Machine Learning for Text Analytics in Software Engineering: The Good, the Bad, and the Ugly

Massimiliano Di Penta
University of Sannio, Italy
dipenta@unisannio,it @mdipenta

Goal of the talk

Reporting and discussing three different experiences of using machine learning for SE-related textual documents

What went well, what not

Lessons we learned in training ML approaches, using tools out-of-the box, or even opting for quite simpler alternatives

Text processing… various alternatives

Treat documents as bag of words

Preprocessing → Weighting → Algebraic model (VSM, LSI, LDA..)

Leverage the sentence structure

NL Parsing → Pattern extraction → **Machine Learning Classifier**

Vector representation

Preprocessing → Extract Embedding → **Neural Network Classifier**

Text processing… various alternatives

Treat documents as bag of words

Preprocessing → Weighting → Algebraic model (VSM, LSI, LDA..)

Leverage the sentence structure

NL Parsing → Pattern extraction → **Machine Learning Classifier**

Vector representation

Preprocessing → Extract Embedding → Neural Network

The Good:
Classifying and clustering short SE text (users' reviews)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2017.2759112, IEEE Transactions on Software Engineering

Listening to the Crowd for the Release Planning of Mobile Apps

Simone Scalabrino, *Student Member, IEEE,* Gabriele Bavota, *Member, IEEE*
Barbara Russo, *Member, IEEE,* Rocco Oliveto, *Member, IEEE,* Massimiliano Di Penta *Member, IEEE*

Abstract—The market for mobile apps is getting bigger and bigger, and it is expected to be worth over 100 Billion dollars in 2020. To have a chance to succeed in such a competitive environment, developers need to build and maintain high-quality apps, continuously astonishing their users with the coolest new features. Mobile app marketplaces allow users to release reviews. Despite reviews are aimed at recommending apps among users, they also contain precious information for developers, reporting bugs and suggesting new features. To exploit such a source of information, developers are supposed to manually read user reviews, something not doable when hundreds of them are collected per day. To help developers dealing with such a task, we developed CLAP (**C**rowd **L**istener for rele**A**se **P**lanning), a web application able to (i) categorize user reviews based on the information they carry out, (ii) cluster together related reviews, and (iii) prioritize the clusters of reviews to be implemented when planning the subsequent app release. We evaluated all the steps behind CLAP, showing its high accuracy in categorizing and clustering reviews and the meaningfulness of the recommended prioritizations. Also, given the availability of CLAP as a working tool, we assessed its applicability in industrial environments.

Index Terms—Release Planning, Mobile Apps, Mining Software Repositories

1 Introduction

The wide diffusion of mobile devices is making the apps market a tremendous success. Developers can easily join such a market by publishing their apps in an online store, making them available for download to interested users. The five leading app stores (*i.e.,* Google Play[1], Apple App Store[2], Windows Store[3], Amazon Appstore[4], and BlackBerry

reported by users and/or by implementing recommended features). Past studies have shown that various indicators of the app quality, such as the used APIs change- and fault-proneness [9], [39], the presence of ads [35], or, in general, characteristics of the apps or of the device on which it is deployed [38] significantly correlate with the app rating,

[Scalabrino et al., TSE 2018]

Doesn’t work

It continuously crashes on my Samsung Galaxy S4.
Please fix.

gbavota | Samsung Galaxy S4 | 25 Apr 2016

Great app!

The best chess game available on the app store!
5 stars!

roliveto | Motorola Moto G | 26 Apr 2016

Good but can be improved

Good app. It would be good to add the possibility to play online with other users.

dipenta | Samsung Galaxy S7 | 29 Apr 2016

Doesn’t work

It continuously crashes on my Samsung Galaxy S4.
Please fix.

gbavota | Samsung Galaxy S4 | 25 Apr 2016

Great app!

The best chess game available on the app store!
5 stars!

roliveto | Motorola Moto G | 26 Apr 2016

Good but can be improved

Good app. It would be good to add the possibility to play online with other users.

dipenta | Samsung Galaxy S7 | 29 Apr 2016

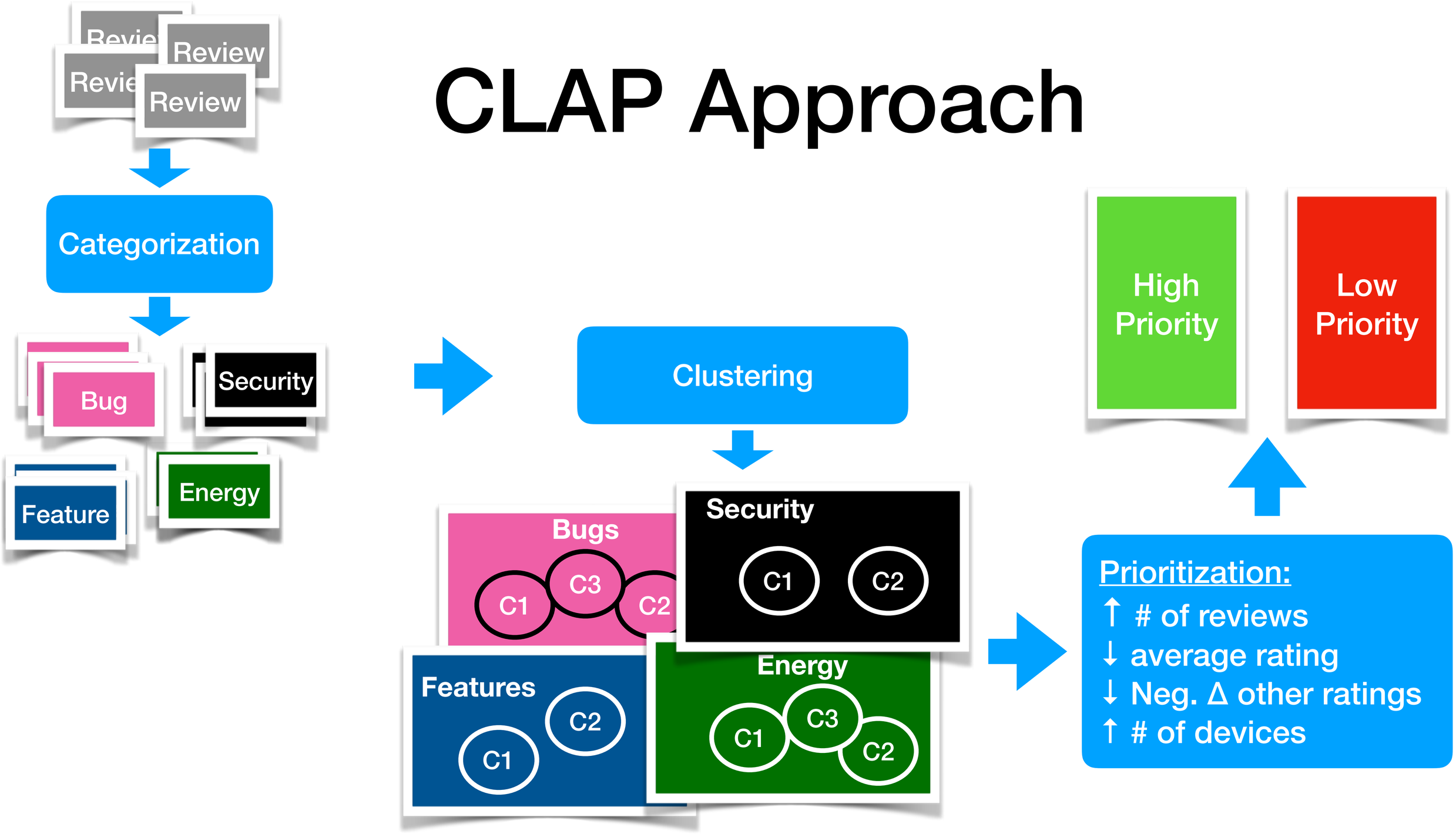
2.0M

3.0M

10M

>50M

CLAP Approach



Categorizing Reviews

Categorizing Reviews

Bug report	Suggestion for new feature	Security	Performance	Energy	Usability	Other

