From code to Insight:

Using NLP and Pattern Analysis in Git History

Pavel Perfilov

About me

- 15+ year in Fintech - in various roles: Engineer, BA Project/Product management
- 7+ years of management experience in Software Development
- 8+ years of dev experience

- Master degree in Finance/Computer Science
- Focused on practical usage of DS, ML and Data Engineering

- Big fan of Emotional Intelligence Science

^ Linkedin

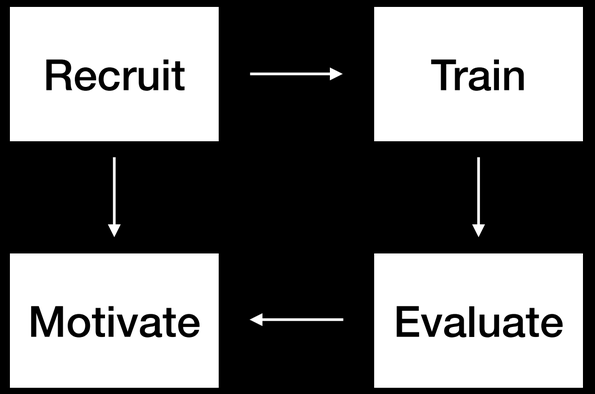
github.com/perfilovp/NLP-git

Speaking as myself and not as a representative of my employer

Project management

- Management theory

- People management



What is missing here?

Context, Emotions, Sentiments 🥳 … NLP is coming

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

Sources of sentiments

Commits, documentation, messages, feedback, conversations, etc

How?

Get insights from communication style by using NLP

Let’s use git log !

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

git log

Code: “What” and “Why”

Commit message:

- Meta information about the code
- Explanation of the new behavior or fixed behavior
- Is it a feature or a bug fix?
- Various links (ticketing systems)
- “Why the code is important?”

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

git log

Dev -> I read it when I need to “travel back in time” 🥹

Team lead -> Explanation of what the team was doing 🧐

Analyst -> Feature implementation logic 😜

Project Manager -> I need a ticket, I’m waiting all my tickets to be closed 🤔

Product Manager -> ..nope.. it’s of dev stuff 🤯

CTO/CIO -> 🤯🤯 I don’t care, I have my own plan and vision

HR -> 🤯🤯 WHAT!? I’m looking on git only if I’m recruiting

Top Management -> 🧐🧐 we trust our CTO/CIO

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

git log Insights

Dev ->	**What makes me encouraged?**
Team lead ->	**How my team is doing, are they happy?**
Analyst ->	**What is the most “problematic logic”?**
🧐 Project Manager ->	**In what direction we’re going ?**
🧐 Product Manager ->	**What are the most exciting features?**
🧐 CTO ->	**What are the trends, what is exciting?**
🧐 HR ->	**What projects demotivate people?**
🧐 Top Management ->	**Do we have a healthy environment?**

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

Natural Language Processing - NLP

NLP + git
Analyze commit messages to provide actionable insights for software teams and project managers.

By using linguistic features of commit messages, tracking sentiments of contributors over time, identifying common patterns so managers/devs/leads could build a comprehensive understanding of software development dynamics and promote data-driven decision making.

