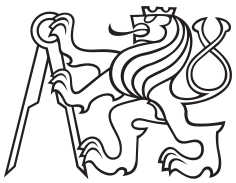
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Frequent Itemset Mining and Multi-Label Classification of User Responses on Well-Being

Lucie Borovičková

Supervisor: Ing. Marek Otáhal
Field of study: Open Informatics
Subfield: Artificial Intelligence and Computer Science
May 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Borovičková Lucie** Personal ID number: **483685**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Frequent Itemset Mining and Multi-Label Classification of User Responses on Well-Being

Bachelor's thesis title in Czech:

Hledání častých podmnožin a multi-label klasifikace uživatelských odpovědí na téma wellbeingu

Guidelines:

1. Motivation & Goal: We aim to search for frequent itemsets of topics detected by a multi-label classifier of answers to user-satisfaction surveys. These itemsets are expected to identify the most important problems of the respondents.
2. Data: We have a dataset collected over a period of 2 years in 18 companies, accounting approx. 1600 entries - each entry is a part of CZ/EN text. These answers are extracted from the following sources (D1-4): D1-likert answers; D2-answers to open-ended questions; D3-"Partial"/contextual answers; and D4-"spontaneous feedback": the users actively reach to us with feedback (so we don't have any context to frame it). The data is manually annotated into a set of classes (eg. "quality of air", "ergonomics", "burnout", etc.) given by a domain-expert (wellbeing) and will be used as ground-truth for training a classifier. The student is granted full access to the data but the dataset remains an intellectual property of a company that collected the data and cannot be published as a part of the thesis.
3. Objectives for the student:
a. Preprocessing: convert the text data from all the sources (D1-4) into a suitable format for the following NLP.
b. Implement a multi-label classifier using a RandomForestClassifier. Train and evaluate (cross-validation, appropriate metric macroF1, Kappa, etc.) on annotated dataset(3a/).
Optionally, try another advanced model (likely based on neural networks) and choose the better classifier.
c. Find frequent itemsets in the dataset now represented as a set of sets of classes (labels detected in 3b/).
i. Compare frequent itemsets found on the annotated data (ground-truth), and on (extended) data labeled by the developed classifier.
ii. Compare itemsets found on the whole dataset vs. computed on data for each company separately.
iii. Can you find interpretation of some of the subsets tracing back to real-life problems?
iv. Optionally, discuss if the found itemsets suggest a possible modification to the classes defined for the classifier.
4. Review relevant literature for points 3a-c/
5. Propose a method and implement a working solution in Python.
6. Outcomes of objectives in 3/ would be evaluated as follows:
a. Data in a common format suitable for NLP, preprocessed to simple sentences in English with self-contained meaning. Discuss why this was needed.
b. The student designed and trained a classifier, and evaluated results
c. Discuss found frequent itemsets and their relevance.

Bibliography / sources:

[1] Nitin Indurkhya, Fred J. Damerau, Handbook of Natural Language Processing, published by Chapman and Hall/CRC, 2010
[2] Goodfellow, I., Bengio, Y. and Courville, A.: Deep Learning, MIT Press, 2016. www
[3] Christian Borgelt, Frequent item set mining, WIREs Data Mining Knowl Discov, 2012
[4] Grigorios Tsoumakas and Ioannis Katakis, Multi-Label Classification: An Overview, International Journal of Data Warehousing and Mining, 2009
[5] J. Lažanský, V. Mařík, O. Štěpánková, Umělá inteligence (1-6), Academia, 2000

Name and workplace of bachelor's thesis supervisor:

Ing. Marek Otáhal, Robotic Perception, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **12.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. Marek Otáhal Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to express my gratitude to my supervisor Ing. Marek Otáhel for his patience with me and his guidance through the topic of this thesis. I would also like to thank my friends and family for their unlimited support during my studies at FEE CTU.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 21. May 2021

Abstract

In the thesis, we developed a method to automatically generate insights into employees' satisfaction with their workplace and perception of their wellbeing based on textual answers to questionnaires. To be able to process a large body of data, we developed a pipeline to preprocess and aggregate textual data from a number of different sources. Further, we trained an NLP classifier (RandomForest) to detect labels in the user responses and verified the classifier performs sufficiently for the task. On these sets of detected labels, we run frequent itemset analysis and subject the results to significant support and (in)dependence tests. On the practical side, the method is open-sourced and available to support decisions at Human Resources (HR) departments based on data-driven feedback. We evaluated the method on real-world data from various companies located in the Czech Republic and showed that we are able to find significant frequent itemsets, and design a workflow to interpret the results and trace them back to actual problems.

Keywords: frequent itemset, NLP, multi-label classification of text, wellbeing insights, chi-squared test, data mining, feedback

Supervisor: Ing. Marek Otáhal

Abstrakt

Tato práce se zabývá automatickou tvorbou vhledů do spokojenosti zaměstnanců s jejich pracovním prostředí a wellbeingem na základě textových odpovědí z dotazníků. Pro zpracování velkého korpusu dat byl vyvinut postup, který zpracoval a seskupil texty z různých zdrojů. Dále byl natrénován klasikátor, který zpracováním přirozeného jazyka odhaluje témata v uživatelských odpovědích a bylo ověřeno, že funguje dostatečně dobře pro naše zadání. Na souborech nalezených témat byly nalezeny časté podmnožiny, tyto výsledky byly podrobeny chí-kvadrát testu pro zjištění (ne)závislosti jednotlivých témat. Z praktické stránky, tato metoda má open-source licence a je HR oddělením k dispozici pro podporu jejich rozhodnutí na základě dat ze zpětné vazby. Metoda byla otestována na reálných datech různých společností se sídlem v ČR a bylo ukázáno, že je pomocí ní možné najít (statisticky významné) časté podmnožiny a navrhnout postup k interpretaci dat a jejich zpětnému vysledování ke skutečným problémům.

Klíčová slova: časté podmnožiny, zpracování přirozeného jazyka, multi-label klasifikace textu, vhled do wellbeingu, chí-kvadrát test, dolování dat, zpětná vazba

Překlad názvu: Hledání častých podmnožin a multi-label klasifikace uživatelských odpovědí na téma wellbeingu

Contents

Figures

Tables

Chapter 1

Introduction

Nowadays, every company, every city, every person generates lots of data. Therefore the challenge is usually not obtaining the data, but its preprocessing for statistical or machine learning methods and interpreting the results. We aim to provide methods for processing textual data from questionnaires regarding employees' wellbeing and offer insights into the results.

Human Resources, and specifically Wellbeing Management (sometimes called Employees Engagement), is a field without much hard data. Decision making often relies on subjective perception and judgement calls of an HR manager. The management might not always fully appreciate this, but a systematic approach to employees' wellbeing is crucial to a company's performance. "Being satisfied" is surely beneficial for the employees but the companies profit from it as well as a link between wellbeing and productivity is assumed. A study conducted by Fisher [Fis03] showed that almost 92% of Australians believed that "A happy worker is likely to be a productive worker." Moreover, talented employees are always hard to find and good reputation of the employer's brand and their workplace wellbeing helps to attract them.

While it is possible to know about 50 people at work by heart, including their wishes and obstacles to productive work, there is a limit to an HR representative's mental capacity to be familiar with everyone on their own. As the company grows substantially (more than 300 full time employees) HR finds itself in the need of some kind of quantitative analysis. Be it external consulting or internal surveys concerning employee satisfaction. Our goal in this thesis is to help HR managers or whoever is tasked with managing employees' wellbeing (meaning satisfaction and productivity) with gaining

insight into such survey's collected data. We aim to do so by taking on the data from questionnaires from various companies as an input and suggest effective automated methods which successfully identify the challenges a given company faces. Crucial factors are the ability to understand the employees' answers in the form of natural language and secondly the ability to distinguish commonly shared opinions in order to formulate representative conclusions. Such obtained insights should serve the management as the grounds for data-driven decision making on improvements of employees' wellbeing and company's workplace. With successful implementation of this thesis, we'll provide a set of tools, a pipeline capable of automatically processing feedback from employees, and a method of generating insights into the most relevant issues for the tested workplace, and back it up with evidence.

Part I

Theoretical Part

Chapter 2

Classifiers in Natural Language Processing (NLP)

In this chapter we will focus on classification tasks in NLP, namely labelling textual data. The outline of the chapter is as follows: Multi-label classification and approaches used for solving, decision trees, Random Forest classifier - used in our solution, and introduction of metrics often used for evaluating classifiers. When figuring out how to process data in a form of natural language one has two options. First one is using a "black box" model such as a (deep) neural network (NN). The input is in a form of raw data which the model pulls through and outputs the result. Second one is using a model such as a decision tree following logical rules. Although neural networks can reach a lot higher accuracy, the latter types of models are often employed (at the expense of accuracy) for their comprehensibility and interpretability, according to [WDH+20]. The top down approach for tackling the data can be studied and separate decisions can be challenged instead of receiving an output without proper justification or trying to interpret the numeric weights of the connections between the nodes. [Kot13]

2.1 Multi-Label Classification

The following part is dedicated to providing an overview of Multi-Label Classification problems. As it is the approach that was used in the practical part to solve the given task. It is frequently used for organizing text documents such as agreements, e-mails, invoices, books, magazines, blog posts etc.

[HCRdJ16]

Note the difference between Multi-Label and Multi-Class Classification problems. In both of them we look at multiple categories. If each instance is assigned only one of those categories i.e. one label we talk about Multi-Class Classification whereas assigning several not mutually exclusive categories indicates we are dealing with the Multi-Label Classification. A nice way to keep this difference in mind is via a pizza similarity. Let one set of labels be the toppings - tomato sauce, mozzarella, basil, cheddar cheese, blue cheese, salami, ham, etc. and the other one the types of pizza - Margherita, Marinara, Capricciosa, Quattro Formaggi, etc. On each round of dough we can put multiple toppings (let's say tomato sauce, mozzarella, basil), this is (multi) labeling. A combination of topics on a dough can be classified only as one type of pizza (Margherita for previous example), this is classification.

2.1.1 Approaches for Solving Multi-Label Classification Problems

There are two, respectively three approaches to solving multi-label classification problems. We can either modify the dataset and use existing algorithms or modify the algorithms. Third technique emerges from the first one - using an ensemble.

Problem/Data Transformation

The original dataset is transformed in such a way that it is possible to treat the problem as a single-label problem. This allows the use of traditional classifiers.

Binary relevance is the most straightforward technique which trains and evaluates a different classifier for each label. This method does not consider label correlations as each of the classifiers is handled independently.[ZLLG18]

Classifier Chains likewise transform an n-label problem to n single-label classifiers. The difference is that the first classifier is trained only on the features and every next classifier is trained on features combined with the information about previously assigned labels. Thus, forming a so-called chain and considering label correlations.[RPHF21]

Label Powerset creates from each unique label combination a different class and so the task now presents a multi-class problem. It is obvious that this method also acknowledges the label correlations.

Adapted Algorithm

Rather than transforming the dataset and creating smaller problems it is possible to modify existing algorithms to generate multi-label outputs. Support Vector Machines (SVM) is an example of an algorithm developed for binary classification, but which can be modified to work also with multi-label [Bur98]. A k-Nearest Neighbours (kNN) can easily predict multi-label without sophisticated adjustments. We just check whether the nearest neighbors accommodate more than just one class.