Predictor variables: rating, words and sentences

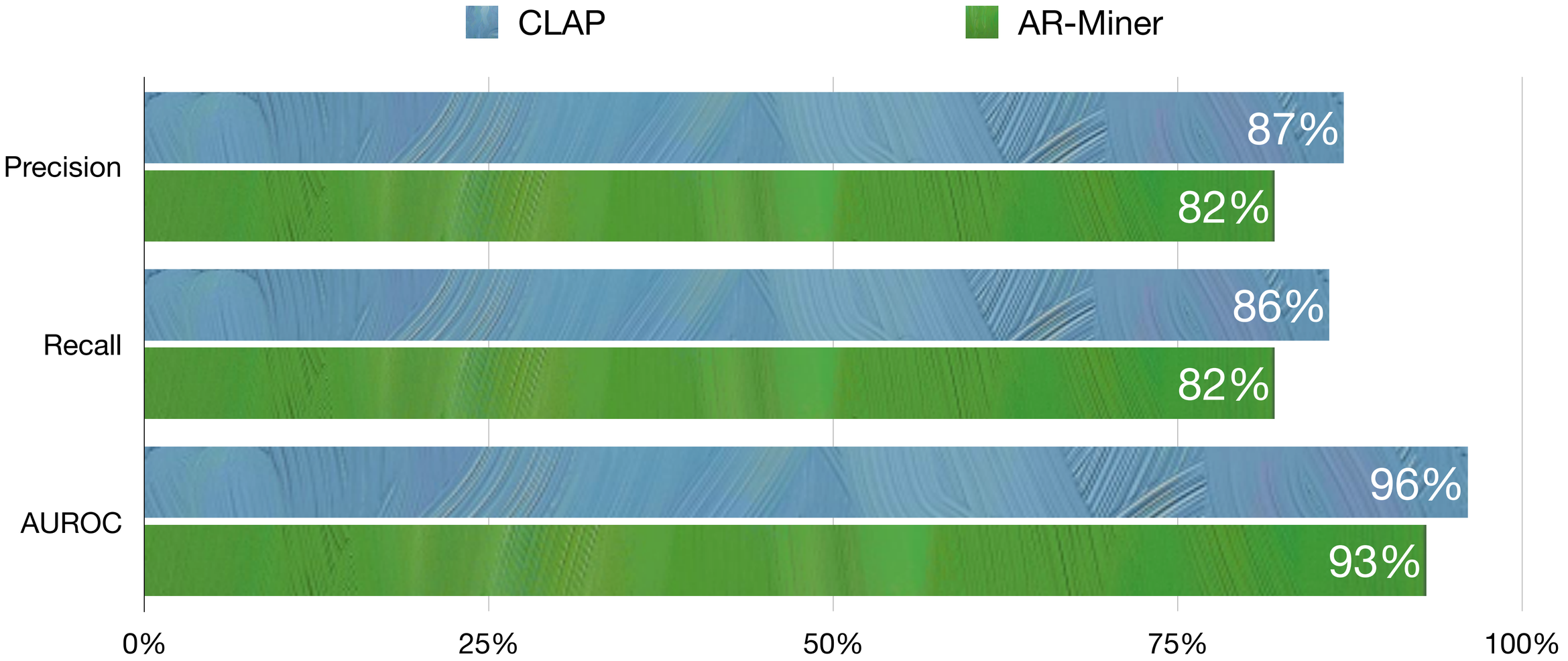
HANDLING NEGATIONS

The app is not very slow in Android

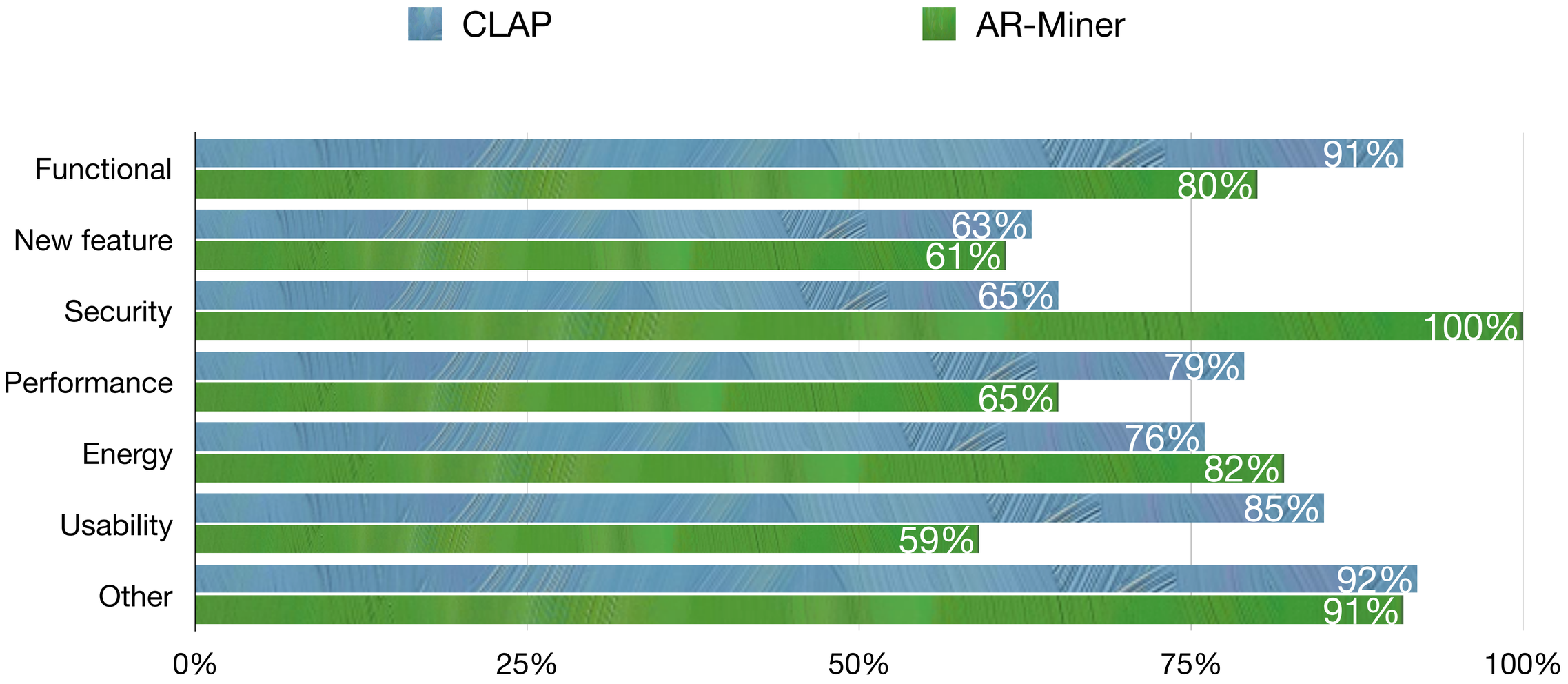
Categorizing Reviews

Categorizing Reviews

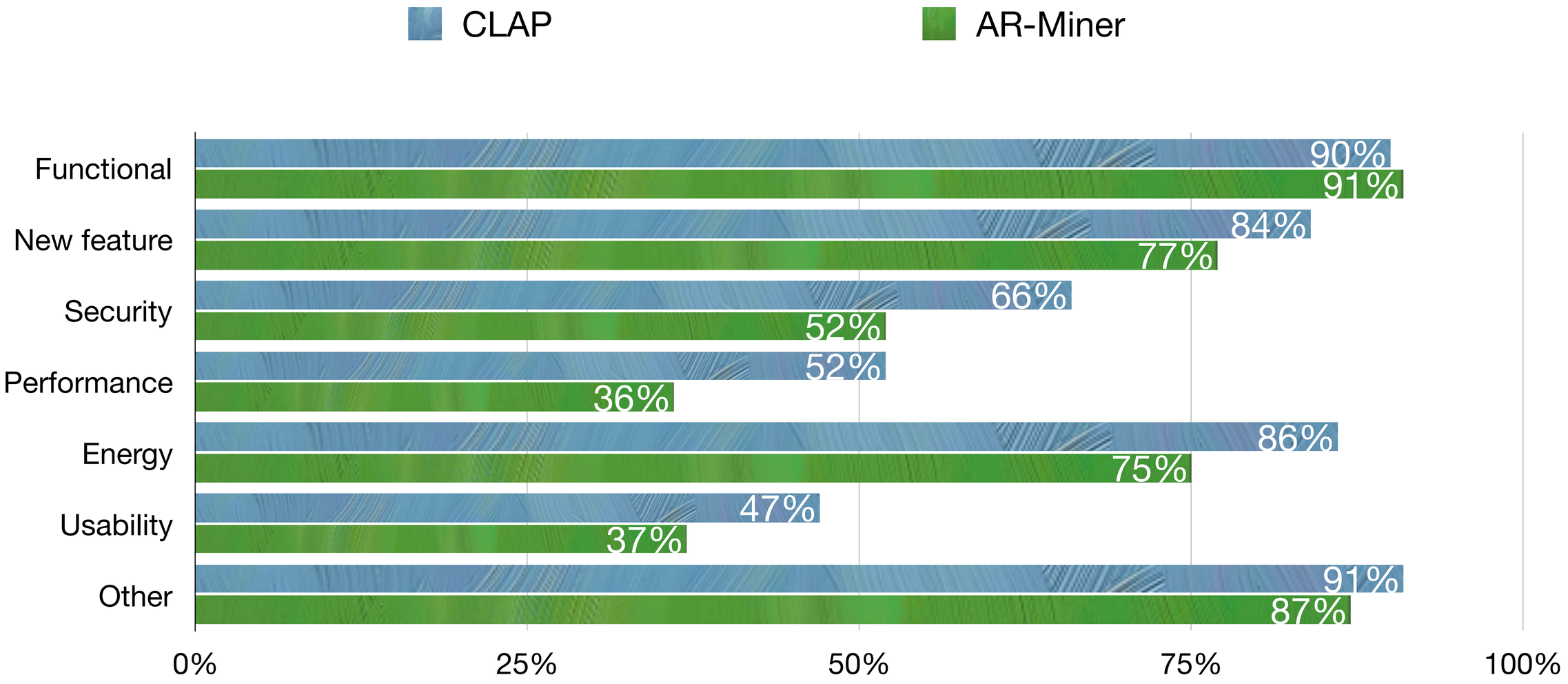
CLAP Classification Performances



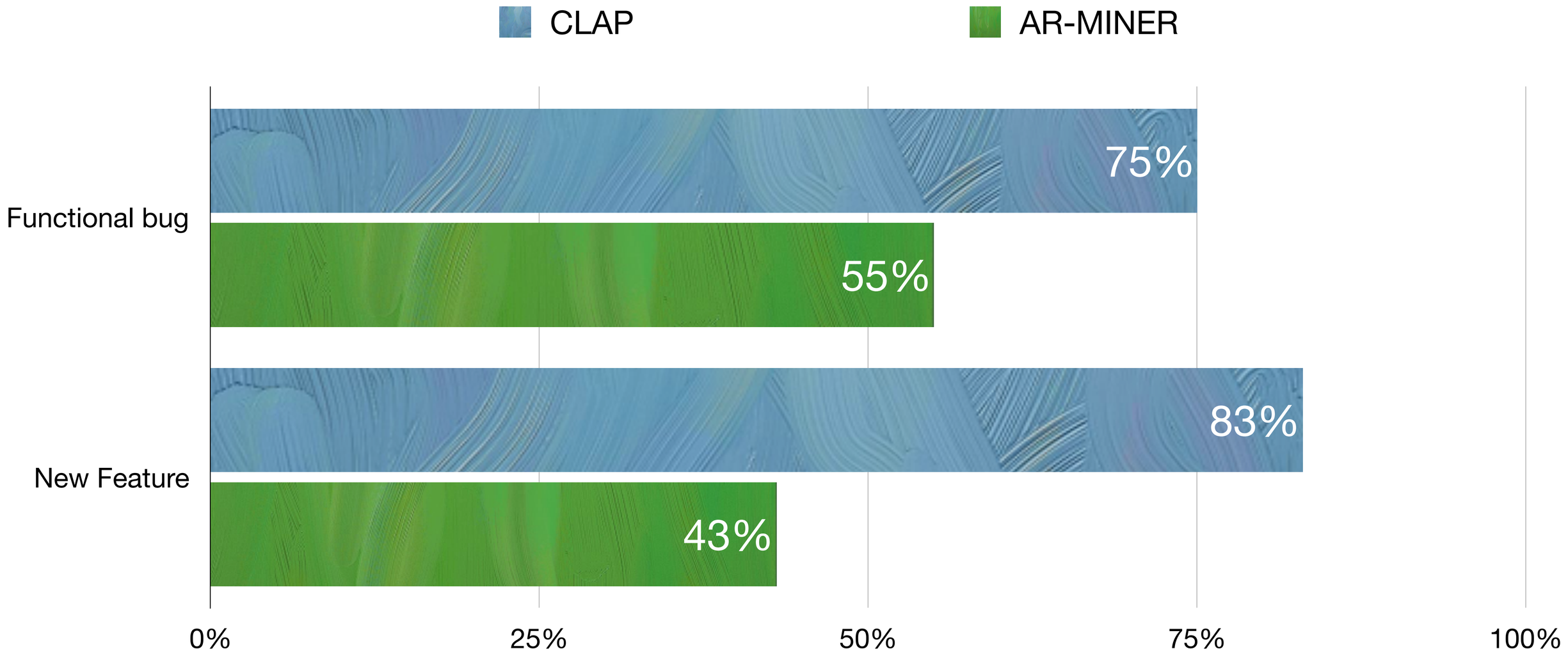
CLAP Classification Precision



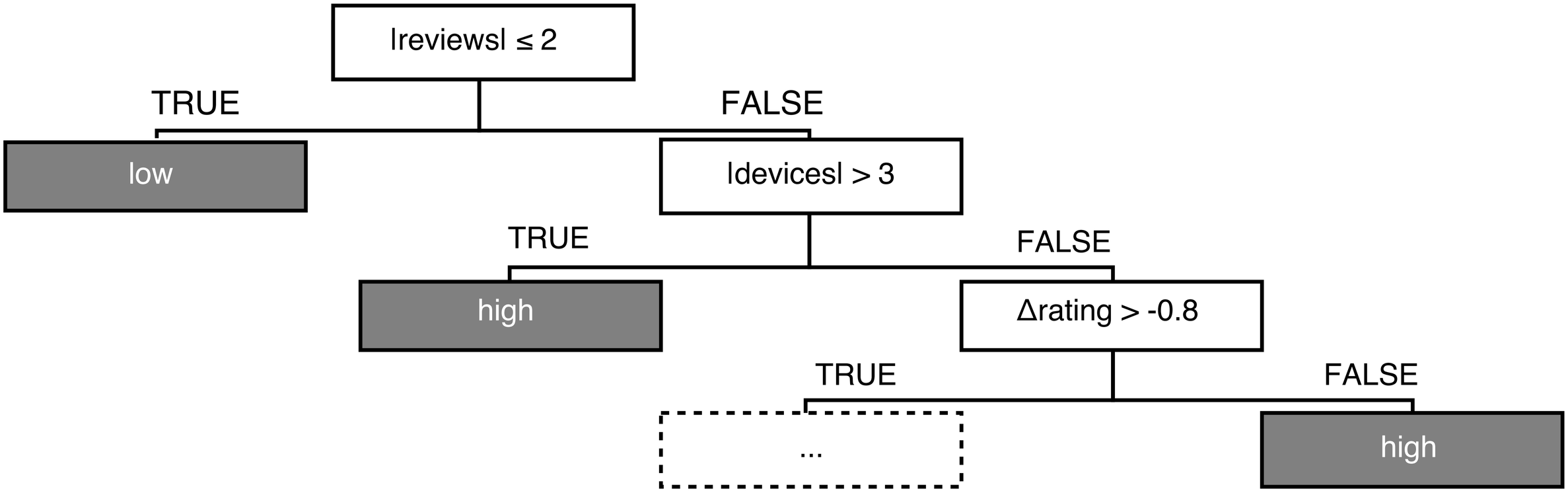
CLAP Classification Recall



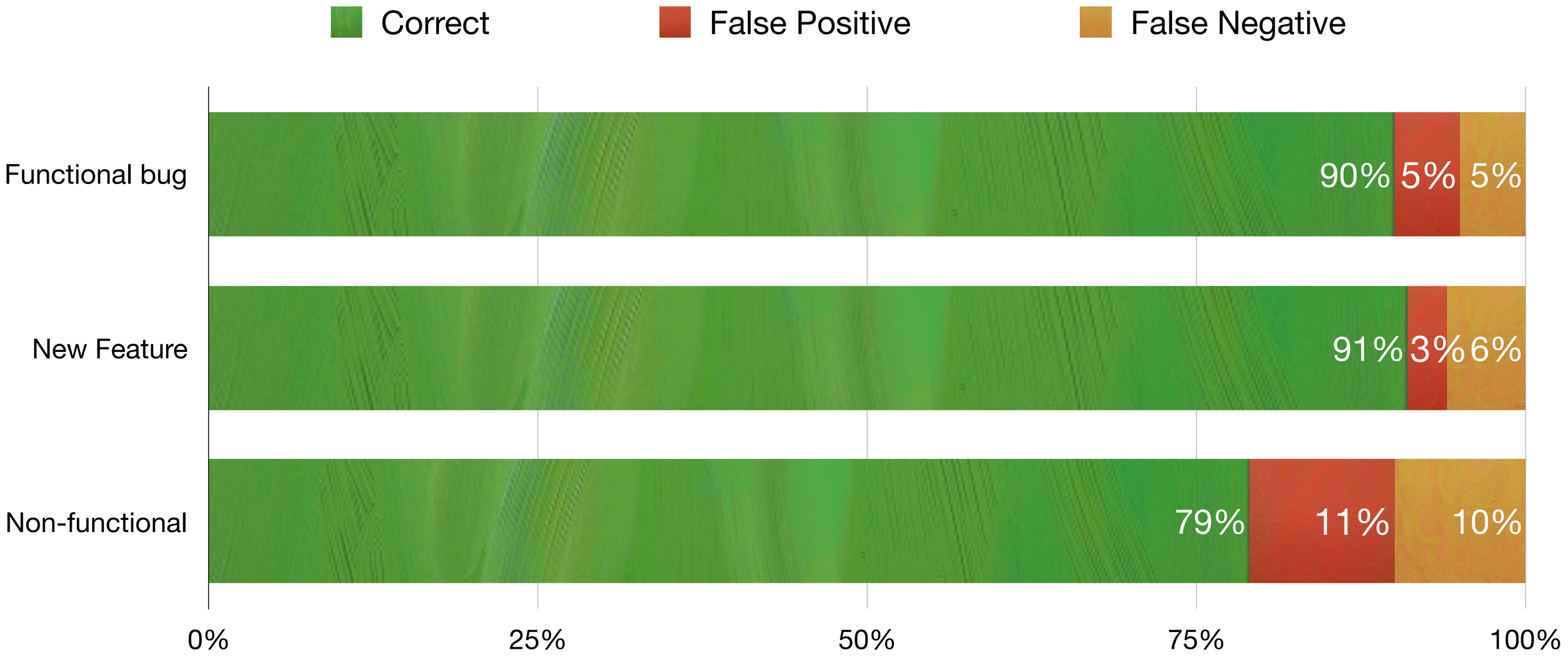
CLAP Clustering Accuracy



Prioritizing Review Clusters



CLAP Prioritization Accuracy



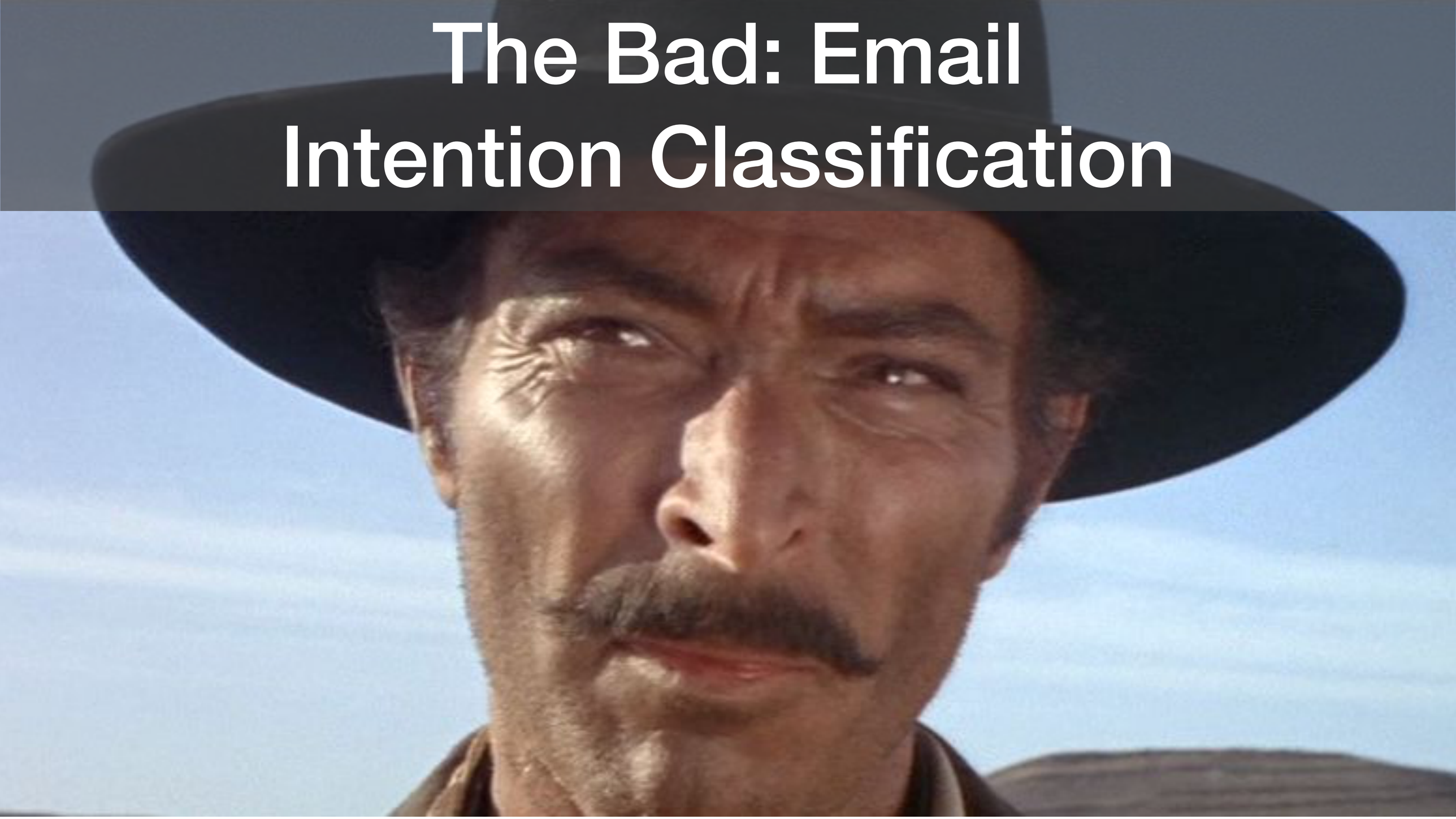
Lessons - I

Bag-of-words model augmented with negation handling and bigrams

Simple models worked out well enough

Easy to explain

The Bad: Email Intention Classification



2015 30th IEEE/ACM International Conference on Automated Software Engineering

Development Emails Content Analyzer: Intention Mining in Developer Discussions

Andrea Di Sorbo*, Sebastiano Panichella†, Corrado A. Visaggio*,
