

Automated detection of failures based on service records

by

Andrea Boroni Grazioli

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday September 18, 2023 at 15:30.

Thesis committee:

Chair:	Prof. dr. ir. G. Jongbloed
Member:	Dr. F. Yu
Supervisor TU Delft:	Dr. P. Chen
Supervisor Tesla:	J. Johansen

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

This thesis is confidential and cannot be made public until September 11, 2025

Abstract

With the ever-increasing need to reduce the use of fossil fuels, Tesla is accelerating the world's transition to sustainable energy. This means replacing all internal combustion vehicles with electric ones over time. The growing number of Tesla vehicles on the road poses interesting scaling challenges for all departments especially for the Service Engineering team. To help prioritize issues and reduce service costs, frequencies, and duration, a way to generate an overview of all the costs separated by types of repair is needed. This thesis aims to automatically generate such an overview by borrowing techniques used in Natural Language Processing.

In particular, the LDA and GSDMM algorithms for topic generation are tested. Additionally, a novel method based on the cross frequencies of items is presented. Methods are also presented to compute the novelty of every topic and a new metric, here called the growing score, is introduced as a mean to monitor the frequencies of the topics over time.

The methods are applied to service records data from Tesla and the results are analyzed in detail. The results are also enhanced by computing additional information regarding costs for all parts of the service visit and also the configurations of the cars. It is found that the novel approach for the topic generation and the novelty and growing scores produce useful information that can be used to optimize and avoid the need for service procedures to reducing total cost of ownership.

Keywords — LDA, GSDMM, topic generation, trends analysis, novelty scores, growing scores, Tesla

Preface

TU Delft has given me more than I could have hoped for. The path was not always laid out and I took a couple of detours along the way, but I reached my destination. Let this be the end and a new beginning.

I would like to thank all the professors of the Faculty of Applied Mathematics for their great teachings. In particular, Dr. Piao Chen for supervising my thesis project, Prof. Geurt Jongbloed for heading the Thesis Committee and MSc Wenda Kang for meeting with me weekly and providing great feedback and support.

A big thank you to all the people at Tesla, in particular to Joachim Johansen for being a great manager and always having the right advice, Alex Badowski for helping me through the journey and all the colleagues I had the pleasure to interact with.

The most sincere appreciation goes to my parents, Michela and Ermete. They have always supported me, even from thousands of kilometers away. I truly couldn't have asked for better. Thank you to my grandparents, Anna, Achille, Giaele and Serapione, for being so kind and generous. Thank you to my brother, Matteo, for always being curious. And thanks to Emma, for everything.

Finally, to everyone I interacted with during my master degree, my housemates in DS4, my course mates at TU Delft and PoliMi, my teammates at DUT, thank you for all the experiences we shared together and for making me who I am today. I am privileged to call some of you my friends. There are truly too many to list.

"I cannot remember the books I've read any more than the meals I have eaten; even so, they have made me."

-Ralph Waldo Emerson

Andrea Boroni Grazioli
Delft, September 2023

Contents

Abstract	i
Preface	ii
1 Introduction	1
1.1 Tesla and Service Engineering	1
1.2 Data at Tesla	2
1.3 Project Goal and Motivation	3
1.4 Tesla Definitions	4
1.5 On Censoring	5
2 Methods	7
2.1 General Method	7
2.2 Topic Generation	8
2.2.1 Latent Dirichlet Allocation	9
2.2.2 Gibbs Sampling for Dirichlet Multinomial Mixture Model	13
2.2.3 Combine and Split Algorithm	15
2.2.4 Correction Based Algorithm	18
2.2.5 Topics Generation Algorithms Conclusions	20
2.3 Novelty Scores	20
2.4 Growing Scores	23
2.4.1 Max Slope	24
2.5 Computing Cost Metrics	28
2.5.1 FRT Cost	29
2.5.2 Parts Cost	29
2.5.3 Days in Service Cost	29
2.6 Car Configuration analysis	29
2.7 Software Used	30
2.7.1 Tableau	30
2.7.2 SQL	31
2.7.3 Python	31
3 Topic Generation Results	34
3.1 Latent Dirichlet Allocation Results	34
3.1.1 Data Pre-processing	34
3.1.2 Topics Generated	37
3.1.3 Conclusions	41
3.2 GSDMM Results	43
3.2.1 Conclusions	45
3.3 Combine and Split Results	46
3.3.1 Conclusions	47
3.4 Correction Based Algorithm Results	48
3.4.1 Conclusions	51

