Final Project

Option 2: p1 and p2

Ishan Aryendu

5/6/22

SSW 565: Software Architecture & Component-Based Design

Introduction

In most software systems, bugs are reported using issue trackers, and code modifications are incorporated using commits to source control repositories. As a result, it is simple to detect vulnerabilities by checking these basic artifacts of software development (a new bug report or commit) in real time. Furthermore, source code updates, bug reports, and commit comments frequently include extensive contextual information expressed in simple English that helps a researcher to determine whether the underlying item is linked to a quality trait. The most typical method for assessing a commit for relevant insights is to categorize the interactions by their subjects, bugs fixes, and feature enhancements.

When trying to improve products, determining the goal of a pull request is extremely useful because it addresses the question of why people are modifying the code. Choosing whether to utilize user commit messages, on the other hand, is often regarded as a classification problem. As a result, getting started will practically always necessitate a large amount of labeled data. For example, both Microsoft's LUIS and Google's Dialogflow assume that you can use either prebuilt domain labeled data or data that has previously been labeled.

But what if you don't have any previously tagged data? The length of the messages expressing the intent adds to the difficulty of having an unsupervised learning problem. After thinking about my topic for a bit

and treating it as an unsupervised learning problem, I used the pretrained models to generate the labels. In this article, I'll explain a method I discovered for automatically clustering short-text message data in order to identify and extract intents, as well as how to categorize new commits using previously classified data. To automate the identification procedure, I employed natural language processing and machine learning techniques on the acquired dataset, which contains a broad mix of security-related commits and bug reports from the OpenStack project.

Problem Statement

Let's establish what we're trying to do before we go any farther. I'd like to look into the following question:

Is there a way to get a helpful labeling of commit messages from an unlabeled set of commit messages in an automated way?

Because this is an unsupervised task and identifying commits can be subjective, I don't expect to discover a perfect solution. Can we do anything to provide preliminary insights before committing to time-consuming manual labeling, just as auto-EDA libraries aren't thorough but can provide a good starting point when confronted with new data? It's possible that the automated results will suffice, saving a week or

more of manually labeling data. It could also help to speed up the labeling process by serving as a helpful starting point.

Related Work

Static analysis is a method of inspecting source code or binaries without having to run them. A large amount of effort has gone into examining software written in high-level languages like C, C++, Perl, PHP, and Python. These analyzers, on the other hand, are language-specific, and they may fail to identify something even in supported languages. RATS, for example, does not detect Cross-Site Scripting (XSS) or SQL Injection flaws. Furthermore, when used in real-world projects, these techniques generate many false positives that are difficult to eliminate.

Dynamic analysis looks at source code by putting it through its paces using real-world inputs. Basic dynamic analysis (or testing) algorithms attempt a range of various inputs in order to find problems. Dynamic taint analysis tools are also available, which track taints in real time. For example, PHP Aspis employs dynamic taint analysis to detect XSS and SQL issues. ZAP, a commonly used industrial testing tool, detects specific types of internet application vulnerabilities. Before scanning, users must disable scan policies and then perform manual penetration testing.

Symbolic execution is a technique for testing a target system's multiple code paths. Rather than running the target system with concrete input values as in dynamic analysis, a symbolic execution engine replaces the inputs with symbolic variables that might be anything at first before executing the target system. KLEE, an open-source symbolic execution tool for analyzing programs and automatically creating high-code-coverage system input sets, is presented by Cadar. The source code must, however, be manually annotated and modified. As with most symbolic execution algorithms, runtime grows exponentially with the number of pathways in the program, resulting in path explosion.

This issue has also been investigated using machine learning techniques, in addition to previous techniques that focused solely on raw source code, provide an alternative to aid in the detection of vulnerabilities by mining context and semantic information outside of source code. Some of these publications [11, 16, 17, 20] are about discovering vulnerabilities, while others are about program analysis. While Shar focus on SQL injection and cross-site scripting, whereas Sahoo investigate malicious URL detection.

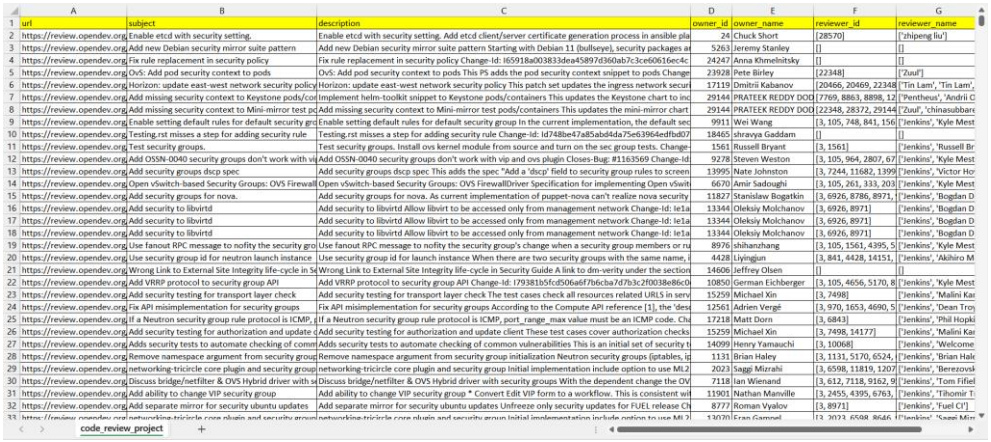
The works mentioned above are related to discovering new vulnerabilities in source code. But this project is limited to classifying the user commits into categories and classifying new commits into those categories.

Approach

I now propose the design of an unsupervised machine learning-based commit-message/bug-report-based topic identifier. The identifier retrieves and categorizes the topic from the dataset of commits/bug messages provided in the file. The new commits are then classified into pre-existing categories to facilitate identification.