Massimiliano Di Penta*, Gerardo Canfora* and Harald C. Gall†
*University of Sannio, Benevento, Italy
†University of Zurich, Switzerland
disorbo@unisannio.it, panichella@ifi.uzh.ch, {visaggio,dipenta,canfora}@unisannio.it, gall@ifi.uzh.ch

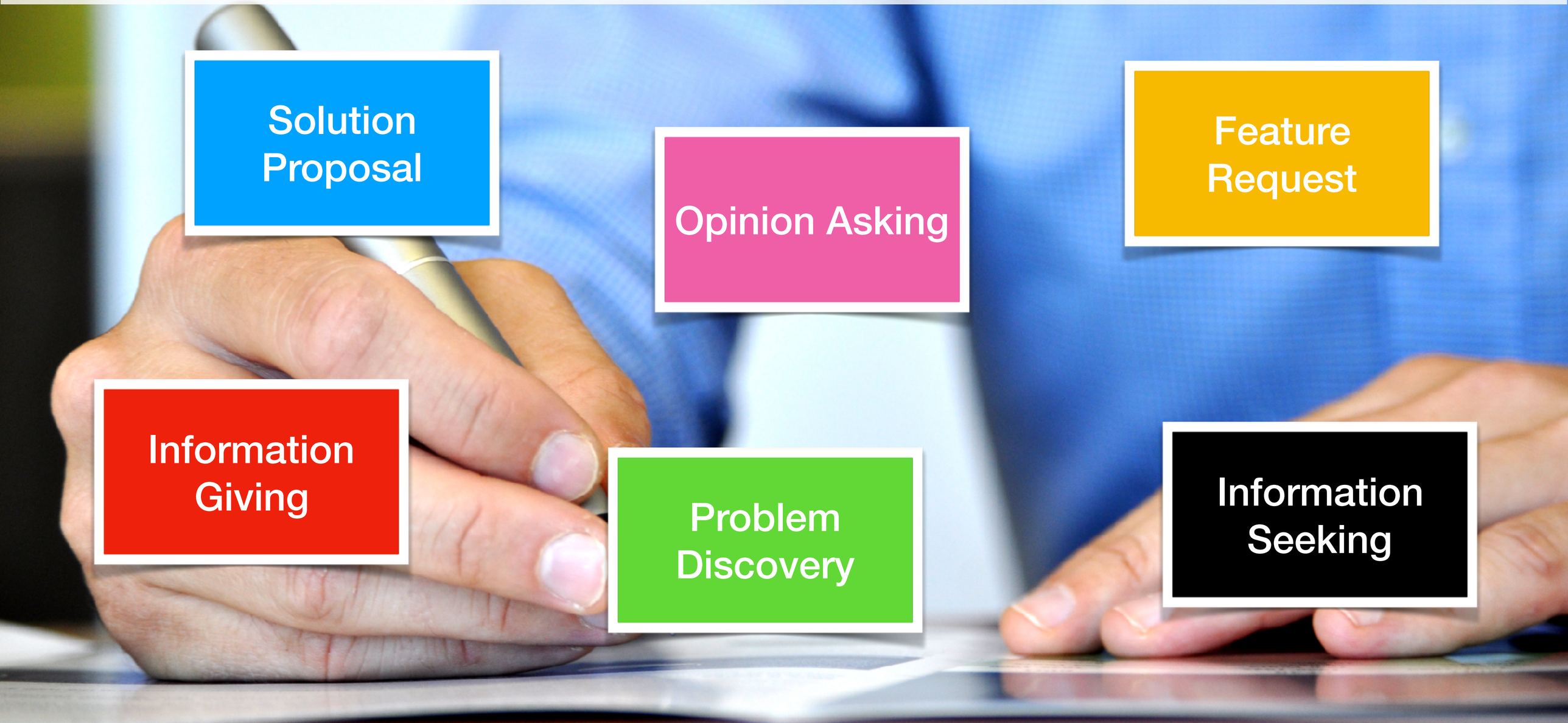
***Abstract*—Written development communication (e.g. mailing lists, issue trackers) constitutes a precious source of information to build recommenders for software engineers, for example aimed at suggesting experts, or at redocumenting existing source code. In this paper we propose a novel, semi-supervised approach named DECA (Development Emails Content Analyzer) that uses Natural Language Parsing to classify the content of development emails according to their purpose (e.g. feature request, opinion asking, problem discovery, solution proposal, information giving etc), identifying email elements that can be used for specific tasks. A study based on data from Qt and Ubuntu, highlights a high precision (90%) and recall (70%) of DECA in classifying email content, outperforming traditional machine learning strategies. Moreover, we successfully used DECA for re-documenting source code of Eclipse and Lucene, improving the recall, while keeping**

For example, an issue report may relate to a feature request, a bug, or just to a project management discussion. For example, Herzig *et al.* [30] and Antoniol *et al.* [2] found that over 30% of all issue reports are misclassified (i.e., rather than referring to a code fix, they resulted in a new feature, an update of documentation, or an internal refactoring). Hence, relying on such data to build fault prediction or localization approaches might result in incorrect results. Kochhar *et al.* [35] shed light on the need for additional cleaning steps to be performed on issue reports for improving bug localization tasks. This, for example, may involve a re-classification of issue reports.

On a different side, certain recommender may require to mine specific portions of a written communication, for example

[Di Sorbo et al., ASE 2015]