4 Trend Analysis Results	52
4.1 Results for one month of data	52
4.2 Results for 1 year of data	53
4.2.1 Conclusions	56
5 Conclusion	59
5.1 Discussion	63
References	65

1

Introduction

1.1. Tesla and Service Engineering

The mission of Tesla is to accelerate the world's transition to sustainable energy and to achieve that, the goal is to produce and sell 20 million vehicles per year and reach level 5 autonomy. Tesla is already the biggest EV car manufacturer in the world, with 1.3 million cars sold in 2022 [1]. The projection for the sales in the year 2023 is set to surpass 2022 as it has done every year since the beginning of the company, continuing to get closer to the 20 million goal.

Tesla has been focusing on cheaper mass-produced vehicles such as the Model 3 and the Model Y, all models can be seen in Figure 1.1. Increasing the production, however, has been mainly blocked by the difficulties of scaling the manufacturing. Tesla has currently 4 factories around the world (Fremont, Texas, Shanghai, and Berlin [2]) that produce EVs each specialized in different sets of models. More factories are set to open shortly to increase the production numbers. However, every factory is constantly pushing to increase its own production capabilities, as evidenced by both Giga Texas [3] and Giga Berlin [4] hitting a record high of 5 thousand cars produced in a single week recently.

Tesla runs repair workshops for servicing Tesla vehicles. When a vehicle visits the Service Center, a data record is generated that includes the symptom of the customer concern, the repair action that was taken and which part was replaced. The task of analyzing this data and optimizing the service procedures is the job of the Service Engineering team.

Therefore, the Service Engineering team has to link a lot of different departments: the design team, the production team, the field quality, and customer service, just to name a few. Additionally, the Service Engineers need to have very high knowledge of the vehicles and all the possible failure modes associated with every subsystem, so that they can assist and react quickly to every possible scenario. Thanks to the Service Engineering team many improvements are made continually in both the design and production of the vehicles as well as Over-the-air (OTA) software updates.

Due to the skill set required, there is a need for specialization and subsystem experts among the Service Engineers. The team is thus split into sub-departments that are each responsible for different subsystems: Powertrain, Infotainment, Chassis & LV, etc. Additionally, to improve the responsiveness, there is one of each team per region: North America (NA); Europe, the Middle East, Africa (EMEA); and Asia-Pacific (APAC). Between January and September 2023, I worked for Tesla as an Intern in the EMEA Powertrain Service Engineering Team.