Ensemble Approach

Combining several binary and/or multi-class classifiers whose outputs can be weighted to compensate for their shortcomings and biases is efficient and commonly used method. It is also possible to tackle the imbalance datasets with this approach.[GFB+11]

2.2 Decision Trees

An algorithm that recursively divides features into separate classes based on maximal gain in each split and thus forming a decision tree. The recursion is repeated until all nodes contain only instances of the same class. More about implementation algorithms and basic issues can be found in [Kot13]

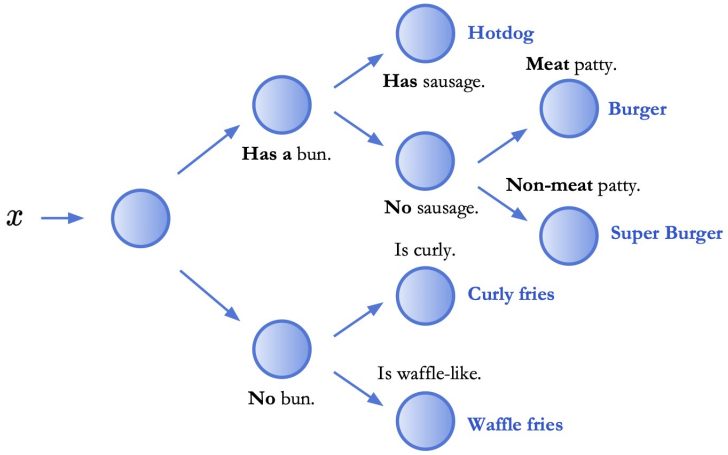


Figure 2.1: Decision tree shown as a graph. Source: [WDH+20]

2.2.1 Random Forest Classifier

Random Forest (RF) reveals quite a lot just by its name. It is a supervised learning algorithm that consists of a series of decision trees which operate as an ensemble.[Pal05]

Random Forest Classifier functions by taking into account a lot of predictions from uncorrelated trees and assigning the sample the label with the most votes. It is vital that the trees are uncorrelated, so that they compensate for each other's errors as the decision trees are known for overfitting training data. This drawback is partly tackled by taking the average of all the predictions and partly by a method called Bagging (or Bootstrap Aggregation). Bagging allows individual trees to randomly replace some samples from the drawn set. Classic decision trees consider all the features and then split based on the one that creates the biggest difference in the observations in the nodes while bagging in RF results in trees deciding based only on a random subset of the features - the outcome of this is more variety among the trees.

2.2.2 Neural-Backed Decision Trees (NBDT)

[WDH+20] recently combined in their work the interpretability of decision trees with the accuracy of neural networks and called these models Neural-Backed Decision Trees. In NBDT predictions are made by a decision tree,

but each node contains a neural network making the low-level decision such as *Is curly* or *Is waffle* like.

2.3 Model Evaluation

In this section we will describe ways of training and measuring performance of a classifier. We will discuss suitability for different tasks.

2.3.1 Cross Validation

Cross Validation (CV) is a method commonly used to reduce bias in the model by ensuring that every sample can serve in the training and testing set. It splits the dataset into k folds (usually 5 or 10), uses k-1 folds for training and the last one for testing. This process is repeated until all of the parts were used for testing. It then averages the obtained scores and that is the performance of the model.

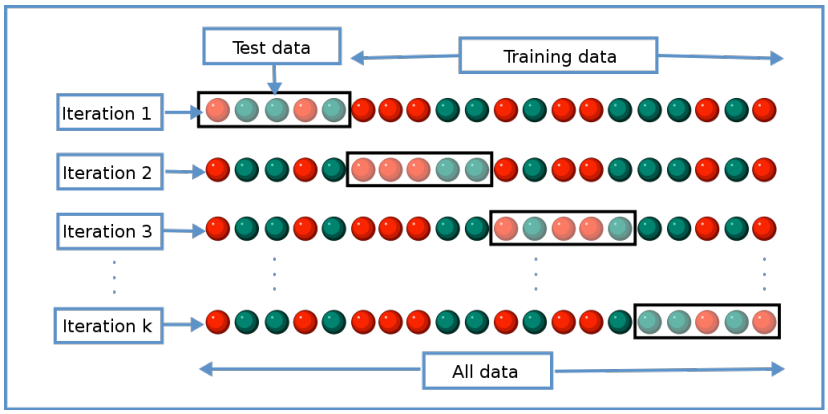


Figure 2.2: Illustration of k-fold cross validation. Source: Wikimedia Commons

2.3.2 Confusion Matrix

A confusion matrix is basically a table that visualizes the performance of a model. The rows represent the number of instances in the predicted class and columns represent the number of instances in an actual class (or vice versa).

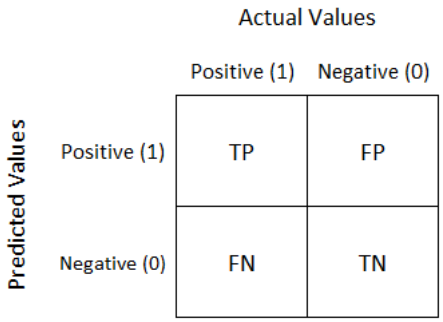


Figure 2.3: Illustration of confusion matrix and its 4 boxes. Source: Understanding Confusion Matrix by Sarang Narkhede on Towards Data Science

Each of the four boxes has its name. True Positive (TP) - model predicted 1 and the actual class is 1. False Positive (FP) - model predicted 1, but the actual class is 0. False Negative (FN) - model predicted 0, but the actual class is 1. True Negative (TN) - model predicted 0 and the actual class is 0.

2.3.3 Metrics

A metric is a quantifiable measure that is used to track and assess the status of a specific process. Our goal is to choose a suitable model and find out its classification capabilities. For this purpose, we will use different metrics. The hard part about using different metrics for model evaluation is setting expectations on what part of the problem we are focusing on. Do we want to minimize false positives even at the cost of decreasing true positives? Do we want to maximize true positives? Then we will likely encounter more false positives.

Below we will list and briefly describe several common and frequently used metrics. Followed by a discussion regarding their pitfalls and suitability for different datasets and tasks.

Classification Accuracy

Classification Accuracy is often shortened to accuracy. It is the number of correct predictions (True Positives and True Negatives) to the total number

of items (which we make predictions about).

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made} \tag{2.1}$$

Its usefulness is highly dependent on the number of items from each class. If the ratio is balanced we can view accuracy as a reliable metric for evaluation, but if class A has an apriori probability of 95% and B only 5%, then we can classify all the items as A and get 95% accuracy. A perfectly balanced dataset (P(A) = 0.5, P(B) = 0.5) with the same classifier gives us only 50% accuracy which is nothing more than a coin flip.

Balanced Accuracy

Balanced accuracy was designed for dealing with unbalanced datasets. [BOSB10]

$$Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2} \tag{2.2}$$

where $Sensitivity$ (also known as $True\ Positive\ rate$ or $Recall$) is computed as

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.3}$$

and $Specificity$ (also known as $True\ Negative\ rate$) is computed as

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} \tag{2.4}$$

Getting back to the example from above and following the same classification, balanced accuracy would be 0.5.

Logarithmic Loss

Logarithmic Loss or Log Loss penalizes the wrong classification.

$$Logarithmic\ Loss = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} * log(p_{ij}) \tag{2.5}$$

where y_{ij} indicates whether sample i belongs to class j or not
and p_{ij} indicates the probability of sample i belonging to class j

Log Loss has no upper bound and it exists on the range $\langle 0, \inf)$ The smaller the better. [Yil]

Note that when there are not just two classes 1/0 or in our case - the text contains or does not contain the topic - the metric is usually called logarithmic loss. When there are more than two classes (whose predictions can be either probabilities or discrete) the metric is called cross entropy loss or simply cross entropy. The underlying math is the same - it is always the negative sum of the products of the logs of the predicted probabilities times the actual probabilities - the only difference is the terminology used. [McC]

Logarithmic loss is often the number one choice of evaluation in Kaggle competitions and justifiably. Other classifiers evaluate based on the final decisions (0/1). However, log loss takes into account the predicted probabilities for each class and adjusts the evaluation by "how wrong the wrong prediction was".

A simple example below illustrates how logarithmic loss functions. We are given a dataset with $y_{true} = [1, 0, 0, 1, 1]$. Consider two predictions: $y_{pred_1} = [1, 0, 0, 0, 1]$ and $y_{pred2} = [1, 0, 0, 0, 1]$ according to the accuracy score they are performing equally, but log loss goes deeper than this. For its evaluation it requires the probabilities of each class. So we give the metric just that $y_{pred1} = [[0.2, 0.8], [0.6, 0.4], [0.7, 0.3], [0.65, 0.35], [0.25, 0.75]]$ and $y_{pred2} = [[0.2, 0.8], [0.6, 0.4], [0.7, 0.3], [0.85, 0.15], [0.25, 0.75]]$, they only differ in the fourth prediction. The log loss for the former predictions is 0.4856 and for the latter 0.6550. Log loss reflects that the first model was closer to getting the right prediction.

Log loss vs. Mean Squared Error

One of the considered metrics was Mean Squared Error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2 \tag{2.6}$$

But it is a lot more suitable for regression instead of classification. Compared to Log loss, MSE also penalizes the wrong classification, but the loss (penalization for misclassification) is not as high as we would like it to be. If we were to use neural networks to obtain the labels, the backpropagation would not show the wrong classification to be as bad as they are and the network would not learn as efficiently. [All71]

Macro F1 Score

F1-Score balances between precision and recall. There are actually two formulas for calculating Macro F1 score - 'averaged F1' and 'F1 of averages' [OB19]. Averaged F1 is calculated as an arithmetic mean over harmonic means meaning F1 scores are computed for each class and then averaged via arithmetic mean.

$$F_1 = \frac{1}{N}\sum_x F1_x = \frac{1}{N}\sum_x \frac{2P_xR_x}{P_x + R_x} \tag{2.7}$$

F1 of averages is the opposite - harmonic mean over arithmetic means. The harmonic mean is computed over the arithmetic means of precision and recall.

$$F_2 = H(\bar{P}, \bar{R}) = \frac{2\bar{P}\bar{R}}{\bar{P} + \bar{R}} = 2\frac{(\frac{1}{N}\sum_x P_x)(\frac{1}{N}\sum_x R_x)}{\frac{1}{N}\sum_x P_x + \frac{1}{N}\sum_x R_x} \tag{2.8}$$

The scale of Macro F1 is $\langle 0, 1\rangle$, 1 being the best value. In this thesis equation 2.7 is used, as 2.8 is likely to provide misleadingly high scores with imbalanced datasets.

Macro F1 vs F1 weighted

Similarly to Macro F1 Score, F1 Score Weighted calculates metrics for each label, but instead of using arithmetic means to find their average it weights the average by the number of true instances for each label. This alters Macro F1 to account for label imbalance.

Macro F1 treats all classes (in our case just two) equally and is insensitive to imbalanced datasets and therefore it will be low for models that do well on the common classes while performing badly on the rare classes. Weighted Macro F1 does just the opposite and when putting together the scores it considers the imbalance. [GM05]

Cohen's Kappa Coefficient

κ (kappa) is often used to measure the level of agreement between two raters. One of them can be a classification model and so kappa assesses the model.

Its value can be obtained by the following equation:

$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{2.9}$$

where

$$p_o = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.10}$$

is overall accuracy and p_e is the hypothetical probability of chance agreement between the model predictions and the actual class values, using the observed data to calculate the probabilities of each observer randomly seeing each category.

$$p_e = p_{e1} + p_{e2} = p_{e1,target} * p_{e1,pred} + p_{e2,target} * p_{e2,pred} \tag{2.11}$$

which can be also written as

$$p_e = \frac{TP + FP}{all} \cdot \frac{TP + FN}{all} + \frac{TN + FN}{all} \cdot \frac{FP + TN}{all} \tag{2.12}$$

where $all = TP + FP + FN + TN$ and p_{e1} is the probability of the predictions agreeing with actual values of class 1 by chance.

The value of Cohen's kappa theoretically varies on a scale from -1 to 1, 1 being the perfect agreement and 0 indicating that the agreement is as good as a random guess. Negative values meaning that the overall accuracy is even worse than a random guess. [Bla08]

Not equally achievable perfect score

Achieving a kappa value equal to 1 would not only mean that there is a complete agreement between the two raters, but also that the labels are perfectly distributed - 50% zeroes and 50% ones. The maximum reachable Cohen's kappa value lowers with the difference between the distributions of the predicted and actual target classes. Obtaining the maximum value means correctly predicting all samples in either class, i.e. the number of false negatives or false positives in the confusion matrix being zero. [Bla08] It can be computed as:

$$\kappa_{max} = \frac{p_{max} - p_e}{1 - p_e} \tag{2.13}$$

where

$$p_{max} = min(p_{target=1}, p_{predicted=1}) + min(p_{target=0}, p_{predicted=0}) \qquad (2.14)$$

Chapter 3

Frequent Itemsets

3.1 Basic Notions

Frequent itemset mining is one of well known and used data mining techniques. The task of finding frequent patterns has grown in popularity likely due to an article by Agrawal et al. [AIS93] published in 1993. They introduced the task in association with a problem known as basket case analysis - finding out which products customers frequently buy together. They followed up in 1994 with an article introducing fast algorithms for solving the problem [AS$^+$94]. According to Google Scholar, those two articles have been cited over 50 000 times. Nonetheless the cornerstone of what's nowadays called "association rules" was proposed as early as in 1966 in an article by Petr Hájek et al. [HHC66]. The first applications of the presented GUHA method were mainly in the field of physiology.