Email Intentions

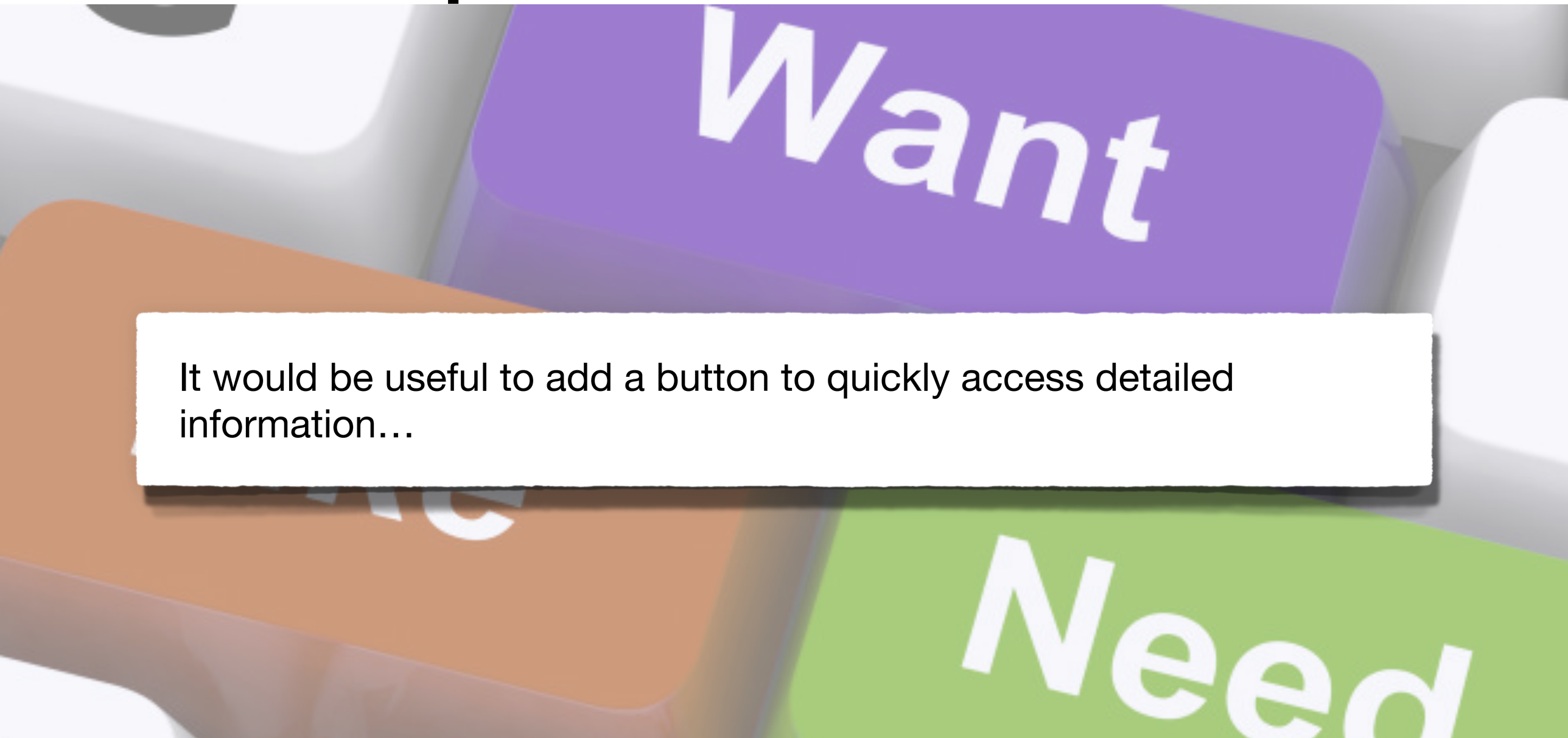



Identifying
Questions
Developers’
Ask

Mining Documentation

in TrecFTParser.parse(), you can extract the logic which finds the date and title into a common method which receives the string to look for parameters (e.g., find(String str, String start, int startlen, String end)

Requirement Elicitation

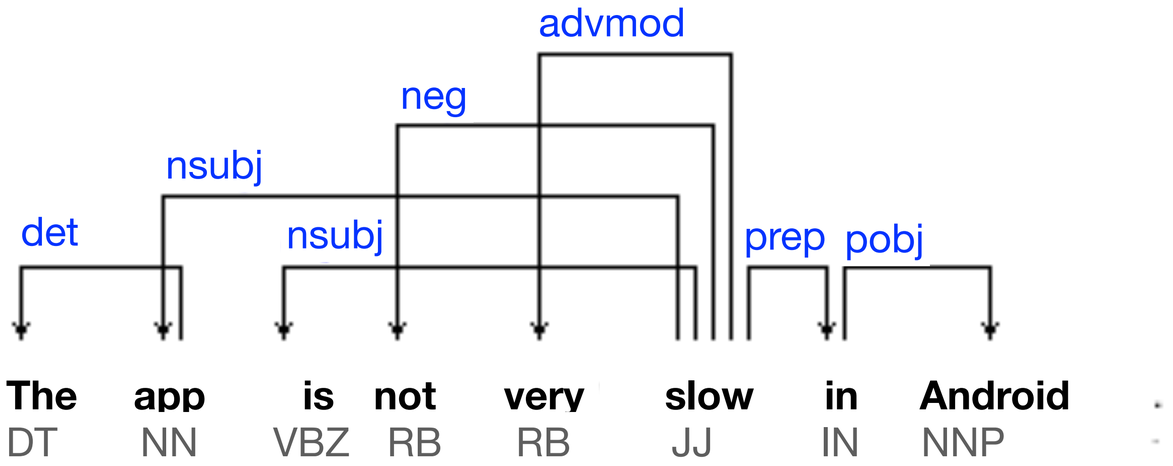


It would be useful to add a button to quickly access detailed information…

Pattern-based approach

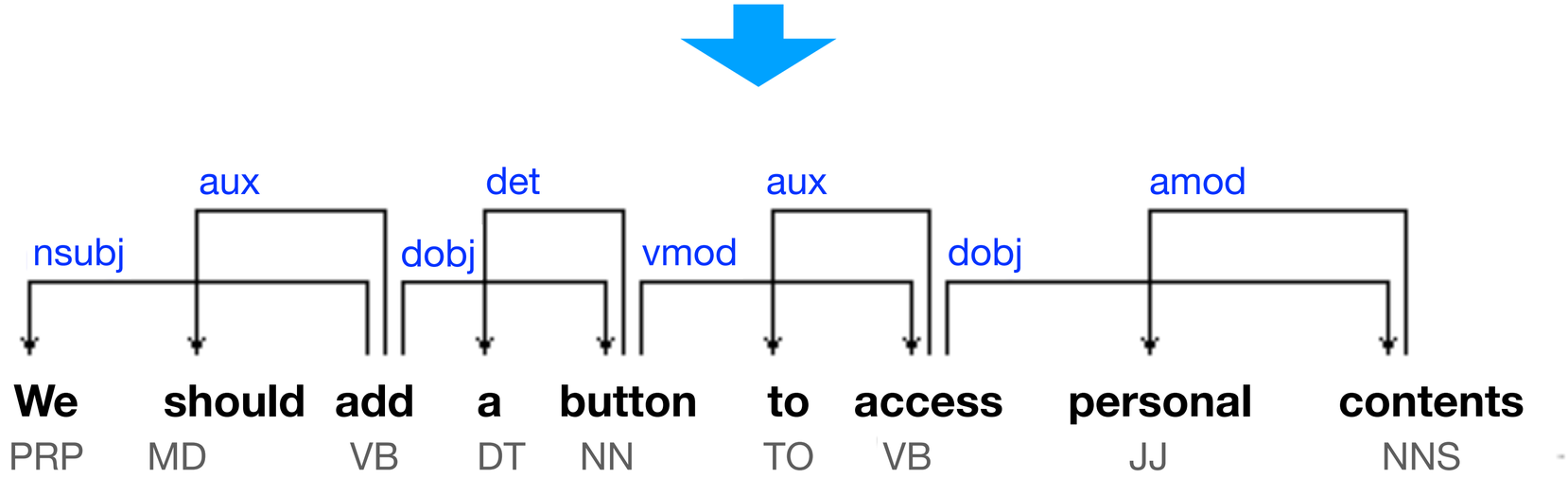
NL documents are not just bag of words, sentences have a structure

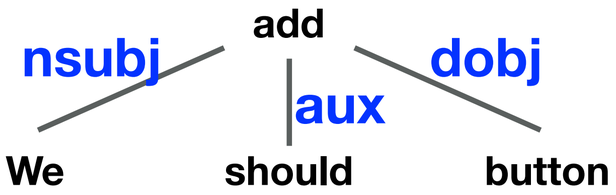
Stanford NL parser (or others such as NLTK) can extract it



Extracting patterns

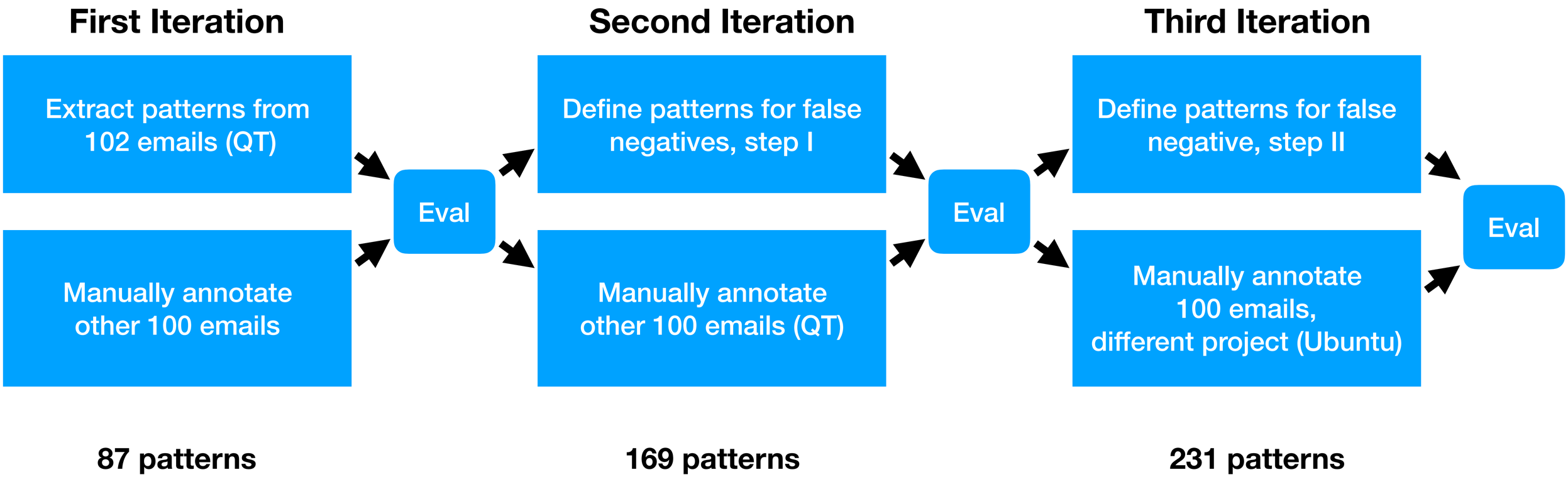
“We should add a button to access personal contents”





nsubj(add-3 [someone])
aux(add-3, should-2)
dobj(add-3, [something])

