and SQLITE and MYSQL databases.

3. Example Script

As an example, we chain together four PATTERN modules to train a *k*-NN classifier on adjectives mined from Twitter. First, we mine 1,500 tweets with the hashtag #win or #fail (our classes), for example: "$20 tip off a *sweet little old* lady today #win". We parse the part-of-speech tags for each tweet, keeping adjectives. We group the adjective vectors in a corpus and use it to train the classifier. It predicts "sweet" as `WIN` and "stupid" as `FAIL`. The results may vary depending on what is currently buzzing on Twitter.

The source code is shown in Figure 2. Its size is representative for many real-world scenarios, although a real-world classifier may need more training data and more rigorous feature selection.

```
from pattern.web    import Twitter
from pattern.en     import Sentence, parse
from pattern.search import search
from pattern.vector import Document, Corpus, KNN

corpus = Corpus()
for i in range(1,15):
    for tweet in Twitter().search('#win OR #fail', start=i, count=100):
        p = '#win' in tweet.description.lower() and 'WIN' or 'FAIL'
        s = tweet.description.lower()
        s = Sentence(parse(s))
        s = search('JJ', s) # JJ = adjective
        s = [match[0].string for match in s]
        s = ' '.join(s)
        if len(s) > 0:
            corpus.append(Document(s, type=p))

classifier = KNN()
for document in corpus:
    classifier.train(document)
print classifier.classify('sweet') # yields 'WIN'
print classifier.classify('stupid') # yields 'FAIL'
```

Figure 2: Example source code for a *k*-NN classifier trained on Twitter messages.

4. Case Study

As a case study, we used PATTERN to create a Dutch sentiment lexicon (De Smedt and Daelemans, 2012). We mined online Dutch book reviews and extracted the 1,000 most frequent adjectives. These were manually annotated with positivity, negativity, and subjectivity scores. We then enlarged the lexicon using distributional expansion. From the TWNC corpus (Ordelman et al., 2007) we extracted the most frequent nouns and the adjectives preceding those nouns. This results in a vector space with approximately 5,750 adjective vectors with nouns as features. For each annotated adjective we then computed *k*-NN and inherited its scores to neighbor adjectives. The lexicon is bundled into PATTERN 2.3.

5. Documentation

PATTERN comes bundled with examples and unit tests. The documentation contains a quick overview, installation instructions, and for each module a detailed page with the API reference, examples of use and a discussion of the scientific principles. The documentation assumes no prior knowledge, except for a background in Python programming. The unit test suite includes a set of corpora for testing accuracy, for example POLARITY DATA SET V2.0 (Pang and Lee, 2004).

6. Source Code

PATTERN is written in pure Python, meaning that we sacrifice performance for development speed and readability (i.e., slow clustering algorithms). The package runs on all platforms and has no dependencies, with the exception of NumPy when LSA is used. The source code is annotated with developer comments. It is hosted online on GitHub (github.com) using the Git revision control system. Contributions are welcomed.

The source code is released under a BSD license, so it can be incorporated into proprietary products or used in combination with other open source packages such as SCRAPY (web mining), NLTK (natural language processing), PYBRAIN and PYML (machine learning) and NETWORKX (network analysis). We provide an interface to MBSP FOR PYTHON (De Smedt et al., 2010), a robust, memory-based shallow parser built on the TIMBL machine learning software. The API's for the PATTERN parser and MBSP are identical.

Acknowledgments

Development was funded by the Industrial Research Fund (IOF) of the University of Antwerp.

References

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. *Proceedings of the Third International ICWSM Conference*, 2009.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

Eric Brill. A simple rule-based part of speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155, 1992.

Chih-Chung Chang and Chih-Jen Li. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 2011.

Damian Conway. An algorithmic approach to english pluralization. *Proceedings of the Second Annual Perl Conference*, 1998.

Tom De Smedt and Walter Daelemans. Vreselijk mooi! (terribly beautiful): A subjectivity lexicon for dutch adjectives. *Proceedings of the 8th Language Resources and Evaluation Conference (LREC'12)*, pages 3568—-3572, 2012.

Tom De Smedt, Vincent Van Asch, and Walter Daelemans. Memory-based shallow parser for python. *CLiPS Technical Report Series*, 2, 2010.

Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From experimental machine learning to interactive data mining. *Knowledge Discovery in Databases*, 3202:537–539, 2004.

Charles Elkan. Using the triangle inequality to accelerate k-means. *Proceedings of the Twentieth International Conference on Machine Learning*, pages 147–153, 2003.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, 1998.

Jeroen Geertzen. Jeroen geertzen :: software & demos : Brill-nl, June 2010. URL `http://cosmion.net/jeroen/software/brill\_pos/`.

Aric Hagberg, Daniel Schult, and Pieter Swart. Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conference*, pages 11–15, 2008.

Adam Kilgarriff and Gregory Grefenstette. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–347, 2003.

Roeland Ordelman, Franciska de Jong, Arjan van Hessen, and Hendri Hondorp. TwNC: A multifaceted dutch news corpus. *ELRA Newsletter*, 12:3–4, 2007.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of the ACL*, pages 271–278, 2004.

Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. Pybrain. *Journal of Machine Learning Research*, pages 743–746, 2010.