The Journal of Open Source Software

Drugsideeffect-test: A Lexicon-Driven Visualization Framework for Drug Side-Effect Signals in Text

Briti Deb[1¶]

1 Independent Researcher, Bangalore, Karnataka, India ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 11 February 2026
Published: unpublished

Summary

Drugsideeffect-test (v0.1.0) is a Python package (import as 'drugsideeffect') for exploring, categorizing, and visualizing drug side-effect mentions from textual data for research purposes; it is not intended for clinical diagnosis or decision-making. Repository: https://github.com/debbdeb/drugsideeffect. It can be installed via PyPI Test repository for testing: pip install -i https://test.pypi.org/simple/ drugsideeffect-test.

Key dependencies include: pandas >=1.5, numpy >=1.23, matplotlib >=3.6, seaborn >=0.12, plotly >=5.11, nltk >=3.8, textblob >=0.17, and spacy >=3.5. Pretrained models are included in the package (drugsideeffect/models/) [1][2].

For stable releases, the package will be available on PyPI: pip install drugsideeffect-test. The package provides tools for exploring, categorizing, and visualizing self-reported drug side-effect mentions from textual data for research purposes; it is not intended for clinical diagnosis or decision-making. The software integrates supervised machine learning classification with multi-layered lexicon-based extraction and visualization. It is specifically designed to bridge informal patient-generated content (e.g., tweets) with structured pharmacovigilance-oriented symptom categories.

The system combines: 1. A pretrained TF–IDF + Naive Bayes classifier pre-trained model for filtering side-effect-related text [3]. 2. A set of curated lexicons for common symptoms, uncommon vaccine-related adverse events, slang / figurative expressions, temporal patterns, and alert keywords. 3. A modular visualization layer for sentiment analysis, symptom distribution, temporal trends, engagement metrics, and onset/duration analysis.

The primary innovation of drugsideeffect-test lies in its lexicon architecture and its coupling with visual analytics. Rather than treating lexicons solely as keyword filters, the package operationalizes them as semantic layers that drive multi-dimensional visualizations.

The full pipeline (Figure 1) can be executed via a single function, sideeffect_pipeline(), which loads data, filters side-effect mentions, processes onset timing, and generates all visual outputs. The GitHub repository link is: https://github.com/debbdeb/drugsideeffect

Figure 1. Overview of the Drugsideeffect-test pipeline. The workflow begins with loading input CSV data, followed by Naive Bayes classification to filter texts mentioning side effects. Filtered texts are then processed using curated lexicons—including known symptoms, uncommon effects, slang, temporal patterns, and alert keywords—for multi-layered extraction. Finally, the extracted information feeds into the visualization module generating sentiment, symptom frequency, temporal trends, and engagement analyses.

Statement of Need

Pharmacovigilance traditionally relies on structured adverse event/side-effect reporting systems such as VAERS or FAERS [8][9]. However, patient-reported outcomes increasingly emerge in unstructured online and other environments including social media [13]. These sources contain valuable early signals but present substantial challenges: ▪ Informal language and slang ▪ Non-standard symptom descriptions ▪ Temporal ambiguity ▪ Sentiment bias and emotional framing ▪ Misinformation amplification ▪ Engagement-driven distortion (likes, retweets) ▪ Metaphor, slang, hyperbole, simile, personification, irony, allusion, understatement, pun, oxymoron, colloquial/idiomatic expressions

Existing NLP tools often focus either on classification (trying to answer questions such as - is this a side-effect post?) or named entity recognition (extract medical terms) [6][7]. Few open-source tools integrate: ▪ Informal-to-clinical lexical mapping ▪ Temporal phrase detection ▪ Sentiment context ▪ Engagement metrics ▪ Visualization of lexicon-derived features

Drugsideeffect-test aims to addresses this gap by combining machine learning filtering with a layered lexicon-driven visualization framework. The package can be particularly useful for: ▪ Public health researchers ▪ Computational social scientists ▪ Pharmacovigilance analysts ▪ Vaccine confidence researchers ▪ NLP researchers studying health misinformation

Implementation

The system is structured into several components:

1. Side-Effect Classification (processing.py) The classification module loads pretrained artifacts included in the package (drugsideeffect/models/): ▪ sideeffect_nb.pkl: Naive Bayes classifier ▪ tfidf_vectorizer.pkl: TF–IDF feature extractor

The classify_sideeffects function takes a DataFrame containing a text column and predicts a binary label indicating likely side-effect mentions. ▪ Requires a text column. ▪ Vectorizes the text. ▪ Predicts a binary label indicating likely side-effect mentions in text for exploratory research only; labels do not confirm medical events. ▪ Filters the dataset to retain only side-effect text. This filtering reduces noise before lexicon-based extraction is applied.

3. Lexicon Architecture The lexicon layer is the core conceptual contribution of the package. It consists of five primary lexicon categories: A. Known Symptom Lexicon

A curated list of common side effects (particularly in the context of COVID-19), e.g., fever, headache, fatigue, muscle pain, insomnia, and anxiety disorders (representative, not exhaustive).