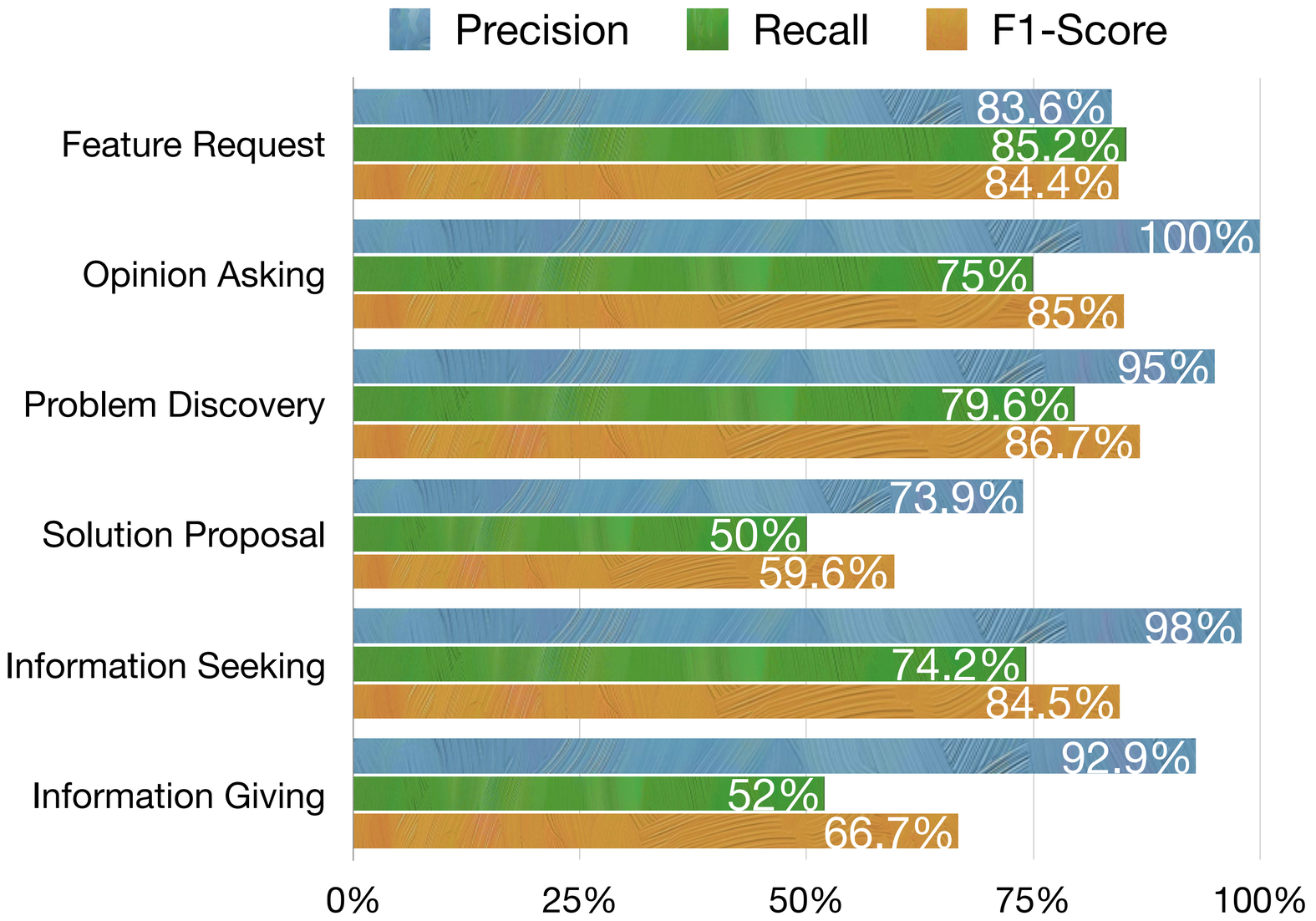
Pattern definition process



Defined patterns

Category	Patterns
Feature request	36
Opinion Asking	5
Problem Discovery	29
Solution Proposal	51
Information Seeking	57
Information Giving	53
Total	**231**

DECA: Achieved Performances



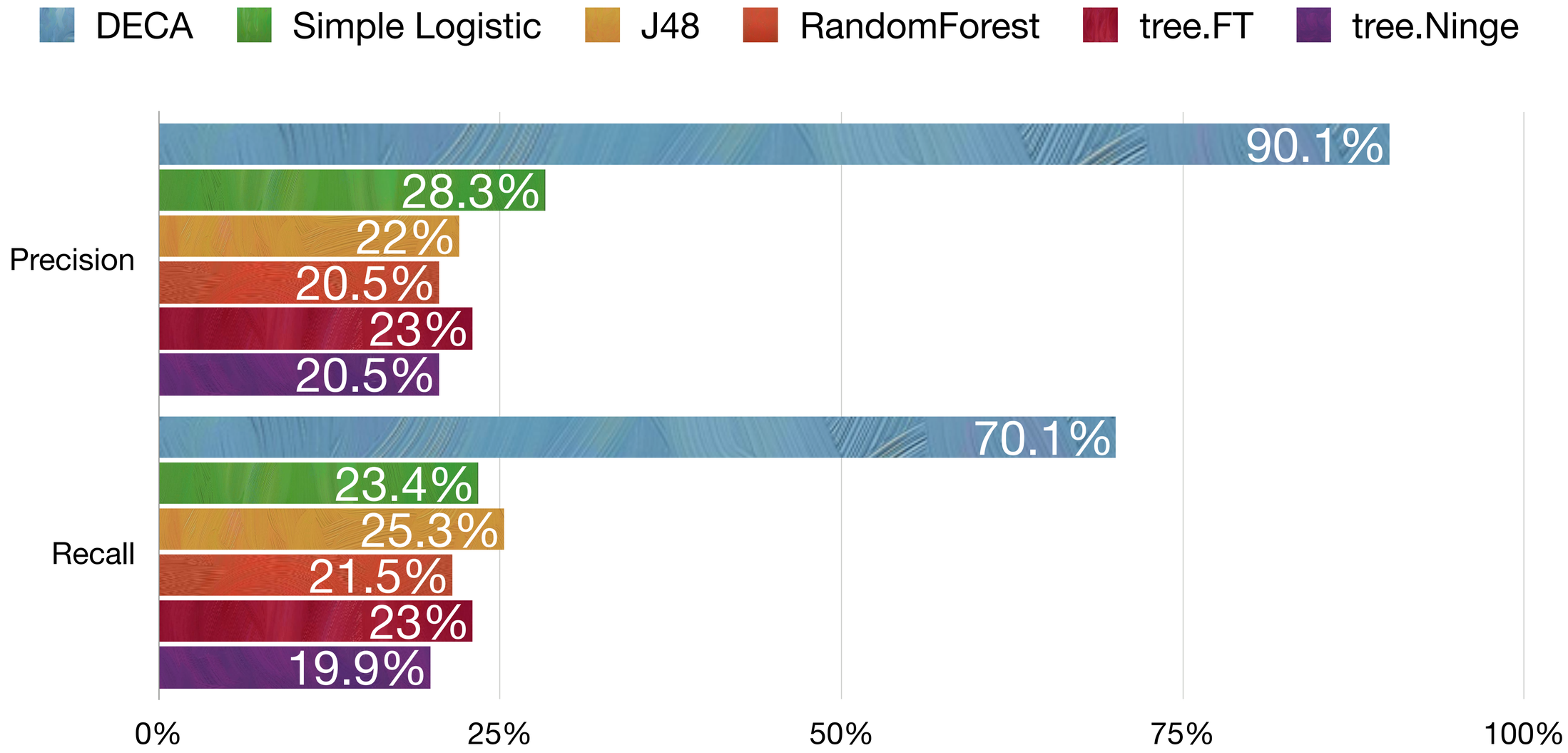
Overall:

Precision: 0.9

Recall: 0.7

F1-Score: 0.8

DECA: Comparison with ML



Examples of patterns

Feature Requests

[someone] **expects** [something] **to work**
[someone] **could/should add/provide/offer/integrate** [something]
[someone] **would love to see/use** [something]
[someone] **thinks** [something] **would be** [modal] **to have** [something]
[someone] **thinks** [something] **should** [verb]

Solution Proposal

[someone] **should/could recommend** [something]
[something] **isn't appropriate**
To make [something] **workable** [someone] **needs** [something]
[someone] **creates an extension of/to** [something]

Lessons - II

Rule-based models tell you exactly why something was classified in a certain way

Easy to link model capturing symptoms with solutions

Require work to be inferred → automated inference?

The Ugly: Training Opinion Miners for SE

Sentiment Analysis for Software Engineering: How Far Can We Go?

Bin Lin
Software Institute
Università della Svizzera italiana (USI)
Switzerland

Fiorella Zampetti
Department of Engineering
University of Sannio
Italy

Gabriele Bavota
Software Institute
Università della Svizzera italiana (USI)
Switzerland

Massimiliano Di Penta
Department of Engineering
University of Sannio
Italy

Michele Lanza
Software Institute
Università della Svizzera italiana (USI)
Switzerland

Rocco Oliveto
STAKE Lab
University of Molise
Italy

ABSTRACT

Sentiment analysis has been applied to various software engineering (SE) tasks, such as evaluating app reviews or analyzing developers' emotions in commit messages. Studies indicate that sentiment analysis tools provide unreliable results when used out-of-the-box, since they are not designed to process SE datasets. The *silver bullet* for a successful application of sentiment analysis tools to SE datasets might be their customization to the specific usage context.

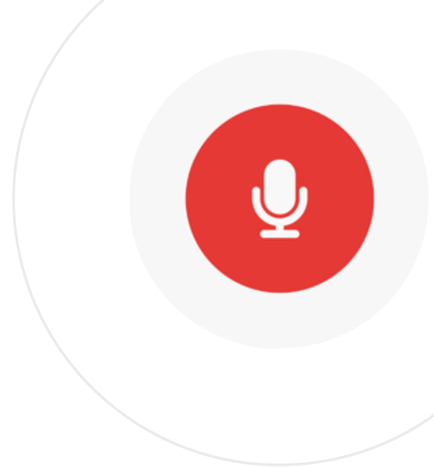
We describe our experience in building a software library recommender exploiting developers' opinions mined from Stack Overflow.

1 INTRODUCTION

Recent years have seen the rise of techniques and tools to automatically mine opinions from online sources [26]. The main application of these techniques is the identification of the mood and feelings expressed in textual reviews by customers (*e.g.*, to summarize the viewers' judgment of a movie [32]). Sentiment analysis [26] is a frequently used opinion mining technique. Its goal is to identify affective states and subjective opinions reported in sentences. In its basic usage scenario, sentiment analysis is used to classify customers' written opinions as negative, neutral, or positive.