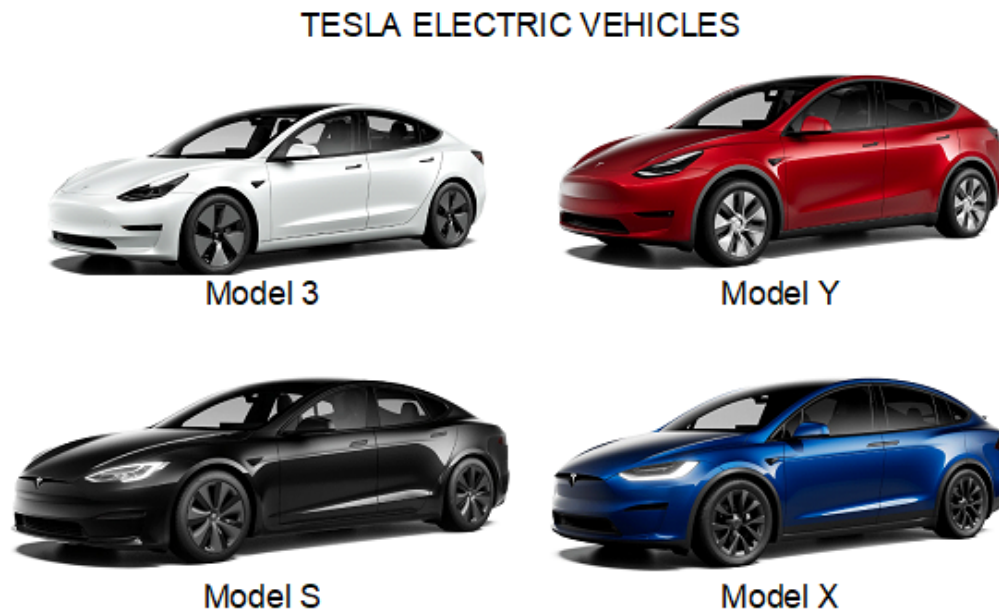


Figure 1.1: Tesla models available as of 2023

1.2. Data at Tesla

Tesla is on the cutting edge of data acquisition. Every car has hundreds of sensors monitoring all and every aspect of the car's functionality. The sensor values are read and used by the controllers in the car, but some of that data is anonymized and sent to Tesla for the purpose of product improvement.

Due to data storage constraints, not all the data is retained, it would be too much data even for a relatively "small" fleet of around 4 million cars. So the cars send data only during specific events, such as during detected failures of components or after a targeted request from an engineer or a technician in order to diagnose and repair a vehicle on request by the vehicle owner.

During a service visit, the technicians record information about actions they perform:

- **Symptoms:** what did the customer experience that led to a request for a service appointment;
- **Parts replaced:** which parts were replaced during the visit. Every part is registered by its unique Part Number;
- **Correction Codes:** which actions were performed during the visit. All actions have codes associated with them to identify exactly what kind of work was performed;
- **Flat Rate Time (FRT):** "hands-on" time the technicians spent diagnosing or repairing the car;
- **Days in Service:** the total amount of days the car has spent in the Service Center;
- **Pay type:** were the parts replaced in warranty or not, and who ended up paying for the visit.

There are other data points recorded for every visit, but the ones mentioned above are the most interesting and the ones used throughout this project. Every Service Visit is split into several **Repair Order Jobs**, for

cfg_country	use_kgc	cfg_rear_drive_unit	cfg_car	cfg_packenergy
DE	Cfb	PM216MOSFET	3	SR
GB	Cfb	PM216MOSFET	3	LR
US	Cfa	PM216MOSFET	Y	LR
CN	Cfa	Small	X	None
CN	Cwa	Small	X	None
CN	Cfa	Small	X	None
SE	Dfb	PM216MOSFET	3	LR
CN	Cfa	Small	X	None
CN	Cfa	Small	X	None
CN	Cfa	Small	X	None

Figure 1.2: Example of a few configurations for some cars

which data is collected separately. This is very useful since, most of the time, separate Jobs in the same Service Visits have to do with unrelated repairs.

Finally, every vehicle has a set of configuration options that are logged and stored in specific databases. Some configurations are constant throughout the lifetime of the car, such as model (S, X, 3, or Y) and factory (Berlin, Fremont, Shanghai). Other configurations are subject to change but remain constant the majority of the time, such as type of LV and HV battery or country of use. Some examples of configurations can be found in [Figure 1.2](#).

1.3. Project Goal and Motivation

Due to the current increasing rate of vehicle production, the focus of Tesla as a whole has been set on cost reduction, continuous demand generation, and value creation. As for the cost reduction part, Service costs are a contributor to the total vehicle cost, both to Tesla and the customer. Therefore, the Service Engineering team has a big role to play in this area.

There are many ways to reduce Service costs, for example: reduce failure rates, reduce the cost of the parts replaced, reduce diagnostic and repair time, and identify new issues earlier to push fixes to the fleet. Some efforts aimed at reducing cost have been discussed in the previous section.

For more efforts to be effective, it is necessary to have an overview of the issues contributing most to diagnostic time, repair time, and cost that are present in Service. The high amount of data makes it challenging to create such an overview. An additional challenge is posed by the fact that the data required for such an analysis is scattered throughout many different data sources that need to be combined to create the full picture.