The knowledge obtained can be used for example to increase sales by putting the items frequently bought together next to each other on the shelf, creating a bargain, marketing the items in a campaign or putting a section "people who bought this also bought that" section on your e-shop.

It can be formally defined as: a set $B = i_1, ..., i_n$ of items, called the item base, and a database $T = (t_1, ..., t_m)$ of transactions. The items may represent for example products offered by a shop or in our case the topics people mention when giving information about the biggest problems in their workplace. The transactions represent sets of items people bought together or

topics an individual employee mentions together. The itembase can be given explicitly, but it is usually done implicitly as the union of all transactions, that is, $B = \cup k \in 1, ..., mt_k$.

3.1.1 The Support of an Itemset

The support of an itemset represents the number of transactions it is contained in, mathematically speaking: Consider the cover $K_T(I) = \{k \in \{1, ..., m\} | I \subseteq t_k\}$ of an item set I. The support $s_T(I)$ is then $s_T(I) = |K_T(I)|$. An itemset is called frequent iff $s_T(I) \geq s_{min}$, where $s_{min} \in \{N\}$ is given by the user. It is also possible to define a frequent itemset based on its relative frequency in the database T as $\sigma_T(I) = s_T(I)/m$ which is compared against a user given $\sigma_{min} \in (0, 1)$

3.1.2 Search Space

Generating all itemsets in the power set 2^B, computing support, and filtering out the non-frequent itemsets can be quite challenging, especially with a larger itembase. Luckily there are some facts that allow us to better manage the computational complexity.

The item set support is *antimonotone*. Meaning that adding an item to an itemset cannot increase its support. Mathematically said: $\forall I \subseteq J \subseteq B : s_T(I) \geq s_T(J)$ This property is the foundation for the *Apriori property*: no superset of an infrequent itemset can be frequent $\forall I \subseteq J \subseteq B : s_T(I) < s_{min} \Rightarrow s_T(J) < s_{min}$

Thanks to this the subset relationships between itemsets form a partial order on 2^B, which can be represented by a Hasse diagram. [Bor12]

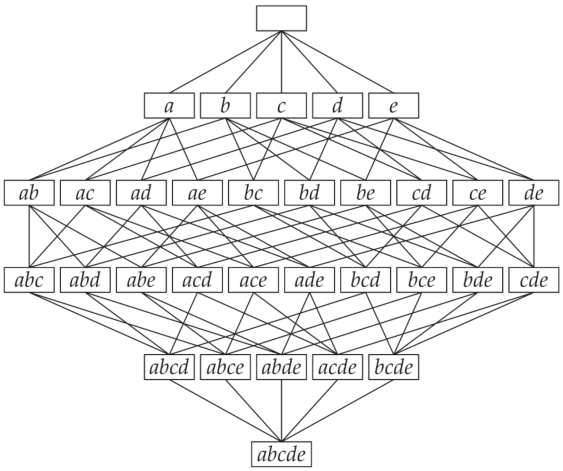


Figure 3.1: Hasse diagram for the partial order induced by $\subseteq$ on $2^{\{a,b,c,d,e\}}$

When searching through the space from the top down some sets are created duplicity by adding the items in a different order. This can be eliminated by transforming the Hasse diagram to a tree by assigning a unique parent to each itemset, see Fig. 3.2. As the sibling sets only differ in the last item it is common practice to merge them into one node as shown in Fig. 3.3. [Bor12]

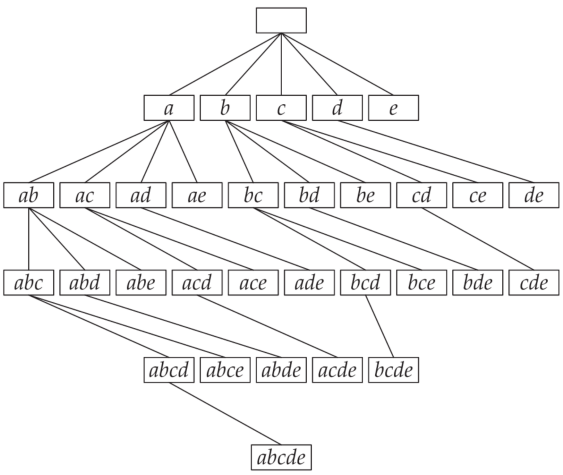


Figure 3.2: Tree that results from assigning a unique parent to each itemset.

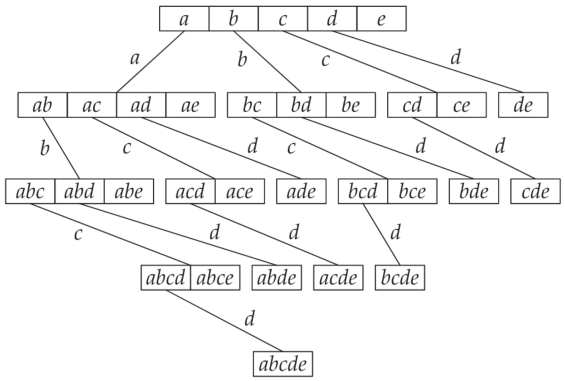


Figure 3.3: A prefix tree in which the sibling nodes with the same prefix are merged.

3.2 Statistically Sound Pattern Discovery

Exploring large search spaces and looking for itemsets that fulfill user-given constraints is extremely prone to type-1 error, that is, finding itemsets that are frequent (satisfy user-given constraints), but appear due to chance alone. [Web07] In statistics, there are two approaches for significance testing - analytical expressions or randomization tests. Gionis et al. focus on the latter approach and uses swap randomization in his work [GMMT07]. Webb proposes two ways of applying statistical tests to pattern discovery to set an upper limit on the risk of experimentwise error [Web07]. The first one divides the significance level α by the number of patterns (itemsets) in the search space in order to obtain the critical value κ (a Bonferroni correction for multiple tests ([Sha95])). The second one splits the dataset into exploratory and holdout sets (or train and test sets). It is then very similar to machine learning processes, the exploratory data are used for itemset mining (in ML this is training the model) which are then assessed by the holdout data (in ML evaluating the trained model on test data). More on the holdout approach [Web06]. Another method for distinguishing statistically significant patterns was developed by [KMP+12]. They offer a method for finding support $s*$ such that any itemset with support at least $s*$ represents a substantial deviation from itemsets in a random dataset with the same number of transactions and same item frequencies. Lastly, we would like to mention [HW19] and [SBM98] who consider statistically dependent patterns in their papers. Dependence being defined as the absence of independence.

3.2.1 The Chi-squared Test for Independence

Silverstein et al. propose measuring the significance of dependence via the chi-squared test for independence [SBM98]. In the supermarket settings they define $R = \{i_1, \overline{i_1}\} \times \cdots \times \{i_n, \overline{i_n}\}$ as all possible combinations (event sets) of presence or absence of items in a basket. Each $r = r_1 \cdots r_k \in R$ represents a basket value. When viewing R as a k-dimensional table, called a contingency table, each r also denotes a cell in this table. They then define $O(r)$ as the number of baskets in cell r. $O(r)$ has to significantly deviate from expected value in order for a cell r to be considered dependent. For single event Silverstein et al. use maximum likelihood estimators $E(i_j) = O_n(i_j)$ and $E(\overline{i_j}) = n - O_n(i_j)$. Assuming independence the expected count for sets of events is calculated as $E(r) = n \times E(r_1)/n \times \cdots \times E(r_k)/n$. They then define chi-squared statistic as:

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E(r))^2}{E(r)} \tag{3.1}$$

Finding out whether the k-items are k-way independent can be done by calculating the chi-squared statistics and obtaining p value which is corresponding to the statistics and a degrees of freedom count (always 1, for boolean variables). P value gives us the probability of observing the baskets if the variables were independent. If the probability is very small (usually between 0.05 and 0.0005) we reject the hypothesis that the variables are independent and say that the itemset is dependent at significance level α for $p \leq 1 - \alpha$.

For p value equal to 0.05 and one degree of freedom the χ^2 cutoff value is 3.84. Hence any itemset with $\chi^2 \geq 3.84$ is significant at the 95% confidence level.

Not all cells contribute to the dependence equally, so in order to give a more precise characterization of the dependence Silverstein et al. suggest the interest of a cell $I(r) = O(r)/E(r)$. The paper then shows that the cell with the highest interest (must be bigger than 1) is, in some sense, the most dependent cell in the contingency table. Interests below 1 indicate negative dependence. Note that the absolute value is meaningless as well as trying to compare interests from different contingency tables. Nonetheless if for example the two highest values were very close to each other, we could say that the corresponding cells have almost the same dependence.

3.3 Methods for Frequent Itemset Mining

Among the most common methods are Apriori algorithm ([AS+94], [AMS+96]) (deriving its name from Apriori property), using breadth first search to traverse its nodes in combination with a priori and a posteriori pruning, Eclat (alt. ECLAT, stands for Equivalence Class Transformation) ([ST04]), FP-Growth (Frequent Pattern Growth, [HPY00]) and LCM (Linear time Closed item set Miner) ([UAUA03], [UKA+04], [UKA05]) which all use depth first search with some form of divide-and-conquer strategy.

The fastest frequent itemset mining algorithms are currently the Eclat-variant LCM and FP-Growth [Bor12]. However, the challenge in this topic does not seem to be the speed, but rather filtering the produced frequent itemsets and discovering relevant patterns among them.

3.4 Closed, Maximal and K-Itemsets

In the problem of frequent itemset mining the number of itemsets can be enormously huge (depending on the chosen support) and thus some sets with restrictions were introduced. An itemset I is *closed (frequent)* if none of its immediate supersets have the same support count as I. This can be formally written as a frequent itemset I is called *closed* iff $\forall J \supset I : s_T(J) < s_T(I)$. An itemset I is *maximal (frequent)* if none of its supersets are frequent. This can be formally written as a frequent itemset I is called *maximal* iff $\forall J \supset I : s_T(J) < s_{min}$. Itemset I which contains K items is called a *K-itemset*. *K-itemset* is frequent if $K \leq$ minimum support count. [Bor12]

3.5 Association Rules

After obtaining frequent itemsets, so-called association rules [AIS93] can be generated. A frequent itemset is split into two disjoint subsets - one of them is used as the antecedent (X) and the other as the consequent (Y) of the rule. The confidence for a rule $X \rightarrow Y$ is computed as $c_T(X \rightarrow Y) = s_T(X \cup Y)/s_T(X)$, s_T being support in the transaction database T. Similarly to finding frequent itemsets the confidences of individual association rules

are compared against a user-specified minimum confidence c_{min} and only the ones with $c_T \geq c_{min}$ are returned.

Part II

Practical Part

Chapter 4

System Design/Architecture

As specified by the assignment we had three main objectives:

1. Convert the text data from all the sources into a suitable format for the following NLP and frequent itemset mining.
2. Implement a multi-label classifier using a RandomForestClassifier. Train, evaluate annotated dataset and choose suitable metric.
3. Find frequent itemsets in the dataset now represented as a set of sets of classes. Discuss gained insights and compare the frequent itemsets from different settings.

Analyzing those objectives resulted in the definition of features we would like to have in our solution:

1. Find ways to reasonably combine data from different sources (D1-likert answers; D2-answers to open-ended questions; D3-“Partial”/contextual answers; and D4-“spontaneous feedback”)
2. Automatic annotation (pipeline) for new data.
3. Select a suitable metric for measuring the performance of the classifier.
4. Find a way to compare a single company to the market average.
5. Identify the most important problems within a single company and the demographic groups it troubles.