[Lin et al., Proceedings of ICSE 2018]

Initial idea of API recommender system



Hi, my API recommender!

How can I help you?

What API can I use for parsing JSON files?

You can use Gson, Jackson, or JSON.simple.

Can you tell me more about Jackson?

Sure, here are pros and cons of using Jackson…

Google Scholar

sentiment analysis + software engineering

SentiStrength

social media

NLTK

social media

Stanford CoreNLP

movie reviews

What should we do?

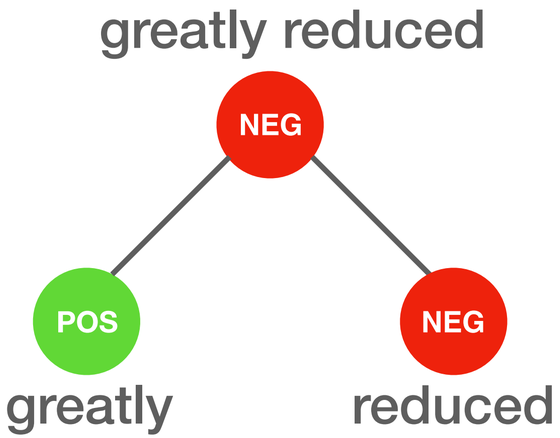
Retrain the tool!

Stanford CoreNLP

Retrain the tool!

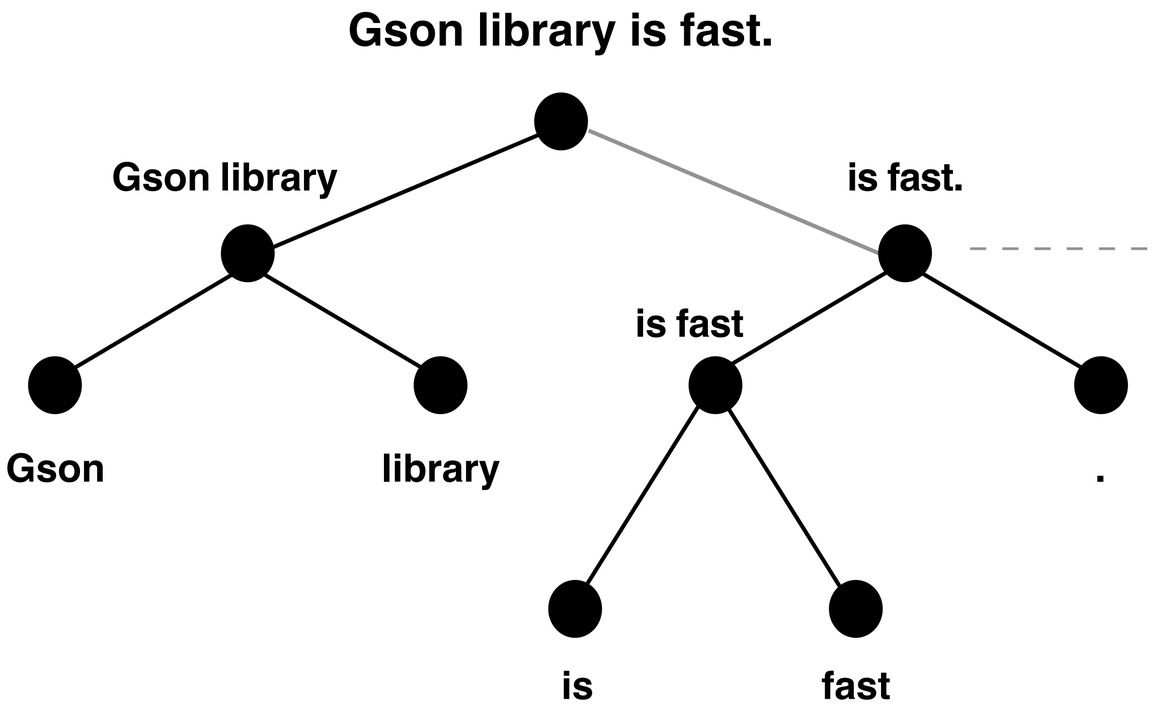
Stanford CoreNLP

Using this API greatly reduced my productivity.



how words compose the meaning of longer phrases

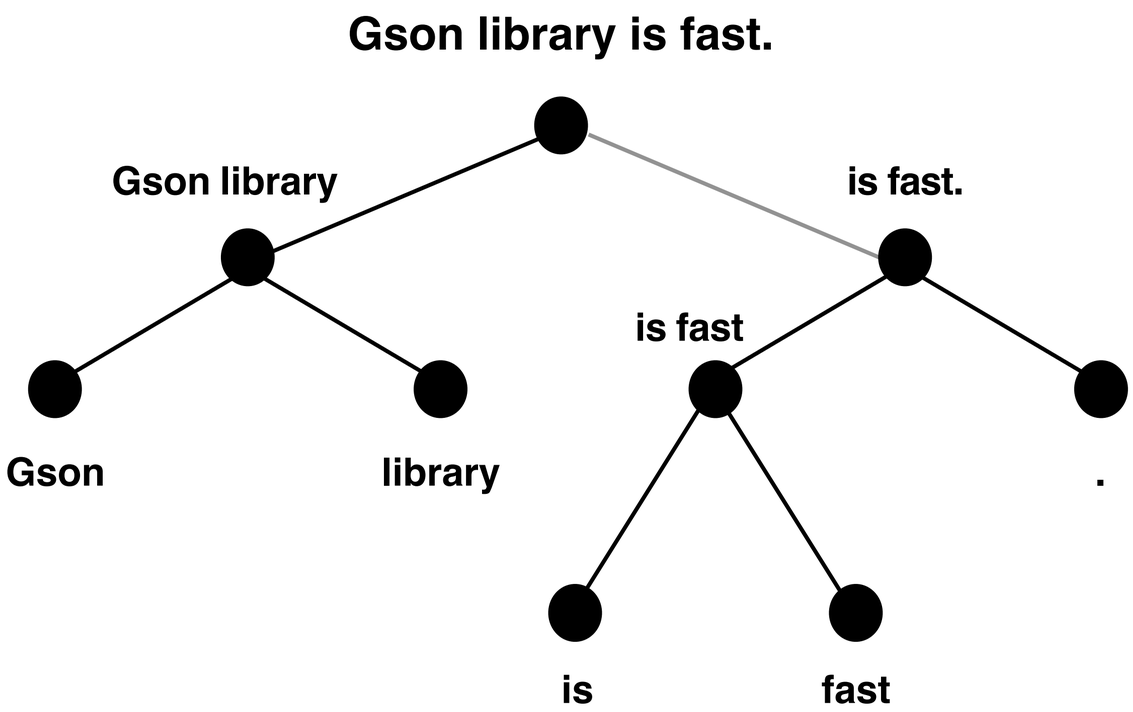
Our manual retraining



is fast.

- (●) positive
- ( ) slightly positive
- ( ) neutral
- ( ) slightly negative
- ( ) negative