Moreover, we would like a way to determine trending new and early life issues, so that the engineering teams can tackle them in time and resolve the core root cause to stop the spread to more vehicles.

This leads to the following Project Goals:

1. **Create an actionable list of the top cost impacting items;**
2. **Create a system to identify new and growing issues.**



Figure 1.3: Tesla Service Center

Figure 1.4 shows the project structure and the flow of information. The data from the repair orders is collected and, using techniques from Natural Language Processing analyzed to produce intermediate results called topics that we associate with individual repairs. For every topic, metrics can be computed, such as novelty and costs, and metadata can be added to enhance the topics' definitions.

The combination of all the data results in a list of topics that can be sorted by many metrics, depending on the interests of the user. The additional metadata and root cause data indicate the actions that can be taken to reduce the impact of the specific issue. The main contribution of the author is the approach to the analysis of the data and the aggregation of the results in an interactive display.

The methods used to generate the results are explained in detail in Section 2. The results of the topic generation algorithms are presented in Section 3, while the results for the trends analysis are reported in Section 4. Finally, Section 5 shows the interactive dashboards created and draws conclusions about the project.

1.4. Tesla Definitions

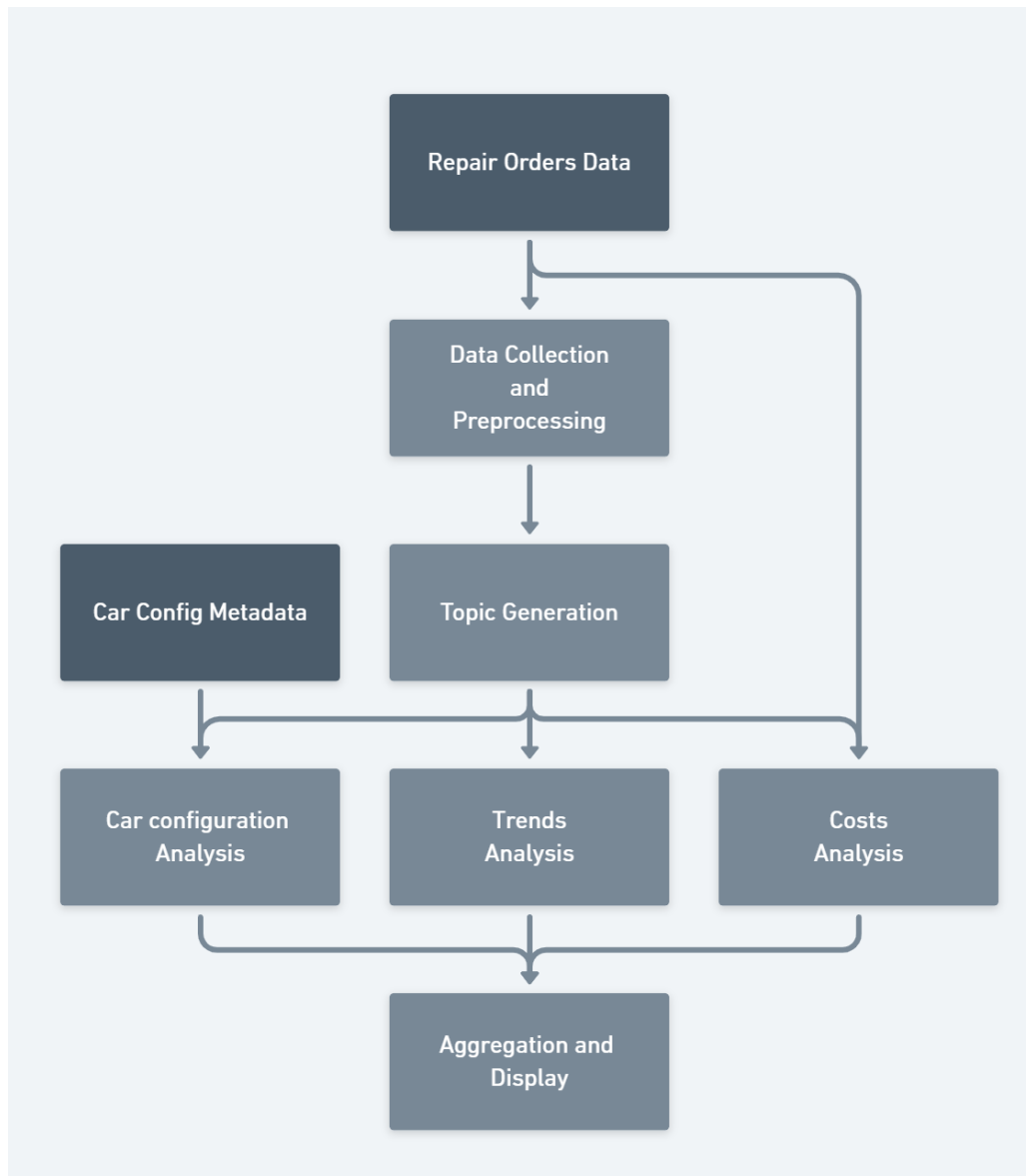
At Tesla, there is specific terminology that is used on a day-to-day basis. Here is a list, with definitions, of the terms that are most needed to understand this document:

- **Service Center:** Site where Tesla cars can go to get repaired, see Figure 1.3.
- **Service Technician:** A person who works in a Service Center repairing cars.
- **Service Engineering:** Group of Tesla employees that focuses on improving the Service process.
- **Article:** An article is a webpage describing a specific known failure. The article contains steps to diagnose and repair the car affected by the issue. Every article is identified by a unique **article ID**.
- **Vin:** Vehicle Identification Number, also used as a synonym for car.
- **Part Number:** String of letters and numbers that identifies a specific part of the car. Every part has a unique part number.
- **Correction Code:** Code that identifies an action performed during a service visit.
- **Symptom:** General issue that the car shows before the service visit according to the customer, selected from a broad, but fixed, set of options.
- **Service Visit:** The act of a single car going to a service center for any reason.

- **Repair Order:** Formal name for a Service Visit. Every Repair Order is registered and gets a Repair Order Number that uniquely identifies it.
- **Repair Order Job:** For every Repair Order, the technician will instantiate different Job lines that focus on different repairs. Thus every Repair Order has one or more Repair Order Jobs associated with it. Every Repair Order Job is identified by a unique number.
- **Service Records:** Databases that store data regarding Service Visits. This includes all parts, correction codes, and symptoms.
- **Car Config:** Parameters that are specific to a car. There are hundreds of different config parameters, some examples are model (*S, X, Y, 3*) or build factory (*Berlin, Fremont, Austin*). This data is stored in Tesla databases.
- **LV:** Low Voltage.
- **HV:** High Voltage.
- **EV:** Electric Vehicle.
- **SoC:** State of Charge (of the battery).
- **FRT:** Flat Rate Time

1.5. On Censoring

Sensitive data has been omitted from this report. The focus of this report is the methods and the process as opposed to the exact results. The majority of the precise costs, frequencies, and part numbers are not mentioned, preferring percentages or qualitative statements instead. The censoring of this data does not compromise the quality of the methods or the processes used for this project. This report will also be under embargo for a duration of 2 years.

**Figure 1.4:** Project Flowchart

2

Methods

This chapter describes the methods used during the project. Section 2.1 introduces the overall ideas of the methods with an introduction to previous research. In Section 2.2 multiple methods for topic generation are presented and explained in detail. The LDA algorithm is described in Section 2.2.1, the GSDMM algorithm in Section 2.2.2 and two algorithms designed by the author in Section 2.2.3 and 2.2.4. Then, in Section 2.3 and 2.4, the formulas for computing novelty and growing scores respectively are presented. Finally, in Section 2.5 and 2.6 additional metrics and analyses are proposed that are specific to our application.