6. Examine whether the frequent itemsets found on manually annotated data differ from the ones found on data classified by the classifier.
7. Come up with functions for advanced filtering of frequent itemsets.
8. Check the statistical significance of found frequent itemsets.

After examining the acquired data several questions arose and some decisions had to be made.

1. Every text record in the file could be treated separately or we could join all texts from one user. Although the classifier implementation would not be affected by either one, we decided on the latter format as the users often mentioned the same thing in multiple text answers and the former approach would ruin the insights gained from frequent itemsets.
2. When including the answers to follow-up questions we noticed that some of them do not involve any keywords that could be used in classification, so we came up with "artificial sentences" (explained in section 5.2.3) constructed from those answers.
3. The choice of programming language - Python - was quite straightforward. It is popular and we managed to find libraries implementing all the diverse tasks we were planning on doing - working with csv files, implementing a random forest classifier and mining frequent itemsets.
4. The data types used for storing the processed data were to a great extent given by functions of the libraries we chose.

All codes and sample data files can be found in this GitHub repository `https://github.com/crimsoncress/bachelor_thesis`.

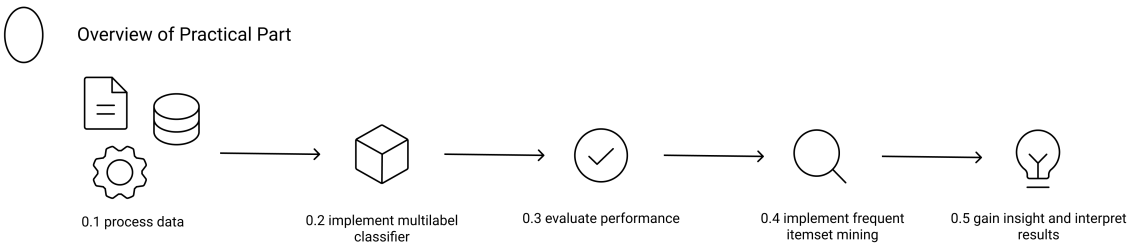


Figure 4.1: An overview of workflow in Practical Part, which is separated into 5 part.

Each part of the flow of data in figure 4.1 is further in this chapter explained in detail. Chapter 5 describes part 0.1 - process data, chapter 6 describes part 0.2, chapter 7 matches 0.3, chapter 8 describe 0.4 and last part 0.5 is explained in 9.

Chapter 5

The Data

We were given access to data collected over the period of two years. They consist of 3 csv files - button_answers.csv, bot_mentions.csv and demog_answers.csv. First one involves answers to several types of questions - general ones about the overall feeling from the company, 12 questions about satisfaction with various aspects of workplace (temperature, air, acoustics, light, ergonomics, culture, coffee and snacks, focus, cleanliness, design, relaxation, meetings), follow-up questions for finding out why specific area is perceived so low and two open-ended questions about biggest pain point, that should be fixed as soon as possible and three wishes for users' workspace. The file bot_mentions.csv involves spontaneous messages from users, nearly all of them turned out to be useless for the NLP since the users were mostly experimenting with the tool(chatbot) and exploring its functions. Last file demog_answers.csv holds predefined answers to demographic questions. First and second file have answers sometimes in english, sometimes in czech. Below are provided previews of mentioned files, more can be found in GitHub repository `https://github.com/crimsoncress/bachelor_thesis`

id	user	question	answer	timestamp
14	user21	How likely are you to recommend working in an office like yours to a friend or colleague?	5	1606204074
14	user21	Are you satisfied with the tidiness?	8	1606204105
14	user21	How well are you able to focus at work?	2	1606204130
14	user21	How could we make it better?	Can't focus much @ work	1606204156
14	user21	Do you feel like you have enough opportunities to restore energy during your workdays?	2	1606204199
14	user21	How could we make it better?	There is no off zone. Too small of an office.	1606204199
19	user7	What is the most important problem in your workplace that should be fixed right now?	1. I have a lot of work at the moment and I don't think anyone appreciates it. For example, a bonus., 2. According to the measures, we do not have enough offices. People have to move often.	1612792937

Table 5.1: An example of raw data from file button_answers.csv. The column "id" identifies the company, "user" is a unique user identifier, "question" is either a likert scale question or an open-ended one, followed by the answer to it in the "answer" column, last column "timestamp" is the time of user's response.

id	user	message	timestamp
14	user22	I hate my chair	1606396158
14	user23	No	1611593222
19	user17	Thank youuuuu!	1613133206
14	user27	Hi bot, you there?	1613135745
14	user27	How are you?	1613135778

Table 5.2: An example of raw data from file bot_mentions.csv. The column "id" identifies the company, "user" is a unique user identifier, "message" is a spontaneous message from a user, last column "timestamp" is the time of user's response.

id	user	question	answer	timestamp
19	user1	What is your gender?	female	1612793367
19	user1	How old are you?	2635	1612793371
19	user1	Do you work in an office or at home?	office	1612793376
14	user1	What is your gender?	male	1612794142
19	user2	How old are you?	3645	1612794145

Table 5.3: An example of raw data from file demog_answers.csv. The column "id" identifies the company, "user" is a unique user identifier, "question" has predefined options, last column "timestamp" is the time of user's response.

5.1 The Questionnaire

The data we obtained and analysed, was collected using a questionnaire which in total consisted of at least 16 questions (up to 28). From that at least 3 open-ended (up to 15) and the scale on the remaining being 1 - 10, 1 - Not at all, 10 - Definitely. It could be divided into four logical parts (with meaning provided below):

1. Initial questions - 1 likert scale, 1 open-ended
2. 12 topic questions - 12 likert scale, up to 12 open-ended
3. Biggest problem and three wishes - 2 open-ended
4. Demographic questions - 2 choose from predefined options

Initial question is the overall look at the company, employees' feelings about it and follow up with why. The 12 topics were chosen by domain experts as the ones with the biggest influence on employee wellbeing (wellbeing being defined as satisfaction and productivity). They include the following: temperature, air, acoustics, light, ergonomics, culture, coffee and snacks, focus, cleanliness, design, relaxation, meetings. If the users answer to topic question was 1 or 2 the respondent was given a follow up open-ended question "How could we make it better?" The whole questionnaire is available as an Appendix B.

5.2 Data Preprocessing

The challenge of compiling a user input is even greater when the input contains natural language. In the following section we will provide methods for preprocessing the existing textual data as well as an automatic pipeline for new data.

5.2.1 Outline

After assessing all obtained data files and reviewing the requirements of our solution we came up with this process for combining all the data from different question types and transforming it into a dataset suitable for NLP classifier as well as frequent itemset mining. The process can be divided into 4 parts:

- creating an empty dictionary for each user,
- filling the dictionary with processed data,
- labelling the text,
- saving the dictionaries incorporating all the data about a user,

that are explained in detail below.

5.2.2 User Structs (1)

For each user a blank struct is created, the struct is in a form of dictionary in our solution.

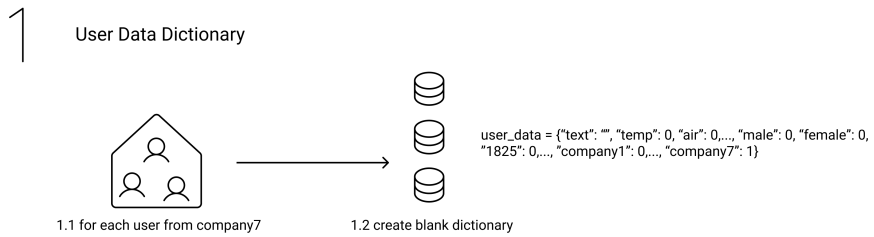


Figure 5.1: An illustration diagram of the first part of data processing - creating blank dictionaries.

```
users = get_all_users(company7) (1.1)
user_data = {}
for u in users:
    user_data[u] = {"text": "", "temp": 0, "air": 0, "acou": 0, ...,
    "male": 0, "female": 0, "1825": 0, "2635": 0, ..., "company1": 0,
    ...,"company17": 1} (1.2)
```

5.2.3 Data Files Processing (2)

The files (2.0) are described in part The Data with examples.

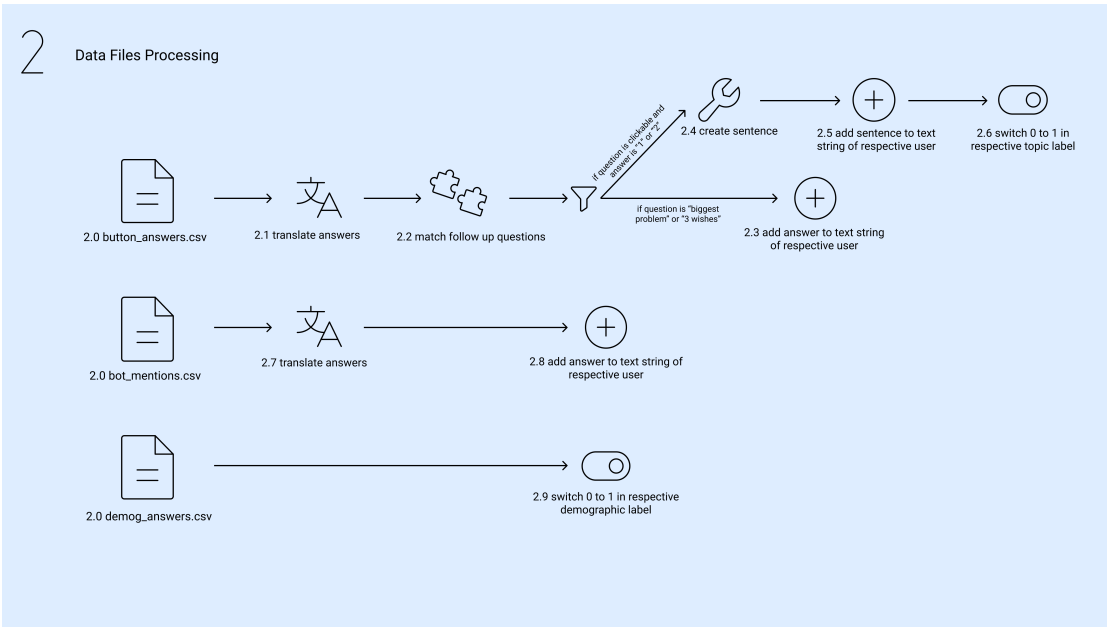


Figure 5.2: An illustration diagram of the second part of data processing - processing files and filling user dictionaries.

First steps (2.1, 2.7) in button_answers and bot_mentions pipelines are translations of czech answers to english. This was needed for unification of language. Bot_mentions did not need any further processing and so the answers were added to corresponding user (2.8):

```
for row in bot_mentions:
user = row[user]
user_data[user]["text"] += row[message]
```

Next part in the button_answers pipeline was matchmaking (2.2). Answers to the follow up questions "How could we make it better?" were saved on a different row in the csv than the question the follow up was related to. The user saw the follow up in relation to some topic question and in order not to lose the connection we matched the follow ups to questions they were related to.

id	user	question	answer	timestamp
19	user10	Do you feel like you have enough opportunities to restore energy during your workdays?	2	1606204130
19	user10	How could we make it better?	make transparent door non-transparent and feel OK to have power nap without interruptions	1606204156

Table 5.4: Example of how the answers were stored before matching the answers to follow-up questions to the topic questions.

id	user	question	answer	timestamp	answer2
19	user10	Do you feel like you have enough opportunities to restore energy during your workdays?	2	1606204130	make transparent door non-transparent and feel OK to have power nap without interruptions

Table 5.5: Example of how the answers were stored before matching the answers to follow-up questions to the topic questions.

After matching there were only two types of rows in the button_answers file that we were interested in. Either the question belonged to initial or topic questions and the answer to it was negative (1 or 2) 5.5 or the question was "What is the most important problem in your workplace that should be fixed

right now?" or "If you could wish for three changes in your work environment, what would they be?". In the former case an "artificial sentence" was created (2.4) using this equation - a random negative start of a sentence (chosen from manually created list - sent_sentences.csv) + the topic of the question + the answer to "How could we make it better?" question (which was after the match making in the same row).