Data

The csv file provided has the following information:



	A	B	C	D	E	F	G
1	url	subject	description	owner_id	owner_name	reviewer_id	reviewer_name
2	https://review.opendev.org	Enable etcd with security setting.	Enable etcd with security setting. Add etcd client/server certificate generation process in ansible pla	24	Chuck Short	[28570]	['zhipeng liu']
3	https://review.opendev.org	Add new Debian security mirror suite pattern	Add new Debian security mirror suite pattern Starting with Debian 11 (bullseye), security packages a	5263	Jeremy Stanley	[]	[]
4	https://review.opendev.org	Fix rule replacement in security policy	Fix rule replacement in security policy Change-Id: I65918a003833dea45897d360ab7c3ce60616ec4c	24247	Anna Khmelnitsky	[]	[]
5	https://review.opendev.org	OvS: Add pod security context to pods	OvS: Add pod security context to pods This PS adds the pod security context snippet to pods Change	23928	Pete Birley	[22348]	['Zuul']
6	https://review.opendev.org	Horizon: update east-west network security policy	Horizon: update east-west network security policy This patch set updates the ingress network secur	17119	Dmitrii Kabanov	[20466, 20469, 22348	['Tin Lam', 'Tin Lam',
7	https://review.opendev.org	Add missing security context to Keystone pods/co	Implement helm-toolkit snippet to Keystone pods/containers This updates the Keystone chart to inc	29144	PRATEEK REDDY DOO	[7769, 8863, 8898, 12	['Pentheus', 'Andrii C
8	https://review.opendev.org	Add missing security context to Mini-mirror test po	Add missing security context to Mini-mirror test pods/containers This updates the mini-mirror chart	29144	PRATEEK REDDY DOO	[22348, 28372, 29144	['Zuul', 'chinasubbare
9	https://review.opendev.org	Enable setting default rules for default security gro	Enable setting default rules for default security group In the current implementation, the default sec	9911	Wei Wang	[3, 105, 748, 841, 156	['Jenkins', 'Kyle Mest
10	https://review.opendev.org	Testing.rst misses a step for adding security rule	Testing.rst misses a step for adding security rule Change-Id: Id748be47a85abd4da75e63964edfbd07	18465	shrayya Gaddam	[]	[]
11	https://review.opendev.org	Test security groups.	Test security groups. Install ovs kernel module from source and turn on the sec group tests. Change-	1561	Russell Bryant	[3, 1561]	['Jenkins', 'Russell Br
12	https://review.opendev.org	Add OSSN-0040 security groups don't work with vi	Add OSSN-0040 security groups don't work with vip and ovs plugin Closes-Bug: #1163569 Change-Id	9278	Steven Weston	[3, 105, 964, 2807, 67	['Jenkins', 'Kyle Mest
13	https://review.opendev.org	Add security groups dscp spec	Add security groups dscp spec This adds the spec "Add a 'dscp' field to security group rules to screen	13995	Nate Johnston	[3, 7244, 11682, 1399	['Jenkins', 'Victor Ho
14	https://review.opendev.org	Open vSwitch-based Security Groups: OVS Firewall	Open vSwitch-based Security Groups: OVS FirewallDriver Specification for implementing Open vSwit	6670	Amir Sadoughi	[3, 105, 261, 333, 203	['Jenkins', 'Kyle Mest
15	https://review.opendev.org	Add security groups for nova.	Add security groups for nova. As current implementation of puppet-nova can't realize nova security	11827	Stanislaw Bogatkin	[3, 6926, 8786, 8971,	['Jenkins', 'Bogdan D
16	https://review.opendev.org	Add security to libvirtd	Add security to libvirtd Allow libvirt to be accessed only from management network Change-Id: Ie1a	13344	Oleksiy Molchanov	[3, 6926, 8971]	['Jenkins', 'Bogdan D
17	https://review.opendev.org	Add security to libvirtd	Add security to libvirtd Allow libvirt to be accessed only from management network Change-Id: Ie1a	13344	Oleksiy Molchanov	[3, 6926, 8971]	['Jenkins', 'Bogdan D
18	https://review.opendev.org	Add security to libvirtd	Add security to libvirtd Allow libvirt to be accessed only from management network Change-Id: Ie1a	13344	Oleksiy Molchanov	[3, 6926, 8971]	['Jenkins', 'Bogdan D
19	https://review.opendev.org	Use fanout RPC message to nofity the security gro	Use fanout RPC message to nofity the security group's change when a security group members or ru	8976	shihanzhang	[3, 105, 1561, 4395, 5	['Jenkins', 'Kyle Mest
20	https://review.opendev.org	Use security group id for neutron launch instance	Use security group id for launch instance When there are two security groups with the same name,	4428	Liyingjun	[3, 841, 4428, 14151,	['Jenkins', 'Akihiro M
21	https://review.opendev.org	Wrong Link to External Site Integrity life-cycle in Se	Wrong Link to External Site Integrity life-cycle in Security Guide A link to dm-verity under the section	14606	Jeffrey Olsen	[]	[]
22	https://review.opendev.org	Add VRRP protocol to security group API	Add VRRP protocol to security group API Change-Id: I79381b5fcd506a6f7b6cba7d7b3c2f0038e86c0	10850	German Eichberger	[3, 105, 4656, 5170, 8	['Jenkins', 'Kyle Mest
23	https://review.opendev.org	Add security testing for transport layer check	Add security testing for transport layer check The test cases check all resources related URLS in serv	15259	Michael Xin	[3, 7498]	['Jenkins', 'Malini Kar
24	https://review.opendev.org	Fix API misimplementation for security groups	Fix API misimplementation for security groups According to the Compute API reference [1], the 'desc	12561	Adrien Vergé	[3, 970, 1653, 4690, 5	['Jenkins', 'Dean Troy
25	https://review.opendev.org	If a Neutron security group rule protocol is ICMP, p	If a Neutron security group rule protocol is ICMP, port_range_max value must be an ICMP code. Cha	17218	Matt Dorn	[3, 6843]	['Jenkins', 'Phil Hopki
26	https://review.opendev.org	Add security testing for authorization and update c	Add security testing for authorization and update client These test cases cover authorization checks	15259	Michael Xin	[3, 7498, 14177]	['Jenkins', 'Malini Kar
27	https://review.opendev.org	Adds security tests to automate checking of comm	Adds security tests to automate checking of common vulnerabilities This is an initial set of security t	14099	Henry Yamauchi	[3, 10068]	['Jenkins', 'Welcome
28	https://review.opendev.org	Remove namespace argument from security group	Remove namespace argument from security group initialization Neutron security groups (iptables, ip	1131	Brian Haley	[3, 1131, 5170, 6524,	['Jenkins', 'Brian Hale
29	https://review.opendev.org	networking-tricircle core plugin and security group	networking-tricircle core plugin and security group Initial implementation include option to use ML2	2023	Saggi Mizrahi	[3, 6598, 11819, 1207	['Jenkins', 'Berezovsk
30	https://review.opendev.org	Discuss bridge/netfilter & OVS Hybrid driver with s	Discuss bridge/netfilter & OVS Hybrid driver with security groups With the dependent change the OV	7118	Ian Wienand	[3, 612, 7118, 9162, 9	['Jenkins', 'Tom Fifiel
31	https://review.opendev.org	Add ability to change VIP security group	Add ability to change VIP security group * Convert Edit VIP form to a workflow. This is consistent wi	11901	Nathan Manville	[3, 2455, 4395, 6763,	['Jenkins', 'Tihomir T
32	https://review.opendev.org	Add separate mirror for security ubuntu updates	Add separate mirror for security ubuntu updates Unfreeze only security updates for FUEL release Ch	8777	Roman Vyalov	[3, 8971]	['Jenkins', 'Fuel CI']
33	https://review.opendev.org	networking-tricircle core plugin and security group	networking-tricircle core plugin and security group Initial implementation include option to use ML2	13070	Eran Gampel	[3, 2023, 6598, 8646,	['Jenkins', 'Saggi Miz

code_review_project

Figure 1. Provided dataset of commits messages from OpenStack repository

The commit description piques our interest because the subject lines are too small and can be misleading. The description, on the other hand, contains much more information that can be used to infer the categories into which the commits can be classified.