2.1. General Method

The analysis of trends has been a very hot topic in the last decade and a half thanks mostly to the explosion in data availability made possible by the popularity of microblogging platforms such as X (before Twitter), Facebook, Tumblr, and many others. The scale and variety of posts published on these platforms created huge opportunities for people to gain valuable information from data mining posts. Twitter, for example, was made popular in part thanks to its evaluation and exposure of trending topics and hashtags. In this thesis, the work done by researchers in the context of microblogs and Natural Language Processing is leveraged and applied to the Tesla problem domain by transferring concepts over to the context of Service visits.

In NLP, the smallest unit of information is a **word**¹. While sometimes single words can be a major contributor to a popular trend, for example the name of a famous person, it is far more common that a set of words all concerning the same topic rises in popularity with the trend. This idea is reflected in the literature [5] [6] [7], where most often the analysis is split into two main steps: first group words into topics, then compute the trends for the topics. This helps reduce the dimensions of the problem since the number of unique topics is usually much smaller than the number of unique words. The thesis, and the hope, is that the topics generated by these methods in the Service Visit domain are related to single issues or repairs.

Here are some definitions for the most common parameters used in the following sections:

- D is the number of documents, the group of all **documents** is called the **corpus**;
- V is the number of unique words present in the documents, the group of unique words is called the **vocabulary**;
- K is the number of **topics**.

¹Technically words can be broken down in to letter, but letters don't have semantic meaning by themselves

In Table 2.1, the concepts in the NLP domain compared to their counterparts in the Service Visit context.

NLP concept	Tesla equivalent
Document	Repair Order Job
Corpus	Collection of Repair Order Jobs
Word	Item (Parts replaced, Symptom, Correction code applied)
Vocabulary	Unique items
Topic	Repair or issue

Table 2.1: NLP concept in the Tesla domain

In [6] and [7] methods are proposed for the monitoring of trends on microblogs. In [6] Online-LDA (OLDA) is proposed, an algorithm that "automatically captures the thematic patterns and identifies emerging topics of text streams and their changes over time". In [7] an improved version of the OLDA algorithm is proposed, where the vocabulary and model size are constrained to not grow with time. To gather results about the trends of the topics over time the outputs of the algorithms are analyzed and conclusions about the frequencies are drawn.

Both of the approaches mentioned above are reliant on the Latent Dirichlet Allocation (LDA) algorithm for topic generation, or slight modification of it. There are, however, some disadvantages in using the LDA algorithm in our context, as is shown in Section 3.1.3. The main issue is that LDA is designed to work on long-form text where the number of words in a document is high, while service visits may only have a handful of items associated with each of them. The reliance of the methods proposed in [6] and [7] on LDA makes it hard to change the topic generation algorithm for a better suited one in our context. So other methods need to be researched.

Methods proposed elsewhere make use of different techniques for evaluating the trends over time. For example other methods for evaluation of emerging features can be used such as: chi-square testing for term's foreground and background distributions [8], exponentially weighted average (EWMA) of terms, and co-occur terms [9] and high utility pattern mining [10]. Then other clustering or classification algorithms are used such as modularity-based partitioning, kNN, or SVM to gather emerging features into emerging topics.

Finally, the method proposed in [5] is a more flexible alternative to [6] and [7], and allows for any topic generation algorithm, this creates possibilities to use alternative methods to LDA that might be more suitable. The basic idea of the method is to compute a **novelty score** for each word and separately compute the topics. Then, in a final pass, the scores and the topics are combined to generate a novelty score for each topic. An overview of the information flow described above can be found in Figure 2.1.