Our manual retraining

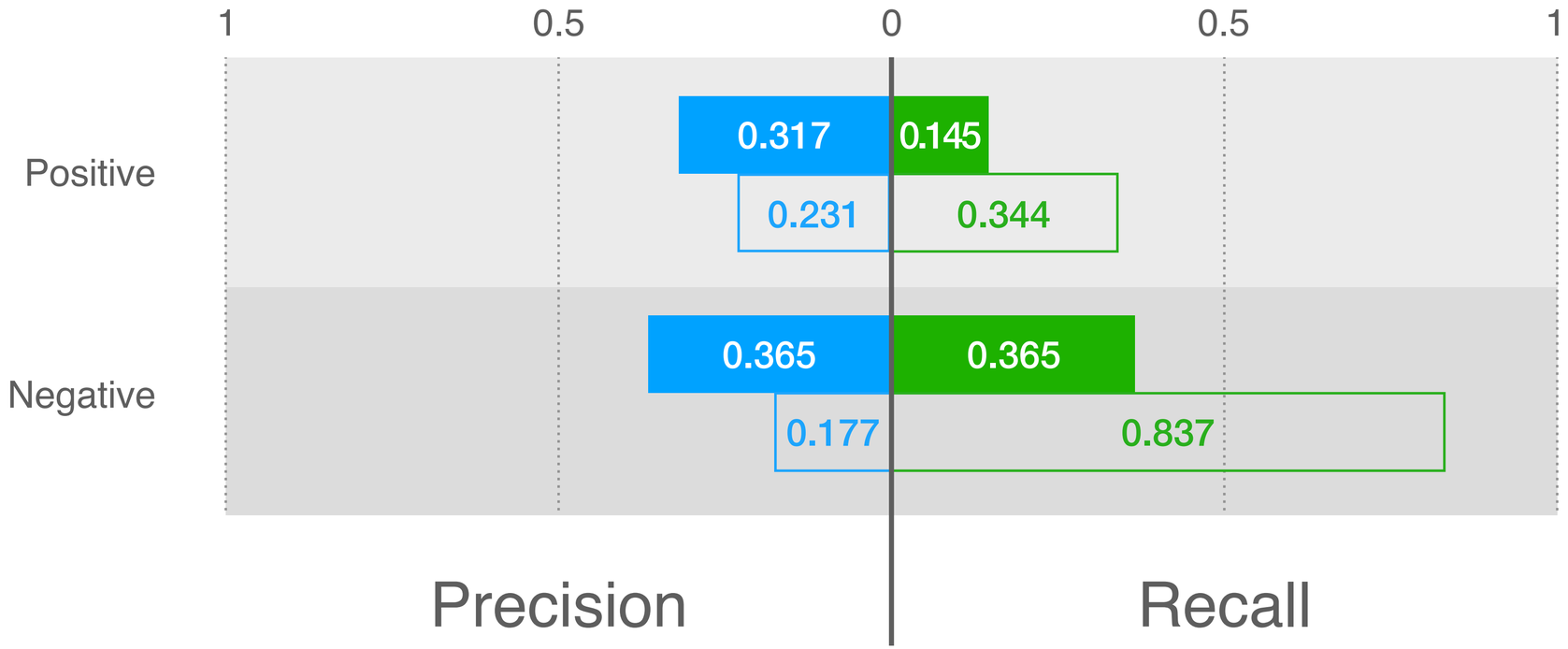


stackoverflow

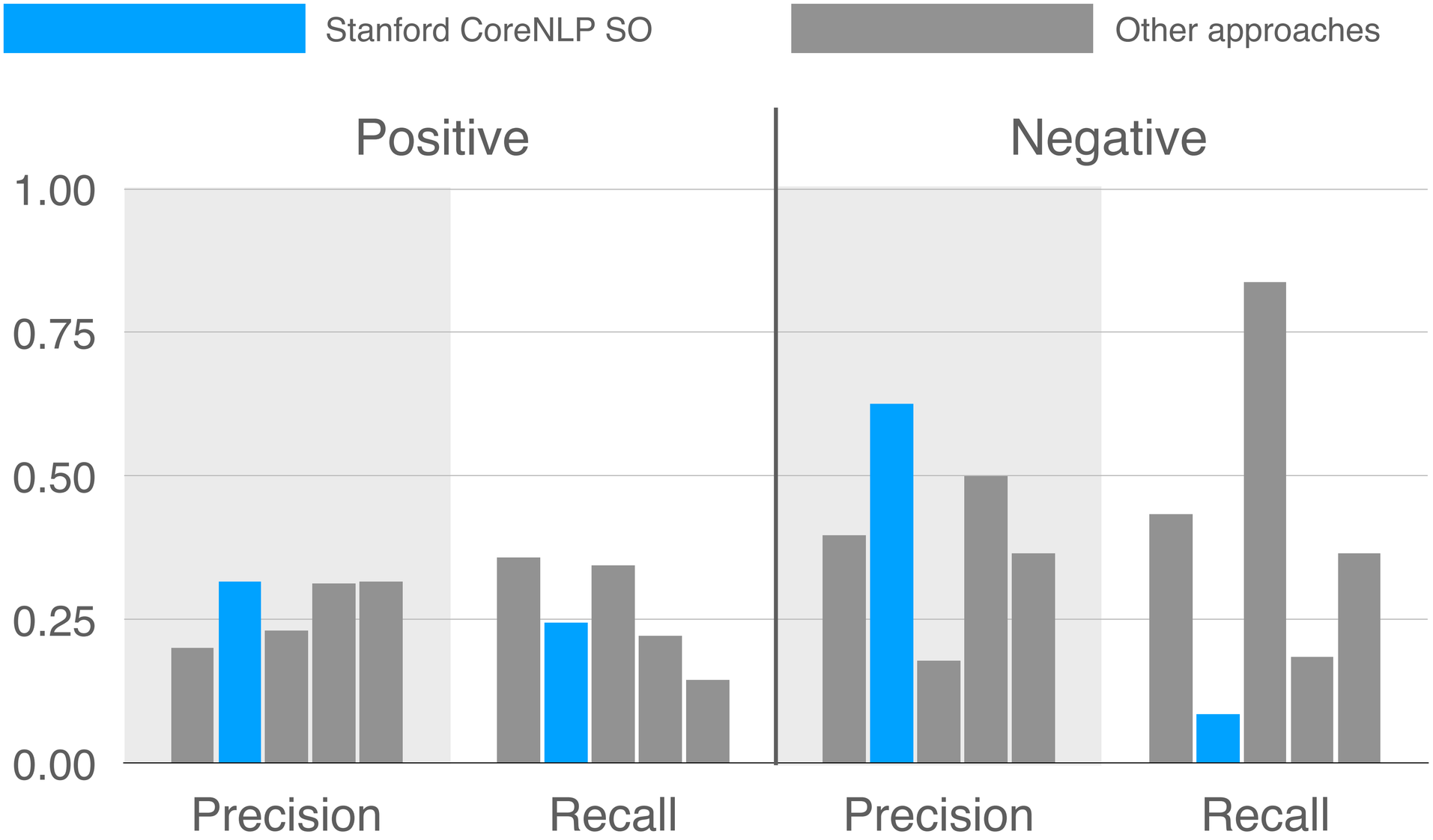
1.5k	sentences
20k * 2	nodes

Huge manual tagging over the summer!

Stanford CoreNLP SO vs Stanford CoreNLP



Stanford CoreNLP SO vs Other Approaches



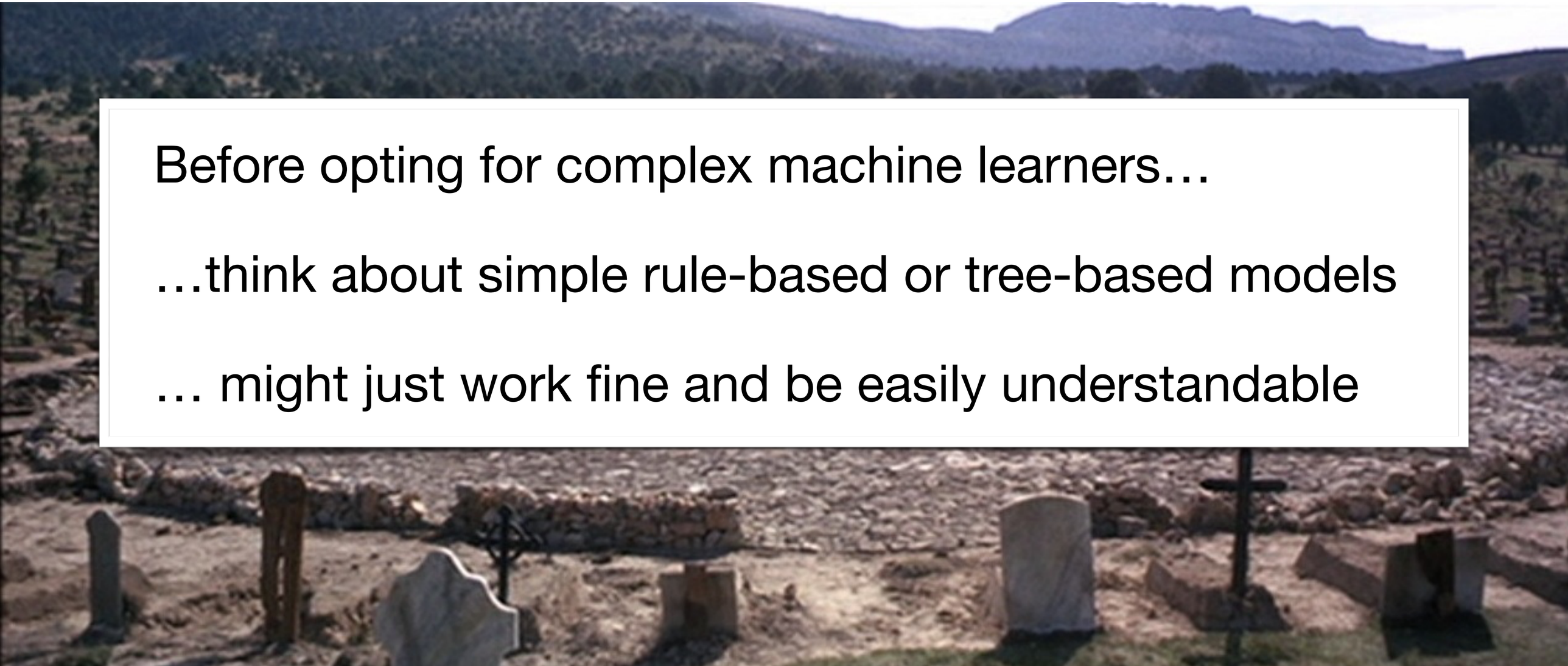
Lessons - III

Beware of using models trained in totally different contexts

Training requires a lot of time, highly subjective as well!

Sentiment analysis difficult for documents with mostly neutral sentences

Take Away

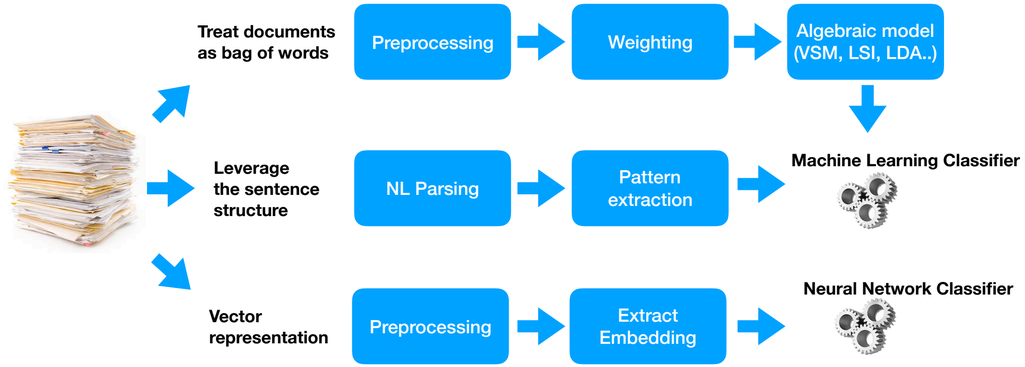


Before opting for complex machine learners…

…think about simple rule-based or tree-based models

… might just work fine and be easily understandable

Text processing… various alternatives



Lessons - I

- Bag-of-words model augmented with negation handling and bigrams
- Simple models worked out well enough
- Easy to explain

dipenta@unisannio.it

@mdipenta

Lessons - II

- Rule-based models tell you exactly why something was classified in a certain way
- Easy to link model capturing symptoms with solutions
- Require work to be inferred → automated inference?

Lessons - III

- Beware of using models trained in totally different contexts
- Training requires a lot of time, highly subjective as well!
- Sentiment analysis difficult for documents with mostly neutral sentences