Topic modeling

An unsupervised learning problem like this can be approached in a number of ways. When confronted with this situation, the first method that came to mind was topic modeling. It's a method for uncovering hidden subjects in a collection of documents.

Understanding which issues or features are being worked on is a critical first step toward automating the process of classifying commits based on quality attributes or security fixes. However, I don't expect to find a perfect solution because I'm using an unsupervised model, and the labelling of commit messages is quite subjective.

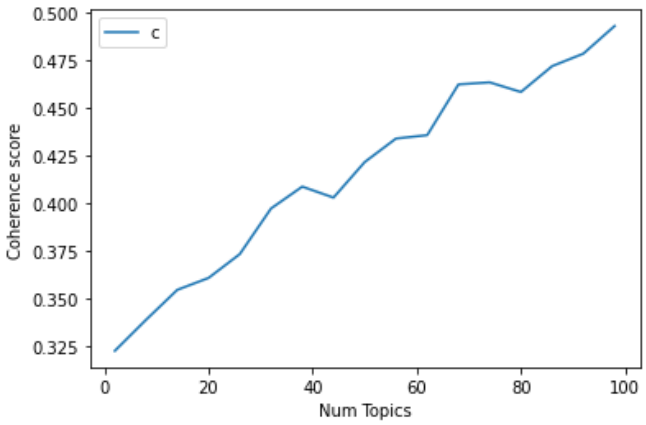
Having stated that, here are the steps for creating the model:

- Step 1: Find the relevant Data
- Step 2: Remove emails, tokenize words and Clean-up text
- Step 3: Create Bigram and Trigram Models
- Step 4: Lemmatize

Topic modeling can be done in a variety of ways, but Latent Dirichlet Allocation (LDA) is one of the most used models. LDA is a probabilistic generative model in which each document is composed of a distribution of a set of subjects, each of which is composed of a set of words. Determining how many topics to use, which is an important model hyperparameter when using LDA (and many other topic modeling approaches), is a key challenge. A higher coherence score suggests that the learned subjects are of higher quality. Coherence is a

metric that analyzes how comparable the words in each topic are. Gensim, a popular subject modeling package, makes calculating model coherence simple. Unfortunately, with the commits we're working with, determining how many subjects to choose using coherence was difficult:

```
[ ]  plt.plot(x, coherence_values)
     plt.xlabel("Num Topics")
     plt.ylabel("Coherence score")
     plt.legend(("coherence_values"), loc='best')
     plt.show()
```




```
[ ]  # Print the coherence scores
     for m, cv in zip(x, coherence_values):
         print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

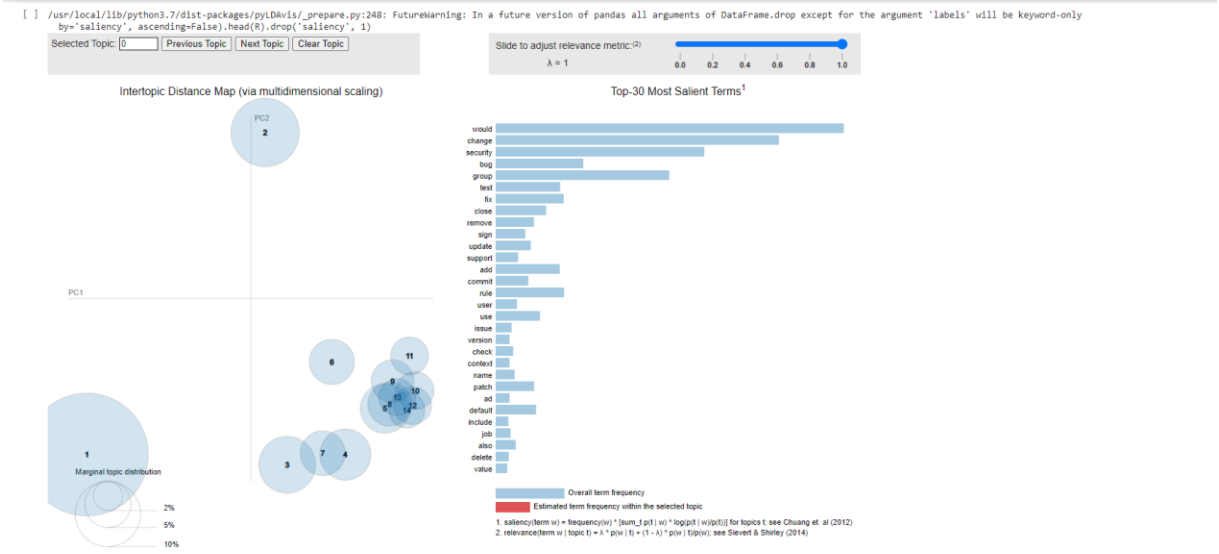
```
Num Topics = 2  has Coherence Value of 0.3225
Num Topics = 8  has Coherence Value of 0.3388
Num Topics = 14  has Coherence Value of 0.3545
Num Topics = 20  has Coherence Value of 0.3608
Num Topics = 26  has Coherence Value of 0.3733
Num Topics = 32  has Coherence Value of 0.3973
Num Topics = 38  has Coherence Value of 0.4088
Num Topics = 44  has Coherence Value of 0.403
Num Topics = 50  has Coherence Value of 0.4217
Num Topics = 56  has Coherence Value of 0.434
Num Topics = 62  has Coherence Value of 0.4358
Num Topics = 68  has Coherence Value of 0.4625
Num Topics = 74  has Coherence Value of 0.4635
Num Topics = 80  has Coherence Value of 0.4585
Num Topics = 86  has Coherence Value of 0.472
Num Topics = 92  has Coherence Value of 0.4786
Num Topics = 98  has Coherence Value of 0.4931
```

Raising the number of subjects appears to improve the dataset's coherence, but this leaves us with little guidance on how many topics to choose. For example, I can choose each data point to be a topic or choose all of the points as single topic and still it would be a valid choice using this model.

Topic models can be [difficult to interpret], which adds to the difficulties. Consider the following examples of identified topics:

```
[ ]  # Select the model and print the topics
     optimal_model = model_list[2]
     model_topics = optimal_model.show_topics(formatted=False)
     pprint(optimal_model.print_topics(num_words=10))
```

```
[(0,
  '0.373*"group" + 0.208*"security" + 0.051*"instance" + 0.030*"table" + '
  '0.025*"parameter" + 0.016*"show" + 0.015*"action" + 0.013*"query" + '
  '0.010*"single" + 0.010*"detail"'),
 (1,
  '0.095*"change" + 0.079*"user" + 0.052*"update" + 0.046*"patch" + '
  '0.044*"fix" + 0.044*"issue" + 0.042*"close" + 0.041*"project" + '
  '0.032*"commit" + 0.028*"bug"'),
 (2,
  '0.089*"support" + 0.074*"neutron" + 0.059*"network" + 0.055*"service" + '
  '0.051*"enable" + 0.047*"driver" + 0.036*"ca" + 0.036*"client" + '
  '0.021*"plugin" + 0.021*"command"'),
 (3,
  '0.142*"fix" + 0.067*"check" + 0.064*"bug" + 0.054*"case" + 0.047*"guide" + '
  '0.045*"error" + 0.037*"fail" + 0.026*"exception" + 0.023*"return" + '
  '0.021*"message"'),
 (4,
  '0.148*"security" + 0.146*"add" + 0.092*"change" + 0.072*"patch" + '
  '0.043*"code" + 0.030*"improve" + 0.025*"number" + 0.023*"type" + '
  '0.018*"performance" + 0.013*"validate"'),
 (5,
  '0.099*"security" + 0.097*"add" + 0.095*"context" + 0.074*"change" + '
  '0.051*"server" + 0.045*"implement" + 0.044*"container" + 0.041*"include" + '
  '0.026*"flag" + 0.026*"true"'),
 (6,
  '0.273*"rule" + 0.176*"group" + 0.105*"default" + 0.057*"create" + '
  '0.040*"security" + 0.031*"method" + 0.026*"creation" + 0.014*"quota" + '
  '0.013*"duplicate" + 0.012*"separate"'),
 (7,
  '0.139*"change" + 0.114*"update" + 0.053*"policy" + 0.046*"make" + '
  '0.025*"description" + 0.022*"object" + 0.022*"section" + 0.022*"field" + '
  '0.021*"security" + 0.015*"information"'),
 (8,
  '0.284*"security" + 0.209*"group" + 0.033*"relate" + 0.032*"exist" + '
  '0.023*"extension" + 0.021*"introduce" + 0.016*"current" + 0.016*"api" + '
  '0.015*"merge" + 0.015*"list"'),
 (9,
  '0.188*"port" + 0.170*"security" + 0.029*"agent" + 0.028*"disable" + '
  '0.025*"group" + 0.025*"tenant" + 0.024*"address" + 0.023*"handle" + '
```



While some of the subjects make sense, others are difficult to comprehend.

Clustering the embeddings

Clustering is a well-known technique for resolving the difficulties with unsupervised learning. To cluster text data, we'll need to make a lot of decisions, including how to process the data and which algorithms to apply.

To begin with, the text data must be numerically represented in order for the computer to operate on it. One method for clustering is to create embeddings, or vector representations, of each word. Because each message contains multiple words, one option is to simply average the individual word embeddings of each message's words.

There are several approaches to determining a single vector representation of a sentence. Google's Universal Sentence Encoder (USE), first published in 2018 by Cer, is a popular sentence embedding model. The USE model was trained on a diverse set of data, including Wikipedia, web news, web question-and-answer pages, and discussion forums, and it excels at task of sentence semantic similarity.

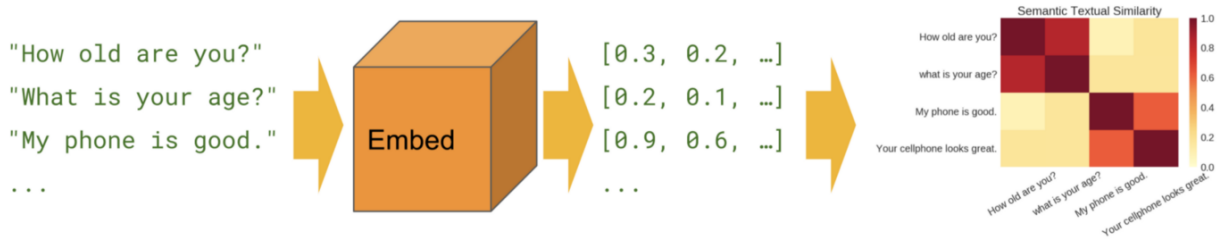


Figure 2. Universal Sentence Encoder model. Source: TensorFlow Hub (https://tfhub.dev/google/universal-sentence-encoder/4)

In a publication released in 2019, Reimers and Gurevych proposed Sentence-BERT, a "modification of the pretrained BERT network that uses Siamese and triplet network architectures to create semantically relevant sentence embeddings that can be compared using cosine-similarity." There are several Python implementations of these models trained on various datasets, which are available to download and use from HuggingFace.

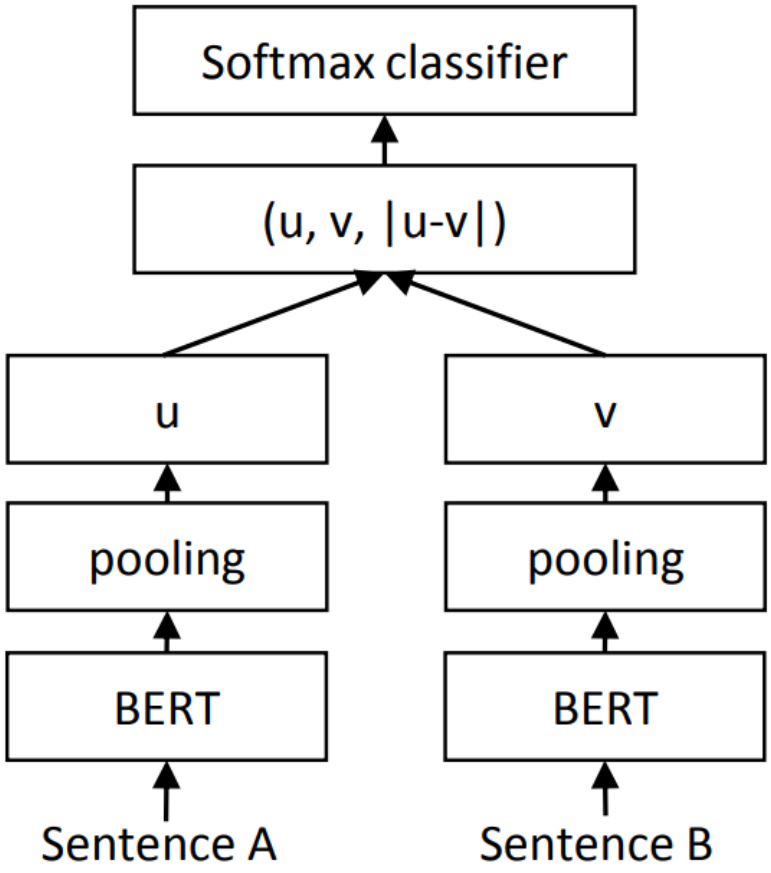


Figure 3. SBERT architecture with classification objective function

A Siamese neural network (sometimes called a twin neural network) is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors.

The triplet is formed by drawing an anchor input, a positive input that describes the same entity as the anchor entity, and a negative input that does not describe the same entity as the anchor entity. These

inputs are then run through the network, and the outputs are used in the loss function.

It is apparent from the image that BERT makes up the base of this model, to which a pooling layer has been appended. The pooling layer enables us to create a fixed-size representation for input sentences of varying lengths. The image shows the base architecture and objective function together which creates the SBERT model.

What is a Softmax classifier? Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

Due to the limited number of data in our dataset, it is preferable to use a pre-trained model. I'll compare the outcomes of four pre-trained sentence embedding models: USE and four distinct sentence-BERT models for this study (all-mpnet-base-v2, all-MiniLM-L6-v2, all-distilroberta-v1 and paraphrase-MiniLM-L6-v2).