This lexicon supports: ▪ Frequency visualization ▪ Common symptom detection ▪ Flag-based filtering

B. Uncommon Vaccine Side Effects Lexicon

This lexicon includes severe or rare conditions, particularly in the context of COVID-19, (representative, not exhaustive): ▪ myocarditis ▪ thrombosis ▪ stroke ▪ psychosis ▪ autoimmunity ▪ long covid

The purpose is solely for detecting narrative prevalence in text data, not for clinical validation or assessment of patient risk.

Visualizations include: ▪ Top uncommon side-effect frequency ▪ Pie charts comparing common vs. uncommon symptom prevalence

C. Slang-to-Symptom Lexicon

A mapping from informal expressions to symptom categories (representative, not exhaustive):

“feel like shit” → [“fatigue”, “malaise”] “brain fog” → [“cognitive”] “my head is pounding” → [“headache”] “heart racing” → [“arrhythmia”]

This lexicon enables: ▪ Informal phrase detection ▪ Bridging colloquial and medical language ▪ Monthly trend analysis of slang usage ▪ Sentiment-stratified slang frequency plots

This component is critical because social media [10][11][12] users rarely use formal clinical terminology.

D. Temporal Pattern Lexicon

Regular expressions detect time references such as (representative, not exhaustive): ▪ “2 days after vaccination” ▪ “few hours post-dose” ▪ “day after the shot”

This supports: ▪ Time-series visualization ▪ Onset pattern extraction ▪ Temporal clustering analysis Temporal detection is essential for pharmacovigilance, as side effects are time-dependent.

E. Alert Keyword Lexicon Terms indicating urgency or severity: ▪ urgent ▪ serious ▪ critical

This lexicon captures emotional escalation or perceived severity.

Visualization Framework

The visualization layer integrates lexicon outputs into interpretable analytics.

Sentiment Distribution Using TextBlob polarity scoring [5]: ▪ Positive ▪ Neutral ▪ Negative This contextualizes symptom mentions within emotional framing.

English Word Proportion A linguistic quality control metric using NLTK’s word corpus [4]: ▪ Measures proportion of recognized English tokens ▪ Assesses data noise or bot-like behavior

Slang and Time-Series Visualizations The function plot_visualize() handles slang and temporal pattern visualizations: ▪ Explodes matched slang and time patterns ▪ Computes tweet volumes, average sentiment, and engagement ▪ Produces bar plots (tweet volume) and line plots (sentiment & engagement)

Engagement-Aware Visualization The function plot_visualize() integrates: ▪ Tweet volume ▪ Average sentiment polarity ▪ Average RT_Like (engagement metric)

Using a triple-axis visualization: ▪ Bar chart → tweet frequency ▪ Line plot → sentiment ▪ Line plot → engagement This reveals amplification dynamics, showing whether emotionally charged posts receive disproportionate engagement.

Symptom Extraction Visualization plot_symptom_extraction(): ▪ Extracts known and uncommon symptoms ▪ Flags posts mentioning each ▪ Produces: ▪ Top 10 bar charts ▪ Pie chart of common vs. uncommon vs. alert keywords

Onset and Duration Visualization plot_onset_times() visualizes: ▪ Distribution of symptom onset (in hours) ▪ Duration of symptoms This component bridges textual temporal expressions with quantitative modeling.

Pipeline Integration The sideeffect_pipeline() function orchestrates the entire workflow: ▪ Load CSV ▪ Print initial row count ▪ Apply ML side-effect filter ▪ Process onset timing ▪ Generate all plots ▪ Return filtered DataFrame This design ensures usability for non-expert users.

Example Usage

import drugsideeffect as dse df_filtered = dse.sideeffect_pipeline(“sample_data.csv”)

Design Philosophy

The system follows four principles: 1. Layered Semantics: ML for text filtering, lexicons for interpretability. 2. Visualization-Centered Analysis: Lexicons drives the visual analytic. 3. Reproducibility: Pretrained models + deterministic lexicon rules. 4. Extensibility: Lexicons are editable Python dictionaries/lists.

Limitations

▪ TextBlob sentiment is rule-based and may miss sarcasm. ▪ Lexicons require manual curation. ▪ English-only focus. ▪ Regex temporal extraction may miss complex phrasing. ▪ Rare condition mentions in social media posts should not be interpreted as medical diagnosis or causal evidence. ▪ The system is designed for signal exploration, not causal inference.

Extensibility

Future directions include: ▪ Transformer-based sentiment models ▪ Multilingual lexicons ▪ Ontology integration ▪ Network analysis of symptom co-occurrence ▪ Dashboard deployment (Streamlit or Dash)

Conclusion

Drugsideeffect-test provides a lexicon-driven framework for analyzing and visualizing drug side-effect discourse in textual data. By combining supervised classification with structured lexical layers and visual analytics, the package supports exploratory pharmacovigilance research and digital epidemiology. The framework enables real-time exploration of textually-reported side-effect narratives, quantifying and visualizing informal expressions (e.g., “drained”, “weary”), temporal patterns (e.g., “days ago”), and sentiment dynamics. These capabilities complement traditional pharmacovigilance methods in research contexts. The software is open source and designed for transparency, reproducibility, and extension by the research community.

License

Drugsideeffect-test is distributed under the MIT License.