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

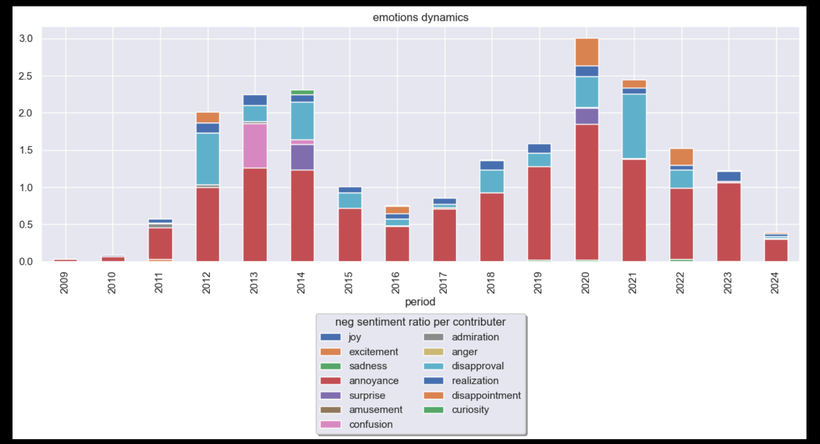
git log Real examples

https://github.com/pandas-dev/pandas.git

date	email	message	period	message_lemmas	message_tagged_lemmas	is_bug	is_feature	bad_words	neg_words	pos_words	avg_neg_score	avg_pos_score	one	polarity
2012-01-03 23:21:12	bebnlmfjo@gmail.com	fixed stupid errors	2012	fixed stupid error	[(fixed, VBN), (stupid, JJ), (error, NN)]	False	False	stupid	{(error, NN, 0.625)}	None	0.625	0.0	1	-0.350000
2012-04-10 03:52:48	lnjdibfm/ bzf@gmail.com	debugged stupid typo	2012	debugged stupid typo	[(debugged, VBN), (stupid, JJ), (typo, NN)]	False	False	stupid	None	None	0.000	0.0	1	-0.800000
2013-01-20 01:29:39	xftndljoo@gmail.com	bug: parser-tokenizer incorrect state in trailing field handling logic. stupid goto. close #2668	2013	bug incorrect state trailing field handling logic stupid goto close	[(bug, NN), (incorrect, NN), (state, NN), (trailing, VBG), (field, NN), (handling, VBG), (logic, JJ), (stupid, JJ), (goto, NN), (close, NN)]	True	False	stupid	None	None	0.000	0.0	1	-0.800000

```
sample.nlargest(5, "surprise")
```

date	email	message	period	message_lemmas	message_tagged_lemmas	is_bug	is_feature
2011-07-02 02:37:49	xftndljoo@gmail.com	amazingly all the unit tests pass post redesign. much cleanup still needed	2011	amazingly unit test pas post redesign much cleanup still needed	[(amazingly, RB), (unit, NN), (test, NN), (pas, NN), (post, NN), (redesign, NN), (much, JJ), (cleanup, NN), (still, RB), (needed, VBD)]	False	False



Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

Typical NLP pipeline

- Technically speaking NLP contains few steps:
  - Preprocessing: cleaning the data, enriching the data, removing “stop words”
  - Tokenization: Splitting the text into individual words or tokens
  - Vectorization: Converting the text data into numerical vectors
  - ML processing
    - Sentiment, polarity
    - Pattern analysis - similarity scores
    - Categorization and tagging
    - Training DeepLearning models, to increase the quality
    - etc.

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History

NLP Python Libraries

- *TextBlob - easy to use, very high level library
- *NLTK - easy-to-use interfaces, a lot of external models

And many more:

- SpaCy - tool for building applications that can recognize and categorize patterns in text
- pyfpgrowth - to find frequent patterns
- sklearn - “standard ML” toolkit
- Gensim - build your own model
- TensorFlow, Keras - deep learning …

Examples:

Descriptive stats

Triggering words

Sentiment analysis

Classification and categorization

Pattern search

Information extraction and summarization

https://github.com/perfilovp/NLP-git

What is important

- NLP model tuning is very iterative process
- Typically small messages are not having enough information about sentiments
- Once you get sentiments, check the original commits.
  Check if scores / categories make sense.
- Slang, language, dev’s writing style might “inject” interesting flavors
- Most of the words in devs slang are having negative sentiment
  (null, error, bug), so don’t be surprised of horrible scores 🤪

NLP insights: PM/Lead Toolset 2.0

Detect when commit messages indicate positive or negative emotions

Identify key areas of focus in the codebase over time, such as bug fixes, refactoring, new features, etc.

Help with sprint planning by prioritizing types of work, and understanding overall development efforts.

Ensure standardized messaging across the team by highlighting vague or poorly written messages and promoting best practices.

Identify rogue commits, possible sabotage, or unintended bulk changes in the codebase.

Understand developer expertise areas to foster better task assignment and ownership of specific code regions.

Prioritize code reviews by recognizing commits that introduce significant functionality or impact multiple areas of the codebase.

Determine how work habits evolve over time, identify periods of technical debt management, and evaluate sprint efficiency.

Tag commits automatically with relevant labels (e.g., bug fixes, performance improvements), facilitating a searchable commit history.

Warn developers when their commits may require extra scrutiny or testing.

And more…

Pavel Perfilov. From code to Insight: Using NLP and Pattern Analysis in Git History


CONF42