```
model_st1 = SentenceTransformer('all-mpnet-base-v2')
model_st2 = SentenceTransformer('all-MiniLM-L6-v2')
model_st3 = SentenceTransformer('all-distilroberta-v1')

model_st4 = SentenceTransformer('paraphrase-MiniLM-L6-v2')
```

Figure 4. The models used for comparison

It's therefore a simple matter of translating our commit messages into sentence embeddings:

```
embeddings_use = embed(model_use, 'use', all_intents)
embeddings_use.shape
```

```
TensorShape([3075, 512])
```

```
embeddings_st1 = embed(model_st1, 'sentence transformer', all_intents)
embeddings_st1.shape
```

```
(3075, 768)
```

```
embeddings_st2 = embed(model_st2, 'sentence transformer', all_intents)
embeddings_st2.shape
```

```
(3075, 384)
```

```
embeddings_st3 = embed(model_st3, 'sentence transformer', all_intents)
embeddings_st3.shape
```

```
(3075, 768)
```

```
embeddings_st4 = embed(model_st4, 'sentence transformer', all_intents)
embeddings_st4.shape
```

```
(3075, 384)
```

Figure 5. Generate embeddings for the sentences using the chosen models

As seen above, most of the sentence embeddings have a high dimensionality (>500 features each). The Curse of Dimensionality is manifested by the fact that distance measures required for clustering, such as Euclidean and Manhattan, become meaningless at such high dimensions (for more information, check "On the Surprising Behavior of Distance Metrics in High Dimensional Space" by Aggarwal et al). While some of the sentence-transformer pre-trained models were designed to keep some distance measures useful, reducing dimensionality before clustering improves the results significantly.

The Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) technique, which was introduced in 2020, has quickly gained popularity as a dimensionality reduction technique. UMAP is much faster and more scalable than t-SNE, and it also

preserves the global structure of the data much better. As a result, it can be used for both visualization and as a pre-clustering dimensionality reduction step. I'll use it for both purposes in this case.

The Scikit-learn documentation gives a good overview of the various clustering algorithms that it supports and when each performs best. It is preferable for our current application to use an algorithm that does not require the number of clusters to be specified in advance and can also tolerate noisy data. Density-based algorithms are ideal because they do not require the number of clusters to be specified and are unconcerned about cluster shape. Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) has gained popularity because it has fewer and more intuitive hyperparameters and is resistant to variable-density clusters. A comparison of various clustering algorithms is included in the HDBSCAN documentation. HDBSCAN worked best for similar kind of problem, so we'll use it for the rest of the project.

There are two popular packages available for linking UMAP and HDBSCAN for topic modeling: Top2Vec and BERTopic. The default hyperparameters in both packages, however, do not perform well for problems with a small corpus, such as the current one (most of the data ends up being classified as noise). To make it easier to tailor to the

current problem of intent extraction, I'll instead use the UMAP and HDBSCAN packages for hyperparameter tuning.

The most important hyperparameters that control how UMAP performs dimensionality reduction are number of neighbors and number of components. The number of neighbors parameter determines how UMAP balances local and global data structures. As this parameter controls the size of the neighborhood UMAP looks to learn the manifold structure, lower values of number of neighbors will focus more on the very local structure. The number of components parameter controls the dimensionality of the final embedded data after dimensionality reduction on the input data. Unfortunately, there is no obvious way to select the best UMAP parameters in the absence of ground truth labels. I have manually created the labels, which would be used to compare how well the models perform at the end. The goal of here is to identify a methodology to use when dealing with unlabeled data. In his Top2Vec paper, Angelov mentions that n neighbors = 15 and n components = 5 worked best for his downstream tasks, but this is unlikely to be true for any generalized dataset.

HDBSCAN also has several critical hyperparameters to consider, the most important of which is the minimum cluster size. This intuitively controls the smallest grouping you want to consider as a cluster. Furthermore, the min samples parameter governs how conservative

the clustering is, which defaults to min cluster size if unspecified. The higher the value, the more points are rejected as noise/outliers. Points that would have been labeled as outliers are essentially kept by merging them with their most similar neighboring clusters by decoupling the two hyperparameters and having a smaller min sample than min cluster size. But here I'm trying to figure out how many clusters there are, this isn't what I want. As a result, I'll only think about changing the min cluster size parameter.

It is important to note that UMAP is a stochastic algorithm that uses randomness to accelerate approximation steps and optimize them. As a result, I will set the random seed state to a constant value in order to obtain consistent results for a given set of UMAP hyperparameters.

Creating a fitness function

The pipeline has three hyperparameters to tune (number of neighbors, number of components, and min cluster size). The next question is how to evaluate our clusters in order to select the best hyperparameters. Although it is commonly used with various clustering algorithms, Silhouette Score is not a good validation metric for density-based algorithms such as DBSCAN and HDBSCAN because it assumes all points are assigned a group and cannot handle noise/outliers appropriately. Some people have proposed and used Density Based Cluster Validation (DBCV) to tune HDBSCAN

hyperparameters. While it is likely to work well for a wide range of applications, it favored having fewer clusters and leaving too many samples in the "noise" category for this problem.

Instead, I'll use the useful probabilities_ HDBSCAN attribute, which is defined as "the degree to which each sample is a member of its assigned cluster." Points in the noise have a probability of zero; points in clusters have values assigned proportionally to the degree to which they persist as a member of the cluster.

The article by Nikolay Oskolkov provides a very intuitive and logical solution of simply defining our cost function that we want to minimize as:

Cost = percentage of dataset with less than 5% cluster label confidence

This will help ensure that as many data points as possible are assigned to actual clusters rather than being labeled as noise. But we can make every point a "cluster," or just make one big cluster? Therefore we'll have to be cautious when choosing the parameters for clustering.

While searching the hyperparameter space at random is effective, there is a better option: Bayesian optimization. In this case, I'll use the well-known hyperopt package to achieve this.

To begin, we must define the fitness function that we wish to minimize. The optimization constraints are incorporated into the objective function by including a penalty term if the number of clusters exceeds the desired range. The Tree-structured Parzen Estimator (TPE) algorithm is then used to minimize the objective function over the hyperparameter search space. Running the Bayesian search over our parameter space with 100 maximum evaluations yields slightly better results than random search.