Example of an "artificial sentence" created from the merged answer in 5.5: "I don't appreciate the relax zones, to improve: make transparent door non-transparent and feel OK to have power nap without interruptions"

The sentence was added to the user text string (2.5), this is the same as in 2.8. On top of that the topic the user marked as unsatisfied with was marked as one in his dictionary (2.6):

```
topic = get_question_topic(question)
user_data[user][topic] = 1
```

The latter case involved answers to open-ended questions Q1: "What is the most important problem in your workplace that should be fixed right now?" and Q2: "If you could wish for three changes in your work environment, what would they be?". Those answers were just added to the user text string like in 2.5 or 2.8.

Not all users answered all the questions and not all answers were useful (cca 2% were "I don't know" or just left blank). Note to why only answers from follow up questions with prior negative answers (1 and 2) were included: As we did not look for sentiment in the sentences we could not ensure that the topic in the sentence was mentioned in a positive way. Therefore we chose only answers in which we believe the topic is associated with a problem or a drawback (note that both the questions Q1, Q2 are meant in a "negative sentiment" - asking what the problems are).

As demog_answers contained only answers that were retrieved form button clicks, we found it to be easiest for processing.

```
gender_values = ["male", "female"]
age_values = ["1825", "2635", "3645", "4660", "60+"]
if answer in gender_values or age_values:
    user_data[user][answer] = 1
```

5.2.4 User Text Labeling (3)

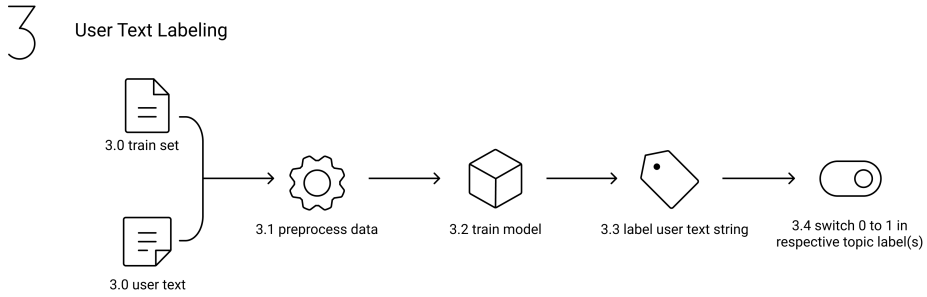


Figure 5.3: An illustration diagram of the third part of data processing - labelling joined texts from one user. Considering we have a train set (3.0), we can preprocess its texts (3.1), train the classifiers (3.2) and label (3.3) the user string (3.0). We then (similarly to 2.6) mark the corresponding found labels in user's dictionary (3.4).

Processing the files left us with strings containing answers to various questions. At this point we will employ the classifiers. This process is explained in figure 5.3. Considering we have a train set (3.0), we can preprocess its texts (3.1), train the classifiers (3.2) and label (3.3) the user string (3.0). We then (similarly to 2.6) mark the corresponding found labels in user's dictionary (3.4).

5.2.5 User Data Saving (4)

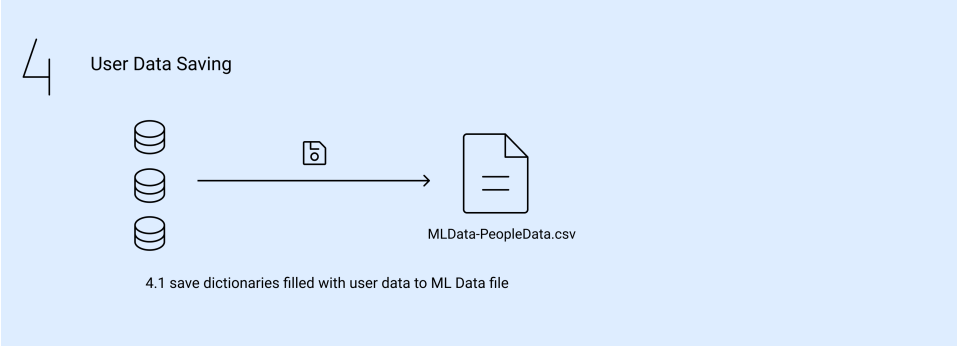


Figure 5.4: An illustration diagram of the fourth part of data processing - saving the dictionaries and creating employable dataset.

At this point the structs are filled with all the data we know about a user. We simply add them to a file containing the rest of our user data (4.1).

```
user_dataframes = []
for u in users:
    df = pd.DataFrame(user_data[u])
    user_dataframes.append(df)
with open('data/MLdata-ClassData.csv', 'a') as f:
    user_dataframes.to_csv(f, header=False, index=False,
    quoting=csv.QUOTE_NONNUMERIC, quotechar='"')
```

Below are some samples from the processed data, more can be found in GitHub repository `https://github.com/crimsoncress/bachelor_thesis`.

text	temp	···	male	female	1825	···	c_1	···
The minimum amount of natural light. No possibility for ventilation and fresh air. Non-adjustable tables, chairs non-ergonomic. Minimum greenery.	0	···	0	1	0	···	0	···
I don't appreciate the relax zones, to improve: make transparent door non-transparent and feel OK to have power nap without interruptions. Some space to move. Clean desk	0	···	1	0	1	···	0	···

Table 5.6: An example of raw data from file demog_answers.csv. (c_1 = company1)

Chapter 6

Implementation of NLP Classifier

After reviewing approaches for tackling the Multi-Label classification problem we decided to implement binary relevance as there might be (class) dependency in our data, but we do not know for sure and do not want the classifier to take it into account. Random Forest (RF) has been suggested as a baseline classifier - likely for its balance between accuracy and moderate requirements for implementation, and its interpretability. Figure 6.1 shows steps for implementing the RF classifier that are further described in following parts:

1. Splitting the dataset into stratified train and test sets
2. Extracting the features by vectorizing the data
3. Training the classifiers using the extracted features
4. Labelling new data

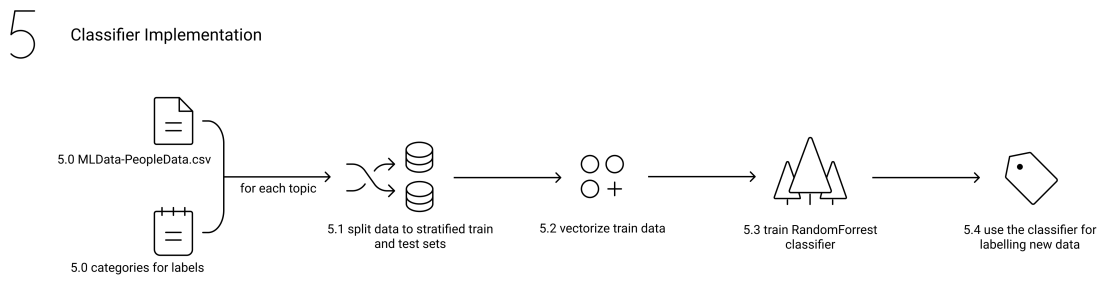


Figure 6.1: An illustration diagram of the implementation of NLP classifier.

6.1 Splitting the Data into Stratified Train and Test Sets (5.1)

Splitting the data is usually the first step of almost every supervised learning method. The usual ratio is 70/30 or 80/20 for train/test. In cases where we don't have large datasets 98/2 is commonly used. Our dataset has numerous labels and each label has different frequency (unbalanced labels) and appears in different texts. Therefore for each of those labels it is needed to stratify the split of the data accordingly, that is, in a way that both train and test sets contain approximately the same percentage of samples of each target class. Below is outlined the train-test split for each label and how the data is stored as a dictionary. "Topics" is an array of all topics (temperature, air...) and "i" is the number of column corresponding to the label. We arranged that the array of labels is ordered by the column headers in the dataframe.

```
df = pd.read_csv("data/MLdata-PeopleData.csv")

for topic, i in zip(topics, range(1, len(topics)+1)):
    train, test = train_test_split(df, train_size=0.8, shuffle=True,
    stratify=df[topic])
    nlp[topic]["text"]["train"] = train.iloc[:, 0].values
    nlp[topic]["text"]["test"] = test.iloc[:, 0].values
    nlp[topic]["train"] = train.iloc[:, i].values
    nlp[topic]["test"] = test.iloc[:, i].values
```

6.2 Vectorizing the Data (5.2)

As we chose one of the supervised learning methods some data preprocessing and feature extraction is required. This is vital for the model's ability to learn and make correct predictions. In the field of NLP this includes parsing/tokenization - splitting the document into sentences and words, removing the uninformative words (stopwords) such as "I, are, a, the"; stemming - cutting the prefixes and suffixes and lemmatization - mapping words like gone, going, went onto one word - go. We can then adjust some more parameters - maximum features extracted, number of words that pose as one feature, minimum or maximum term/document frequency, etc. At the end of this process, each raw text has its row in the dataset and the columns correspond to words and their frequencies or some other kind of representation such as TF/IDF (Term Frequency/Inverse Document Frequency, [SFW83]).

For each label an individual matrix of TF-IDF features (X) is created, see below.

```
Xs = {}

for topic in topics:
    Xs[topic] = vectorizers[topic].fit_transform(nlp[topic]["text"]["train"])
    .toarray()
```

6.3 Fitting/Training the Classifier (5.3)

We now possess an array of features for each of the labels (Xs[topic]) . We will build trees for the forest and fit those features (training input samples) to previously extracted labels of the train data.

```
for topic in topics:
    nlp[topic]["classifier"] = RandomForestClassifier(n_estimators=200)
    nlp[topic]["classifier"].fit(Xs[topic], nlp[topic]["train"])
```

6.4 Label New Data (5.4)

Successfully finishing part 5.3 we now have a set of trained classifiers for each label ready to predict. A new string (text, document) is transformed to a document-term matrix and then each classifier is employed to yield a prediction, creating a binary array, where 1 indicates the corresponding label is contained in the text.

```
predicted_labels = []
for topic in topics:
    data = vectorizers[topic].transform([text]).toarray()
    preds = nlp[topic]["classifier"].predict(data)
    predicted_labels.append(preds[0])

return np.array(predicted_labels)
```

Chapter 7

Model Evaluation

In chapter 5 (see 5.3) we use the classifiers to evaluate new user answers, we want to know how much we can trust those predictions. This chapter is dedicated to evaluating the model.

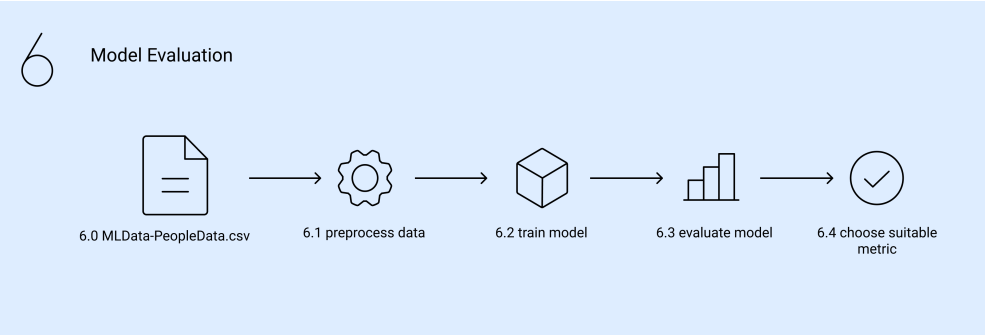


Figure 7.1: An illustration diagram of evaluation of NLP classifier.

Considering specifications of our task which are: NLP, labeling, unbalanced, possibly mislabeled data, etc. we consider the following metrics: accuracy, balanced average, logarithmic loss, macro F1, weighted F1 and kappa.

Fig. 7.2 shows how substantially some of the topics are unbalanced. This is important for understanding why cross validation and stratification when splitting the dataset into test and train (train and test sets contain the same percentage of classes) was crucial.