```
hspace = {
    "n_neighbors": hp.choice('n_neighbors', range(3,16)),
    "n_components": hp.choice('n_components', range(3,16)),
    "min_cluster_size": hp.choice('min_cluster_size', range(2,16)),
    "min_samples": None,
    "random_state": 42
}

label_lower = 30
label_upper = 100
max_evals = 100
```

```
model_use.bayesian_search(space=hspace,
                          label_lower=label_lower,
                          label_upper=label_upper,
                          max_evals=max_evals)
```

```
100%|| 100/100 [31:42<00:00, 19.02s/trial, best loss: 0.1643089430894309]
best:
{'min_cluster_size': 13, 'min_samples': None, 'n_components': 8, 'n_neighbors': 9, 'random_state': 42}
label count: 4
```

Figure 6. Results of performing Bayesian search of hyperparameter space using UMAP+HDBSCAN.

It's therefore simple to run the pipeline with embeddings from the selected models:

```
model_use.best_params
```

```
{'min_cluster_size': 13,
 'min_samples': None,
 'n_components': 8,
 'n_neighbors': 9,
 'random_state': 42}
```

```
model_use.trials.best_trial
```

```
{'state': 2,
 'tid': 81,
 'spec': None,
 'result': {'loss': 0.1643089430894309, 'label_count': 4, 'status': 'ok'},
 'misc': {'tid': 81,
  'cmd': ('domain_attachment', 'FMinIter_Domain'),
  'workdir': None,
  'idxs': {'min_cluster_size': [81],
   'n_components': [81],
   'n_neighbors': [81]},
  'vals': {'min_cluster_size': [11], 'n_components': [5], 'n_neighbors': [6]}},
 'exp_key': None,
 'owner': None,
 'version': 0,
 'book_time': datetime.datetime(2022, 4, 22, 18, 1, 2, 344000),
 'refresh_time': datetime.datetime(2022, 4, 22, 18, 1, 22, 526000)}
```

```
model_st1.bayesian_search(space=hspace,
                          label_lower=label_lower,
                          label_upper=label_upper,
                          max_evals=max_evals)
```

```
100%|██████████| 100/100 [31:47<00:00, 19.07s/trial, best loss: 0.1808130081300813]
best:
{'min_cluster_size': 8, 'min_samples': None, 'n_components': 6, 'n_neighbors': 3, 'random_state': 42}
label count: 90
```

```
model_st2.bayesian_search(space=hspace,
                          label_lower=label_lower,
                          label_upper=label_upper,
                          max_evals=max_evals)
```

```
100%|██████████| 100/100 [26:57<00:00, 16.17s/trial, best loss: 0.20910569105691057]
best:
{'min_cluster_size': 6, 'min_samples': None, 'n_components': 11, 'n_neighbors': 12, 'random_state': 42}
label count: 89
```

In this case, I also have access to the ground truth labels, for evaluating how well the loss function correlates with performance. Manually examining how the models performed on some of the ground truth clusters gives us the following results:

Model	ARI	NMI	loss	label_count	n_neighbors	n_components	min_cluster_size	random_state
st1	0.171	0.568	0.180813008	90	3	6	8	42
st2	0.052	0.301	0.209105691	89	12	11	6	42
st3	0.145	0.179	0.179918699	4	8	7	15	42
use	0.112	0.149	0.164308943	4	9	8	13	42

Figure 7. Statistical analysis of the models

The Adjusted Rand Index (ARI) is frequently used in cluster validation since it is a measure of agreement between two partitions: one given by the clustering process and the other defined by external criteria.

Normalized mutual information (NMI) gives us the reduction in entropy of class labels when we are given the cluster labels. In a sense,

NMI tells us how much the uncertainty about class labels decreases when we know the cluster labels. It is like the information gain in decision trees.

Column1	label_st1	count	label	top_ground_category	top_cat_count	perc_top_cat
0	-1	549	add_id_security_group	add_security_group_policy	1001	182
1	59	190	add_rule_security_group	add_security_group_policy	233	123
2	74	189	add_id_security_bug	add_security_policy	191	101
3	32	182	adds_context_security_container	add_context_security_container	217	119
4	81	100	add_test_security_group	add_unit_test	100	100
5	26	69	nsx|p_id_security_group	add_security_policy	95	138
6	88	68	add_id_security_group	add_security_group_policy	1001	1472
7	58	64	add_group_security	add_security_group_policy	240	375
8	80	62	add_id_security_group	add_security_group_policy	1001	1615
9	79	60	add_group_security	add_security_group_policy	240	400
10	56	49	adds_id_security_bug	add_security_policy	49	100
11	35	49	add_group_security	add_security_group_policy	240	490
12	39	48	add_id_security_keystone	add_security_policy	54	112
13	22	39	use_id_security_runtime	add_security_policy	39	100
14	84	39	changed_group_security	add_security_group_policy	41	105
15	19	37	add_quota_security_group	add_security_group_policy	39	105
16	40	37	add_bandit_security	add_bandit_security	37	100
17	48	35	dnm_job_openstack	testing	35	100
18	69	34	adds_group_security	add_security_group_policy	45	132
19	82	34	add_group_security	add_security_group_policy	240	706

Figure 8. Comparison of the labels generated by the model with my labels

Column1	text	label_use	label_st1	label_st2	label_st3	label_st4
0	Enable etcd with security setting Add etcd client/server certificate generation process in ansible play	2	29	-1	1	5
1	Add new Debian security mirror suite pattern Starting with Debian 11 (bullseye)_ security packages a	2	55	34	2	1
2	Fix rule replacement in security policy Change-Id: I65918a003833dea45897d360ab7c3ce60616ec4c	2	71	41	2	5
3	OvS: Add pod security context to pods This PS adds the pod security context snippet to pods Change	1	32	11	1	5
4	Horizon: update east-west network security policy This patch set updates the ingress network securi	2	59	29	2	5
5	Implement helm-toolkit snippet to Keystone pods/containers This updates the Keystone chart to inc	1	32	11	1	5
6	Add missing security context to Mini-mirror test pods/containers This updates the mini-mirror chart	1	3	11	1	5
7	Enable setting default rules for default security group In the current implementation_ the default se	2	58	43	2	5
8	Testingrst misses a step for adding security rule Change-Id: Id748be47a85abd4da75e63964edfbd07	2	81	-1	2	5
9	Test security groups Install ovs kernel module from source and turn on the sec group tests Change-I	2	35	-1	2	5
10	Add OSSN-0040 security groups don't work with vip and ovs plugin Closes-Bug: #1163569 Change-Id	2	35	68	2	5
11	Add security groups dscp spec This adds the spec "Add a 'dscp' field to security group rules to screen	2	64	-1	2	5
12	Open vSwitch-based Security Groups: OVS FirewallDriver Specification for implementing Open vSwit	2	35	68	2	5
13	Add security groups for nova As current implementation of puppet-nova can't realize nova security g	2	58	72	2	5
14	Add security to libvirtd Allow libvirt to be accessed only from management network Change-Id: Ie1a	2	13	8	-1	3
15	Add security to libvirtd Allow libvirt to be accessed only from management network Change-Id: Ie1a	2	13	8	-1	3
16	Add security to libvirtd Allow libvirt to be accessed only from management network Change-Id: Ie1a	2	13	8	-1	3
17	Use fanout RPC message to nofity the security group's change when a security group members or ru	2	-1	-1	2	5
18	Use security group id for launch instance When there are two security groups with the same name_	2	83	81	2	5
19	Wrong Link to External Site Integrity life-cycle in Security Guide A link to dm-verity under the section	2	74	42	2	5
20	Add VRRP protocol to security group API Change-Id: I79381b5fcd506a6f7b6cba7d7b3c2f0038e86c0	2	82	87	2	5
21	Add security testing for transport layer check The test cases check all resources related URLS in serv	2	74	42	2	5
22	Fix API misimplementation for security groups According to the Compute API reference [1]_ the 'des	2	69	-1	2	5
23	If a Neutron security group rule protocol is ICMP_ port_range_max value must be an ICMP code Cha	2	59	70	2	5
24	Add security testing for authorization and update client These test cases cover authorization checks	2	22	64	2	5
25	Adds security tests to automate checking of common vulnerabilities This is an initial set of security t	2	81	64	2	5
26	Remove namespace argument from security group initialization Neutron security groups (iptables_ i	2	72	61	2	5
27	networking-tricircle core plugin and security group Initial implementation include option to use ML2	2	-1	65	2	5
28	Discuss bridge/netfilter & OVS Hybrid driver with security groups With the dependent change the OV	2	35	68	2	5
29	Add ability to change VIP security group * Convert Edit VIP form to a workflow This is consistent wit	2	59	-1	2	5
30	Add separate mirror for security ubuntu updates Unfreeze only security updates for FUEL release Ch	2	55	34	2	5