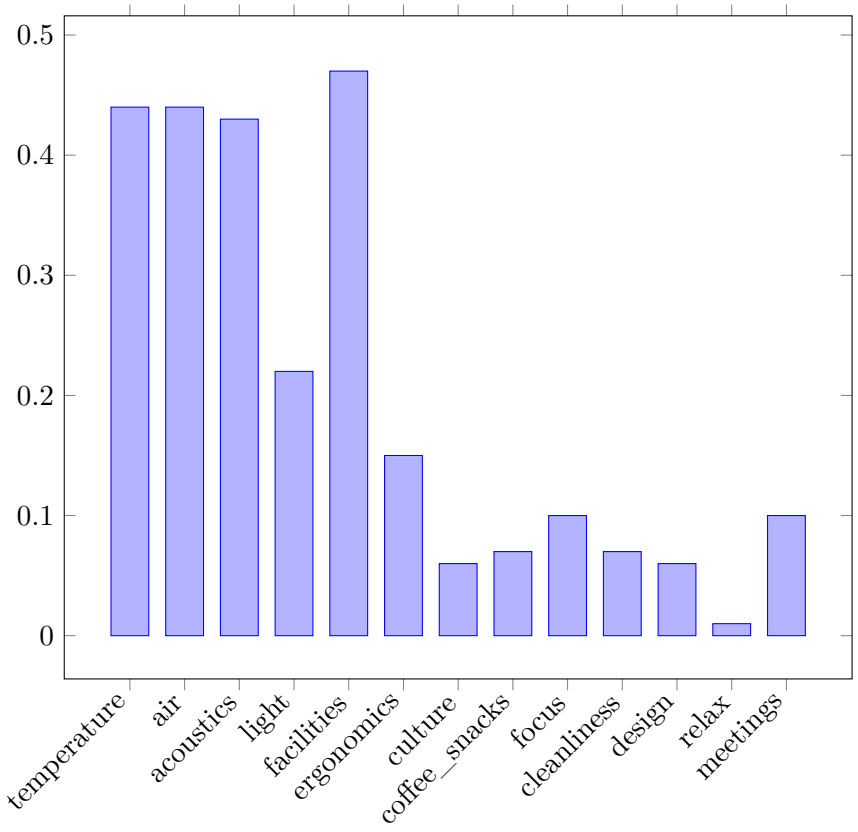


Figure 7.2: Percentage representation of each topic label.

	accuracy	balanced accuracy	log loss	macro F1	weighted F1	kappa
temperature	0.932	0.930	2.342	0.931	0.932	0.862
air	0.933	0.933	2.305	0.932	0.933	0.865
acoustics	0.871	0.865	4.462	0.867	0.870	0.735
light	0.952	0.911	1.673	0.927	0.950	0.855
facilities	0.707	0.704	10.114	0.705	0.706	0.410
ergonomics	0.945	0.846	1.897	0.879	0.941	0.759
culture	0.936	0.499	2.193	0.484	0.907	0.002
coffee/snacks	0.940	0.569	2.082	0.604	0.918	0.226
focus	0.901	0.539	3.421	0.545	0.864	0.126
cleanliness	0.945	0.598	1.896	0.644	0.928	0.300
design	0.952	0.500	1.673	0.488	0.928	0.000
relaxation	0.948	0.547	1.785	0.565	0.928	0.149
meetings	0.936	0.743	2.194	0.786	0.929	0.575

Table 7.1: Results for all topics and the chosen metrics. The data was stratified and the results are 5-fold cross validated.(part in 7.1) As suitable metrics for our task are considered balanced accuracy and macro F1.

As mentioned earlier some of the classes in our dataset are highly unbalanced and so plain accuracy expectedly gave us a false sense of doing very well even on those. Second one is balanced accuracy which - as the name suggests - takes into account the a priori probabilities of each of the labels on the classes. Balanced accuracies on balanced topics in the dataset were very similar to results of previously mentioned accuracy. With differences as low as 0.018. However, given a highly unbalanced topic for example culture where the a priori probability is 0.063 balanced accuracy dropped significantly. Specifically by 0.437 in the culture example.

One of the most unbalanced topics - culture will be a nice example again. Macro F1 for culture is 0.685 while the weighted version gives us 0.938. The huge difference between the two metrics reveals that our model performs poorly only on the rarer class. Given that we know 1 is the rarer class it can be concluded that the model does not recognize the topic culture in the text very well.

Interpreting Cohen's Kappa score is not straightforward at all given the imbalance in our dataset. As mentioned in chapter 2.3.3 the first thing we have to keep in mind is that the maximum value of kappa depends on the difference between distribution of the predicted and actual target classes. Secondly different papers define the fair performance of a classifier based on the kappa value differently, see 7.2 below.

	Landis and Koch	Cicchetti and Sparrow	Fleiss
<0.01	Poor	Poor	Poor
.00 - .20	Slight		
.21 - .40	Fair		
.41 - .60	Moderate	Fair	Fair to good
.61 - .75	Substantial	Excellent	
.76 - .80			Excellent
.81 - 1.00	Almost Perfect		

Table 7.2: Results for all topics and the chosen metrics. The data was stratified and the results are 5-fold cross validated.(part in 7.1)

While Cicchetti and Sparrow consider .61 excellent Landis and Koch moved this boundary up to .81 and .61 consider only substantial.

7.1 Selected Metrics for Our Task

Our task deals with a quite delicate subject - employees' wellbeing. Therefore some problems might be only subtly indicated in text and hard for the classifier to detect. We also do not want to replace HR as we believe only people can truly understand other people and perceive this task as only an enhancement of HR's existing methods. Recognizing all this we are preferring more false positives even at the cost of more true negatives ergo we prefer the classifier falsely predicting that the text contains the topic.

That said, we consider balanced accuracy and macro F1 suitable metrics for our task if using RandomForest Classifier (part 5.4 in 7.1). Both of them give both classes the same weight unlike accuracy or weighted F1. They are also easily interpretable in contrast with kappa where the upper limit can vary. If using some form of neural network with the need for backpropagation, logarithmic loss would probably be our metric of choice.

Chapter 8

Frequent Itemset Mining Implementation

In the following chapter we will implement frequent itemset mining. As we use a state of the art library for the task we need to ensure the data is in required format - represented as sets of binary labels. We then compute the itemsets, and discuss the meaning of the support and its interpretation. We discuss the issue of comparing a single entity (a company) to the whole dataset.

The illustration below shows the data flow from raw data to the found itemsets.

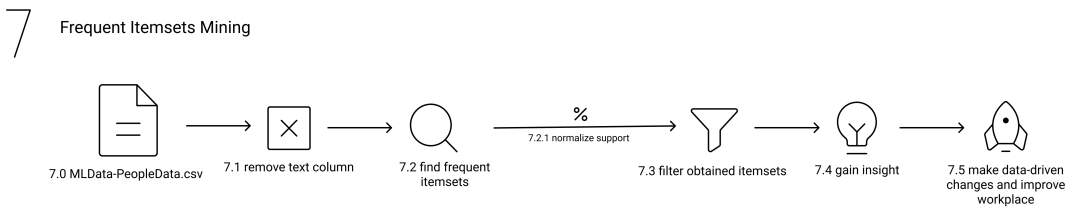


Figure 8.1: An illustration diagram of mining frequent itemsets.

8.1 Used Functions and Libraries

In our solution, we implemented the frequent itemsets method with the help of python library mlxtend. It offers Apriori frequent pattern mining algorithm as well as FP-Growth frequent pattern mining algorithm. The difference between those two is the speed. Note considering that our dataset is quite small and our focus is on filtering and gaining insight, we do not aim for the fastest algorithm, but one that is reasonably well implemented and easy to use. FP-Growth does not require candidate generation. Instead it uses a so-called FP-tree (frequent pattern tree) - a datastrucure without generating the candidate sets explicitly, which makes it especially handy when dealing with large datasets and not being patient. Lastly the library involves FP-Max algorithm - a variant of FP-Growth, which focuses on obtaining maximal itemsets.

8.2 Dataset Preparation

The first thing we needed to do was prepare our dataset. Functions implementing the algorithm for frequent itemset mining that we chose require the data to be in binary form. So when creating the dataset in part Data Preprocessing we already knew the specifics and so the dictionary for user data was equipped with variables that were not needed for classification, but only for frequent itemset mining. Those were categorical variables that represented information about the users: gender - male, female, age group - 18-25, 26-35, 36-45, 46-60, 60+ and company they are employed at - company1, company2, ..., company17. During the data preprocessing this categorical data was one-hot encoded [CVK18], that is for every possible value/category of each attribute, a separate column had to be made. The last part of transforming the data into binary form (as required by functions from mlxtend) was deleting the column with texts (part 6.1 in 8.1).

After meeting the demands of our chosen library, we required some more conditions to be met:

- df["male"] + df["female"] == 1,
- the sum of all age categories equals to one,
- the sum of all company columns equals to one,

As a person cannot have two genders, belong to more than one age group or work at two companies.

temperature	air	···	male	female	1825	···	c_1	···
0	0	···	0	1	0	···	0	···
1	0	···	1	0	1	···	0	···
0	0	···	0	1	0	···	1	···
0	1	···	1	0	1	···	1	···

Table 8.1: Head of data prepared for frequent itemset mining. (c_1 = company1)

8.3 Mining Frequent Itemsets (7.2)

Firstly we used the `mlxtend.frequent_patterns.apriori()` for computing the a priori probabilities, one of the parameters is min_support which tells us how many percent of data is following the rule. We set it as low as 0.001 as we can specify this number later on when we are filtering the frequent itemsets.

```
frequent_itemsets = apriori(df, min_support=0.001)
```

The second part was counting the number of items in each item set and saving this number for filtering.

```
frequent_itemsets['length'] = frequent_itemsets['itemsets']
.apply(lambda x: len(x))
```

The last thing we would like to filter the itemsets by is if the itemset contains a certain item, such as "male", "company17", or "air".

Using all of the options mentioned above we were able to create something like querying the database. We can specify minimum support - how frequently the items occur together in the database), length - how many items are there in the itemset and lastly fname - an item of interest contained in the itemsets (part 7.3 in 8.1).

```
frequent_itemsets[(frequent_itemsets['support'] >= 0.02) &
(frequent_itemsets['length'] > 1) &
(frequent_itemsets['itemsets'].astype(str).str.contains(fname))]
```

Chapter 9

Interpretation of Found Frequent Itemsets

In the previous chapter we have shown a method to obtain frequent itemsets from the data. The following part focuses on interpretation of the sets, their support, and independence of items. Such knowledge is used to analyse significance of the found itemsets and help us interpret them in terms of real life relevance, outcomes, etc. The relativity of support and significance of found frequent itemsets will be discussed.

9.1 The Relativity of Support

The support of itemset from the whole dataset is proportional to the item frequencies of all companies combined, so if a company has only 50 employees and the dataset contains over 900 entries, then the support of frequent itemsets regarding the company can be quite small and therefore might seem unimportant. If one is interested in only a specific company and want the minimum support to be relative only to the company it is possible to pick only the rows from dataframe which belong to the company `df = df.loc[df["company17"] == 1]` (part 7.2.1 in 8.1).

In our example we take two different dataframes:

```
//file with frequent itemsets dataset
df1 = pd.read_csv("data/PeopleData-notext.csv")
```

```
//choose only rows with messages from employees of company4
df2 = df1.loc[df['company4'] == 1]

frequent_itemsets1 = apriori(df1, min_support=0.01)
frequent_itemsets2 = apriori(df2, min_support=0.01)
```

For the same "query request", where fname = 'company4':

```
frequent_itemsets[(frequent_itemsets['support'] >= 0.02) &
(frequent_itemsets['length'] > 1) &
(frequent_itemsets['itemsets'].astype(str).str.contains(fname))]
```

We get exactly the same itemsets, but with different support. See below first 10 items meeting our criteria, descending by support.

support1	support2	itemsets	length
0.060703	0.575758	(26-35, company4)	2
0.059638	0.565657	(company4, male)	2
0.053248	0.505051	(air, company4)	2
0.050053	0.474747	(company4, acoustics)	2
0.048988	0.464646	(company4, light)	2
0.046858	0.444444	(facilities, company4)	2
0.045793	0.434343	(company4, female)	2
0.034079	0.323232	(26-35, company4, male)	3
0.031949	0.303030	(air, company4, light)	3
0.031949	0.303030	(air, 26-35, company4)	3

Table 9.1: Support 1 is support of the itemset in the whole dataset. Support 2 is support of the itemset in dataset of only company 4.

One can see that the itemsets are the same, but the support is different. Meaning that this way we have the support relative only to company4 and it is, therefore, higher and the frequency of the itemsets might be more apprehensible.

Another way of achieving the support with respect only to one company (or any other one item) (i.e. transforming the support values from the first example to the second one) can be done by multiplying the support by the apriori probability of company4 i.e. the percentage of rows (employees) w.r.t. the whole dataset.

9.2 Statistical Significance of Found Frequent Itemsets

There is one more obvious question that begs for an answer. Considering the whole dataset one might be tempted to say some unexpected correlations were discovered. E.g. If we compare the support for itemsets (18-25, temperature) and (25-36, temperature) the support is 0.066028 and 0.223642 respectively, a huge difference - but only seemingly. It is important to remember that the different age groups have different apriori probabilities in the dataset.

We offer two ways for avoiding spurious patterns. The basis of the first one was mentioned in chapter The Relativity of Support. Below is an illustration of how to treat the example situation in the first paragraph.

We have to first filter only the age groups by using `df = df.loc[df['18-25'] == 1]` or divide the support with apriori probability of the age group as mentioned above and work with those numbers.

Doing this it can be found that roughly 48% of employees aged 18-25 think temperature is a problem at their workplace compared to employees aged 26-35 of which circa 46% considers temperature a problem. The difference is actually marginal.

This method normalizes in some way the support values and allows us to compare them, but discloses nothing regarding the statistical significance. Therefore we present the second approach developed by [SBM98] and reviewed in part Statistically Sound Pattern Discovery of this thesis.

	female	not female	row sum
acoustics	192	210	402
not acoustics	274	263	537
col sum	466	473	939

Table 9.2: Contingency table for female and acoustics.

$E(acoustics) = O(acoustics) = 402$, $E(female) = O(female) = 466$

The chi-squared value is

$$\frac{(192 - 402 \times 466/939)^2}{402 \times 466/939} + \frac{(210 - 402 \times 473/939)^2}{402 \times 473/939} + \frac{(274 - 537 \times 466/939)^2}{537 \times 466/939} + \frac{(263 - 537 \times 473/939)^2}{537 \times 473/939} = 0.282 + 0.278 + 0.211 + 0.208 = 0.979 \tag{9.1}$$

As mentioned earlier the chi-squared cutoff value for $p = 0.05$ and $\eta = 1$ is 3.84 and because 0.979 is less we do not reject the independence hypothesis at the 95% confidence level.

The statistical test tells us that the independence between those two variables is very likely, while the support for this itemset is one of the highest with 0.204(192/939).

	company17	not company17	row sum
coffee and snacks	51	15	66
not coffee and snacks	304	569	873
col sum	355	584	939

Table 9.3: Contingency table for coffee and snacks and company17.

Let us show another example where the chi-squared value is 47.044 which is more than 3.84 and therefore significant at the 95% significance level, whereas the support of this itemset is 0.054(51/939.

	company17	not company17
coffee and snacks	2.046	0.365
not coffee and snacks	0.921	1.046

Table 9.4: Interest of cells from 9.3.

The obviously standing out value in the up left cell shows significant dependence between being dissatisfied with coffee and snacks and being an employee of company 17.

There is also quite a large negative dependence between being dissatisfied with coffee and snacks and not being an employee of company 17.

9.3 Frequent Itemsets in Annotated Data vs. Data Labeled by the Classifier

Aim of this chapter is to evaluate whether using the developed classifier somehow (negatively) affects the found frequent itemsets. To test this, we develop and train a classifier, then create a "classifier's dataset" by detecting labels in our (text) dataset. As a comparative method (ground truth) we use manually labeled sentences. We then find frequent itemsets on each of these new datasets, and compare whether the found frequent itemsets could be considered the same. When researching the most common problems of employees of a new company the classifier will predict the topics with some accuracy, but we do not know whether the accuracy will be high enough and the frequent itemsets found on those data will be meaningful. Therefore we concluded an experiment to find out whether using the classifiers predictions would affect the frequent itemsets distinctly. We recreated the 5-fold cross validation process. Trained the classifier using each of the train sets and classified the test sets. We then found frequent itemsets with minimum support = 0.2 on the original dataset (9.5) and on a dataset we created by merging the test sets where the labels were as predicted by the classifier (9.6).

id	support	itemset
A	0.267306	(female, temperature)
B	0.244941	(female, air)
C	0.239617	(facilities, male)
D	0.238552	(air, temperature)
E	0.232162	(facilities, female)
F	0.230032	(facilities, 26-35)
G	0.223642	(temperature, 26-35)
H	0.223642	(male, acoustics)
I	0.219382	(air, 26-35)
J	0.218317	(temperature, company17)
K	0.209798	(facilities, company17)
L	0.204473	(female, acoustics)

Table 9.5: Frequent itemsets found on the original dataset.

id	support	itemset
A	0.268371	(female, temperature)
B	0.251331	(female, air)
D	0.250266	(air, temperature)
H	0.223642	(male, acoustics)
I	0.223642	(air, 26-35)
E	0.220447	(facilities, female)
C	0.219382	(facilities, male)
F	0.216187	(facilities, 26-35)
G	0.211928	(temperature, 26-35)
J	0.204473	(temperature, company17)

Table 9.6: Frequent itemsets found on the dataset where data was labeled by the classifier.

Both sets contain almost exactly the same itemsets - the one with the classified data misses K - (facilities, company17) and L - (woman, acoustics). The support of itemset K is 0.187433 and of L it is 0.186368 in the dataset with classifier predicted labels. The order of the itemsets (descending by support) is shuffled, but the supports differ only marginally - maximum absolute difference is 0.020235 and minimal is only 0.001065 (considering only itemsets containing at least one topic item). The biggest differences in support are between itemsets with facilities (remember, that the only itemset missing also contains facilities). Classifier for the label facilities was one of not so well performing (macro F1 = 0.705), so this is expected.

The performance of our set of classifiers was fairly solid on labels with high frequency (temperature, air) and mediocre on labels with few samples (design, culture). It was easier for itemsets containing items (topics/labels) with high frequency to fulfill the user-specified minimum support and thus those items naturally appear in more frequent itemsets. Taking those facts we concluded that if the classifier is very likely to recognize the topics that most frequently appear in the found frequent itemsets, then the ones found on the classifier annotated data will not differ much from the ones mined from the manually annotated data (ground truth). As our classifier does not perform well regarding the non frequent labels (most misclassifications are false negatives) we would recommend to lower the minimum support when mining itemsets on classifier annotated data. In conclusion the found frequent itemsets are similar using both methods for labelling, so we can assume it is safe to use the automation (classifier) in the preprocessing pipeline.

9.4 Comparing a Single Company to the Whole Dataset

Comparing frequent itemsets from the whole dataset and from a single company is hugely dependent on having a large enough dataset to derive meaningful statistics. It is crucial that one company is not dominant in the dataset, that is easily tested by the following check (in equation 9.2 s is support):

$$\forall \text{company} : s_{wholedataset}(\text{itemset}) \approx s_{dataset-company}(\text{itemset}) \tag{9.2}$$

Note that comparing a company to the whole dataset is too general and therefore can prove to be meaningless. We suggest comparing the company to only a group of companies that are alike or might deal with similar issues (e.g. due to the same office location). For example if our company's industry is IT and the dominant group are males aged 18-50 we would probably want to compare with another IT company.

Below we provide examples of frequent itemsets found on company 15 in comparison with frequent itemsets found on dataset without this company's data.

id	support	itemset
A	0.310345	(female, facilities)
B	0.275862	(female, temperature)
C	0.275862	(acoustics, male)
D	0.275862	(facilities, acoustics)
E	0.275862	(female, acoustics)
F	0.241379	(meetings, acoustics)
G	0.206897	(facilities, male)
H	0.206897	(air, facilities)
I	0.206897	(air, female)
J	0.206897	(meetings, facilities)
K	0.172414	(meetings, acoustics, male)
L	0.172414	(air, acoustics)
M	0.172414	(light, facilities)

Table 9.7: Frequent itemsets found on data from company 15.

id	support	itemset
B	0.267033	(female, temperature)
I	0.246154	(female, air)
G	0.240659	(facilities, male)
A	0.229670	(facilities, female)
C	0.221978	male, acoustics)
E	0.202198	(female, acoustics)
H	0.194505	(facilities, air)
	0.190110	(facilities, temperature)
M	0.180220	(acoustics, air)
D	0.179121	(facilities, acoustics)
	0.158242	(acoustics, temperature)

Table 9.8: Chosen frequent itemsets found on the dataset without company 15.

A lot of frequent itemsets can be found in both of those sets, some of them have similar support (G, H, I, L, average difference is 0.023). While the rest have bigger differences ranging from 0.053 to 0.096. Some of those differences can be likely caused by the fact that some of those itemsets contain "female" and the employees of company 15 are 69% females, while the whole dataset is gender balanced. Whereas others like D - (facilities, acoustics) clearly show that this itemset constitutes an above average problem in company 15.

Itemsets F, J, K and M do not have a counterpart, so we can assume that they pose a truly significant problem for the company.

9.5 Frequent Itemsets Tracing to Real-Life Problems

Trying to deduce what truly upsets people by considering one number is meaningless. However, going through significant frequent itemsets can bring an insight about what to focus on when reading the answers or discussing improvements during a meeting. We propose a method for obtaining the insights while minimizing the risk of accepting spurious patterns.

1. Find frequent itemsets and order them by support.
2. From top down check whether the items in itemsets are dependent, using chi-squared test.

3. Check whether the itemset can make sense. (It is very unlikely that snacks and light might be actually dependent.)
4. Filter out texts with labels corresponding to your itemset.
5. Read some of them and try to determine what is the reason for the dependency of those items or how could this be interpreted.

Below we provide possible interpretations of such itemsets after reading some of the user texts with corresponding labels.

- (acoustics, focus)
  - support: 0.067093
  - chi-squared value: 20.614
  - explanation: Loud and/or sudden noises often interrupt one's deep work, the noises can be door slamming, colleagues talking to each other or on the phone etc.
  - samples:
    - Noise from others (no noise cancellation in headphones. Hearing doors slam, colleagues talk etc.
    - There is always someone running, stomping.
    - Closed but small room for 11 people - everyone hears everything and the noise can not be switched off.
- (temperature, air)
  - support: 0.238552
  - chi-squared value: 26.312
  - explanation: Stale and stuffy air is often associated with higher temperatures, second common explanation is cold air from air condition blowing too close to one's working station.
  - samples:
    - Strange functioning air conditioning. My work place is cold and still on me blowing cool air
    - Air circulation - sometimes stuffy, due to a late response of the air temperature outside, that is too hot or cold
- (facilities, temperature)
  - support: 0.187433
  - chi-squared value: 7.437

- explanation: Unsatisfactory temperature is often associated with not working air conditioning (or badly set) or the inability to open windows (in new office buildings)
- samples:
  - Better air quality (maybe change the windows so it's possible to open up a little and get fresh air?)
  - inoperable air conditioning, heating
  - Bad air in the office - air conditioning is either cooling so that everyone in office wears jackets, or is not working whole day

- (facilities, cleanliness)
  - support: 0.043663
  - chi-squared value: 9.570
  - explanation: This itemset was very frequent in one company where the users answers regarded the cleanliness of toilets, kitchen spaces, and sometimes colleagues desks and the absence of soap.
  - samples:
    - that they would fix the tap in the bathroom cause it is broke for months and also they do not refill the soap
    - Ladies toilets in very poor conditions (in regards of cleanliness), soap often not available.
    - proper cleaning, clean desk policy for shared seats

9.6 Possible Adjustments to the Topics

The topics/labels in our data are given by experts. We want to check that the labels are optimal in a sense of (semantically) non-overlapping partitions. The library mlxtend we chose for the implementation of mining frequent itemsets offers in addition to apriori and fpgrowth functions a function for computing maximal itemsets - fpmax(). Using the FP-Max function on our dataset, sorting the results, picking nine with the highest support we obtain following results:

support	itemset
0.0224	(focus, 26-35, male, acoustics)
0.0224	(46-60, male, acoustics)
0.0213	(female, company15)
0.0213	(company11, 26-35, male, facilities)
0.0213	(temperature, female, company17, air, 26-35, facilities)
0.0202	(male, facilities, meetings)
0.0202	(light, temperature, male, acoustics)
0.0202	(light, male, 36-45, acoustics)
0.0202	(air, male, 46-60)

Table 9.9: Results of maximal itemsets on the whole dataset, sorted descending by support.