31	networking-tricircle core plugin and security group Initial implementation include option to use ML2	2	-1	65	2	5

Figure 9. Comparison of the labelling done by the models

For problem statement p1, I described a framework for using domain knowledge to create a constrained optimization problem that can automatically tune UMAP and HDBSCAN hyperparameters. This enables us to easily group code commits and assign descriptive labels. Before deciding or needing to complete time-intensive manual labeling, the clustering results provide useful insights into unlabeled text data in a very short amount of time. This finally becomes a matter of grouping the clusters based on the keywords that we are looking for.

For problem statement p2 I'll focus on the Regression Objective Function. The computation of cosine similarity between the two sentence embeddings u and v is described in the fig below. The mean squared-error loss is used as the fitness function.

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

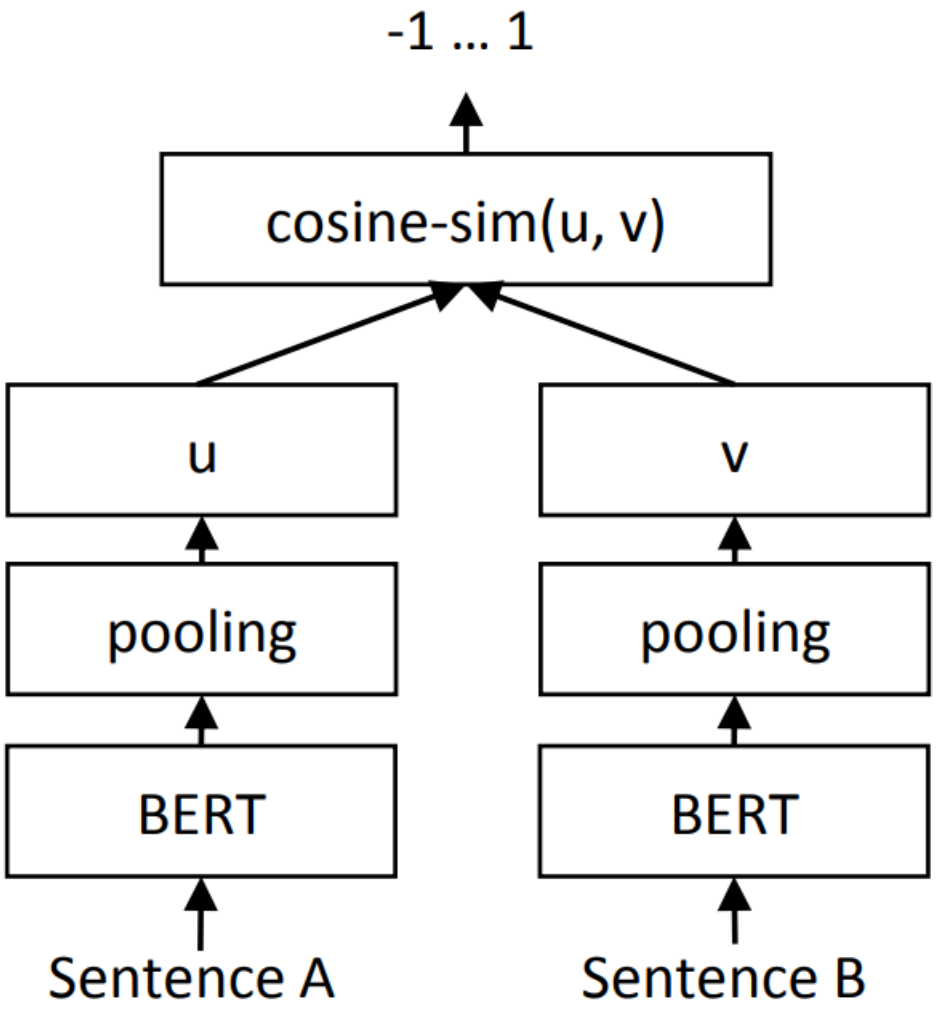


Figure 10. Cosine similarity using SBERT

The model gives us the following output for the test sentence `"updated the security mirror for the Debian repositories"`:

	Sentences	Similarity
2531	Temporary overwrite of the infra mirror for debian security repo Change-Id: I7c41f560251e16256a5d109fddedee189eb12664 (	0.76561737
2538	Temporary overwrite of the infra mirror for debian security repo Change-Id: I23163676420498bb3fd3d0e411554ca34dfbe318	0.75169861
2537	Temporary overwrite of the infra mirror for debian security repo Depends-On: https://reviewopendevorg/686825 Change-Id: I2	0.75160265
2532	Temporary overwrite of the infra mirror for debian security repo Change-Id: I7c41f560251e16256a5d109fddedee189eb12664	0.75013119
220	Add security mirror override for debian-minimal Set the security mirror URL independently in the debian-minimal element_ sinc	0.66724682
1844	Add new Debian security mirror suite pattern Starting with Debian 11 (bullseye)_ security packages are in bullseye-security as o	0.66075248
2549	Add security mirror override for debian-minimal Add option to set the security mirror URL independently in the debian-minima	0.65629131
1979	Remove workaround for missing debian stretch security repo Debian stretch is no longer tested on OSA master branch and this	0.65133035
2548	Override security mirror for Debian image builds Diskimage-builder defaults to a separate mirror value for the Debian security a	0.63167346
1633	Fix paths to updates and security repositories According to a new scheme of repos Change-Id: I1b78a50603cfbb0d2f7b4aba455	0.62580919
1709	Fix paths to updates and security repositories According to a new scheme of repos Change-Id: I1b78a50603cfbb0d2f7b4aba455	0.62580919
2523	Add security suite name override in debian-minimal Add option to set the suite subpath after the release name for the security	0.60700434
1778	Add security repository into default set of repos Change-Id: I19b75bf9a1e6a3af6a2215f23ad96eaa4a0665cd Closes-bug: #1466	0.60073054
1779	Add security repository into default set of repos Change-Id: I19b75bf9a1e6a3af6a2215f23ad96eaa4a0665cd Closes-bug: #1466	0.60073054
1295	Fix invalid security repo Change-Id: I0365b7430ff17348fa2e0a4b8a3476b149a7ccca	0.59910089
829	Add debian security gpg key We have debian-security defined for mirroring in [1] with key C857C906; however we don't seem t	0.59837085
1203	Fix invalid security repo Change-Id: I0365b7430ff17348fa2e0a4b8a3476b149a7ccca Closes-Bug: #1618397 (cherry picked from	0.59757996
1	Add new Debian security mirror suite pattern Starting with Debian 11 (bullseye)_ security packages are in bullseye-security as o	0.58133215
1716	Fix paths to updates and security repositories According to a new scheme of repos Closes-Bug: #1482710 Change-Id: Ibf16be5c	0.57987952
1687	update the release tags for security deliverables Change-Id: I34d0103d36635974df8a83899b255d23861b2a2e	0.57207799
1199	Fix variable name for cache debian security packages Change-Id: I925b389634c9f52dcc2ff61bfa605863a12a507d	0.55789101

Figure 11. Similarity of the given statement with other sentences

As we can observe from the fig above, the commit number 2531 in the given dataset is most similar to the sentence I supplied to the model. We can run the description of the incoming commits to see if they are like other commits, and the type of check-in can be automatically evaluated.

You can find all the code samples and data here.

References:

1. https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-concept-intent
2. https://cloud.google.com/dialogflow/es/docs/intents-overview
3. https://github.com/pandas-profiling/pandas-profiling
4. https://www.acsac.org/2000/papers/78.pdf

5. https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
6. http://dl.acm.org/citation.cfm?id=2002168.2002170
7. https://dl.acm.org/doi/10.1145/360248.360252
8. https://www.usenix.org/events/osdi08/tech/full_papers/cadar/cadar.pdf
9. https://dl.acm.org/doi/10.1145/2566486.2568024
10. http://arxiv.org/abs/1701.07179
11. http://dl.acm.org/citation.cfm?id=2486788.2486873
12. https://doi.org/10.1109/HSI.2012.22
13. https://en.wikipedia.org/wiki/Topic_model
14. https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
15. http://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf
16. https://radimrehurek.com/gensim/index.html
17. https://medium.com/pew-research-center-decoded/making-sense-of-topic-models-953a5e42854e
18. https://tfhub.dev/google/universal-sentence-encoder/4
19. https://arxiv.org/abs/1803.11175
20. https://arxiv.org/pdf/1908.10084.pdf
21. https://huggingface.co/
22. https://bib.dbvis.de/uploadedFiles/155.pdf
23. https://arxiv.org/pdf/1802.03426.pdf

24. https://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods
25. https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html#comparing-python-clustering-algorithms
26. https://arxiv.org/pdf/2008.09470.pdf
27. https://github.com/MaartenGr/BERTopic
28. https://towardsdatascience.com/how-to-cluster-in-high-dimensions-4ef693bacc6
29. https://medium.com/district-data-labs/parameter-tuning-with-hyperopt-faa86acdfdce
30. https://www.youtube.com/watch?v=4I3gS1cmqe4