Based on the first row we suggest merging focus and acoustics into one topic (e.g. deep work). This would make sense as the topic acoustics is associated with noises that can disturb your flow. In contrast merging age group - “46-60” and gender - “male” does not make sense as those are clearly independent.

Chapter 10

Conclusion

First, we developed a pipeline to automatically process the obtained data and transform it into a format suitable for NLP. Contextual answers, posing as the biggest challenge, were combined with topic questions and "artificially" combined into one sentence with self-contained meaning.

Further on we trained an NLP classifier (RandomForest) that performs the labeling task. We evaluated its performance using suitable metrics. Comparison with the ground truth (manually annotated dataset) confirmed that the results are indeed well sufficient for the use for frequent itemset mining.

To perform the frequent itemset mining, the categorical data were one-hot encoded as needed by the functions.

We suggested a way for modifying the support to be proportionate to only one item and a way for determining the (in)dependence of items in frequent itemset using the chi-squared test for independence.

Lastly, we offered an interpretation of some of the frequent itemsets in real life by reading some of the users texts and suggested possible adjustment to the topics - merging acoustics and focus.

Appendices

Appendix A

Bibliography

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami, *Mining association rules between sets of items in large databases*, Proceedings of the 1993 ACM SIGMOD international conference on Management of data, 1993, pp. 207–216.

[All71] David M Allen, *Mean square error of prediction as a criterion for selecting variables*, Technometrics **13** (1971), no. 3, 469–475.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al., *Fast discovery of association rules.*, Advances in knowledge discovery and data mining **12** (1996), no. 1, 307–328.

[AS+94] Rakesh Agrawal, Ramakrishnan Srikant, et al., *Fast algorithms for mining association rules*, Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, Citeseer, 1994, pp. 487–499.

[Bla08] Martin Bland, *Cohen's kappa*.

[Bor12] Christian Borgelt, *Frequent item set mining*, Wiley interdisciplinary reviews: data mining and knowledge discovery **2** (2012), no. 6, 437–456.

[BOSB10] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann, *The balanced accuracy and its posterior distribution*, 2010 20th international conference on pattern recognition, IEEE, 2010, pp. 3121–3124.

[Bur98] Christopher JC Burges, *A tutorial on support vector machines for pattern recognition*, Data mining and knowledge discovery **2** (1998), no. 2, 121–167.

[CVK18] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl, *Similarity encoding for learning with dirty categorical variables*, Machine Learning **107** (2018), no. 8, 1477–1494.

[Fis03] Cynthia D Fisher, *Why do lay people believe that satisfaction and performance are correlated? possible sources of a commonsense theory*, Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior **24** (2003), no. 6, 753–777.

[GFB+11] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera, *A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42** (2011), no. 4, 463–484.

[GM05] Nadia Ghamrawi and Andrew McCallum, *Collective multi-label classification*, Proceedings of the 14th ACM international conference on Information and knowledge management, 2005, pp. 195–200.

[GMMT07] Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas, *Assessing data mining results via swap randomization*, ACM Transactions on Knowledge Discovery from Data (TKDD) **1** (2007), no. 3, 14–es.

[HCRdJ16] Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María J. del Jesus, *Multilabel classification*, pp. 17–31, Springer International Publishing, 2016.

[HHC66] Petr Hájek, Ivan Havel, and Michal Chytil, *The guha method of automatic hypotheses determination*, Computing **1** (1966), no. 4, 293–308.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin, *Mining frequent patterns without candidate generation*, ACM sigmod record **29** (2000), no. 2, 1–12.

[HW19] Wilhelmiina Hämäläinen and Geoffrey I Webb, *A tutorial on statistically sound pattern discovery*, Data Mining and Knowledge Discovery **33** (2019), no. 2, 325–377.

[KMP+12] Adam Kirsch, Michael Mitzenmacher, Andrea Pietracaprina, Geppino Pucci, Eli Upfal, and Fabio Vandin, *An efficient rigorous approach for identifying statistically significant frequent itemsets*, Journal of the ACM (JACM) **59** (2012), no. 3, 1–22.

[Kot13] Sotiris B Kotsiantis, *Decision trees: a recent overview*, Artificial Intelligence Review **39** (2013), no. 4, 261–283.

[McC] James McCaffrey, *Log loss and cross entropy are almost the same*, `https://jamesmccaffrey.wordpress.com/2016/09/25/log-loss-and-cross-entropy-are-almost-the-same`, Accessed: 2021-04-10.

[OB19] Juri Opitz and Sebastian Burst, *Macro f1 and macro f1*, arXiv preprint arXiv:1911.03347 (2019).

[Pal05] Mahesh Pal, *Random forest classifier for remote sensing classification*, International journal of remote sensing **26** (2005), no. 1, 217–222.

[RPHF21] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank, *Classifier chains: a review and perspectives*, Journal of Artificial Intelligence Research **70** (2021), 683–718.

[SBM98] Craig Silverstein, Sergey Brin, and Rajeev Motwani, *Beyond market baskets: Generalizing association rules to dependence rules*, Data mining and knowledge discovery **2** (1998), no. 1, 39–68.

[SFW83] Gerard Salton, Edward A Fox, and Harry Wu, *Extended boolean information retrieval*, Communications of the ACM **26** (1983), no. 11, 1022–1036.

[Sha95] Juliet Popper Shaffer, *Multiple hypothesis testing*, Annual review of psychology **46** (1995), no. 1, 561–584.

[ST04] Lars Schmidt-Thieme, *Algorithmic features of eclat.*, FIMI, 2004.

[UAUA03] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura, *Lcm: An efficient algorithm for enumerating frequent closed item sets.*, Fimi, vol. 90, Citeseer, 2003.

[UKA+04] Takeaki Uno, Masashi Kiyomi, Hiroki Arimura, et al., *Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets*, Fimi, vol. 126, 2004.

[UKA05] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura, *Lcm ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining*, Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations, 2005, pp. 77–86.

[WDH+20] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Henry Jin, Suzanne Petryk, Sarah Adel Bargal, and Joseph E Gonzalez, *Nbdt: neural-backed decision trees*, arXiv preprint arXiv:2004.00221 (2020).

[Web06] Geoffrey I Webb, *Discovering significant rules*, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 434–443.

[Web07] _____, *Discovering significant patterns*, Machine learning **68** (2007), no. 1, 1–33.

[Yil] Soner Yildirim, *All the way from information theory to log loss in machine learning*, `https://towardsdatascience.com/all-the-way-from-information-theory-to-log-loss-in/machine-learning-c78488dade15`, Accessed: 2021-04-11.

[ZLLG18] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng, *Binary relevance for multi-label learning: an overview*, Frontiers of Computer Science **12** (2018), no. 2, 191–202.

Appendix B

The Questionnaire

1. How likely are you to recommend working in an office like yours to a friend or colleague?
   - 0 - 10 scale, 0 - Not at all, 10 - Definitely
2. And why?
   - open-ended
3. How well are you able to focus/concentrate at work?
   - 0 - 10 scale, 0 - Not at all, 10 - 100 %
4. How satisfied are you with the acoustic conditions at your workplace?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied
5. How satisfied are you with the temperature at your workplace?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied
6. How satisfied are you with the lighting conditions at your workplace?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied
7. How satisfied are you with the air quality at your workplace?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied
8. How satisfied are you with the design of your workplace?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied

9. How well are you sitting?
   - 0 - 10 scale, 0 - Terribly, 10 - Great
10. Are you satisfied with the tidiness?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied
11. Do you feel like you have enough opportunities to restore energy during your workdays?
   - 0 - 10 scale, 0 - Not at all, 10 - Definitely
12. How satisfied are you with the coffee and snacks at your workplace?
   - 0 - 10 scale, 0 - Not at all, 10 - Very satisfied
13. Do you agree with the decisions this company makes?
   - 0 - 10 scale, 0 - Not at all, 10 - 100 %
14. Do you feel like the meetings you attend are valuable for the work you are doing?
   - 0 - 10 scale, 0 - Not at all, 10 - Definitely
15. What is the most important problem in your workplace that should be fixed right now?
   - open-ended
16. If you could wish for three changes in your work environment, what would they be?
   - open-ended
17. Are you a male or a female?
   - options: male, female
18. How old are you?
   - options: 18-25, 26-35, 36-45, 45-60, 60+

Note: If the user's answer to questions 3 - 14 had a low rating/negative sentiment (value ≤ 2 on scale 1 - 10) , then the respondent was given a follow up open-ended question "How could we make it better?"

Appendix C

Examples of Most Common Misclassifications

There are six kinds of mistakes our classifier makes. First one is not actually the classifier's error, but simply a human error - mislabeled data. Meaning the person who labeled the data did not indicate that the text contains the topic temperature, but it in fact does, and our classifier labeled the data correctly and vice versa. Then there are cases in which the text was correctly labeled as "contains the topic temperature", but the classifier labeled it as "does not contain the topic temperature" and vice versa. Next reason for misclassification is losing the key words due to translation from colloquial czech (e.g. "it is vydejcháno", "I can work in separate místnůstkách", "there are still some drills, rambajs"). The last one is that identifying some topics can be hard, because they do not have common key words (e.g. culture - "nobody appreciates me", "Anna is always annoying",...).

Examples where a person mislabeled the text as "does not contain the topic temperature", while the classifier correctly labeled the text as "contains the topic temperature".

- text: Bc of air condition and impossibility to open windows. Open space may be loud and air condition sometimes is too windy... Windy cold air condition from the ceiling All tables to be adjustable, better air condition, lemon for tea in the kitchen :)
  - predicted: {temperature, air, acoustics, ergonomics, coffee_snacks}
  - ground truth: {air, acoustics, ergonomics, coffee_snacks}

- text: Lack of light, bad air, a lot of people in small spaces (in the afternoon, the air is stale quite unnatural temperature (cold in summer). air (possibility of opening windows)
  - predicted: {temperature, air, light, facilities}
  - ground truth: {air, light, facilities}
- text: Noise, hot, stale air. Fresh air
  - predicted: {temperature, air, acoustics}
  - ground truth: {air, acoustics}

Examples where a person mislabeled the text as "contain the topic temperature", while the classifier correctly labeled the text as "does not contain the topic temperature".

- text: noisy areas, lots of noisy people around me, not respect others
  - predicted: {}
  - ground truth: {temperature, acoustics, culture}
- text: A little too noisy. Unhealthy air conditioning, more quiet spaces, standing tables
  - predicted: {air, acoustics}
  - ground truth: {temperature, air, acoustics, ergonomics}

Examples where a person correctly labeled the text as "contains the topic temperature", but the classifier mislabeled the text as "does not contain the topic temperature".

- text: Open spaces are often distracting. There are no places for napping. Warmer mornings
  - predicted: {focus, relax}
  - ground truth: {temperature, focus, relax}
- text: open space, beeping every time to go to the kitchen for a water, etc, strong sun shining against the eyes, shades not helping, high temperatures in the morning while sun is shining...
  - predicted: {light}

  - ground truth: {temperature, facilities}

Examples where a person correctly labeled the text as “does not contain the topic temperature”, but the classifier mislabeled the text as “contains the topic temperature”.

- text: air quality, to be able to ventilate (in winter) better air
  - predicted: {temperature, air}
  - ground truth: {air}
- text: The working place is comfortable but there could be more options to stretch out or warm up after a long sitting
  - predicted: {temperature}
  - ground truth: {relax}