2.2. Topic Generation

There are many types of topic generation algorithms, each with its strengths and weaknesses. Examples are Latent Dirichlet Allocation (LDA) [11], Gibbs Sampling for the Dirichlet Multivariate Mixture (GSDMM) [12], Hierarchical Latent Tree Analysis (HLTA) [13], and Probabilistic Latent Semantic Indexing (pLSI) [14]. In this section, the algorithms used in this project are presented, together with their differences and their preferred domain of application. There are, however, two assumptions that are common to all the algorithms: the exchangeability of words in a document and the exchangeability of documents [15]. This means that the order of words in a document can be neglected as well as the order of the documents themselves. Both of these assumptions are meant to simplify the inference without losing much information. In the case of the documents, it is pretty intuitive to justify their exchangeability in most contexts where the timespan between the first and last document is relatively low and the generation of documents is independent. The order of the

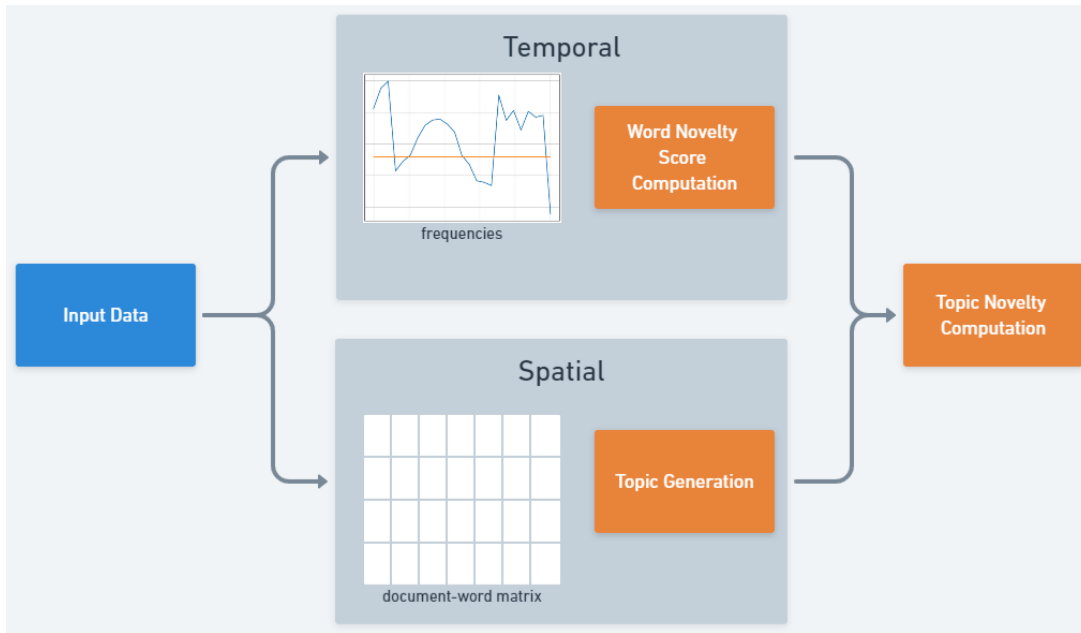


Figure 2.1: Trend analysis overview

words, on the other hand, conveys much more meaning that is lost by considering them exchangeable. In our application, however, there is no intrinsic order in the items and thus this assumption is perfectly reasonable.

This leads to the inputs being simplified to a matrix B of size $D \times V$, where the value of B_{dw} is equal to the number of times that the word w occurs in document d .

The LDA algorithm is described first in Section 2.2.1 as it is the most commonly used and the one suggested in [5]. The outputs of the LDA algorithm are taken as the blueprint for all the others and in places where the outputs don't match exactly, some effort is made to do so. Then the GSDMM algorithm is analyzed in detail in Section 2.2.2 as it has nice properties when working in a short-text domain, such as our own. Section 3 will show that the quality of the topics generated by both LDA and GSDMM could be improved, and thus two more algorithms designed by the author are presented. The first, deemed the Combine and Split algorithm of Section 2.2.3, aims at extracting topics by using simply the cross frequencies of the items, while the second, deemed the Correction Based algorithm of Section 2.2.4, utilizes more domain knowledge by treating correction codes differently from other items.

2.2.1. Latent Dirichlet Allocation

LDA was first proposed by Pritchard et al. [16] in 2000 and then later applied in the context of machine learning in 2003 by Blei et al. [11], since then it has been a staple of Natural Language Processing. To understand the LDA model, it is useful to first explain how the model assumes the documents are generated.

Useful Distributions In the course of this section, a few probability distributions will be used, and here we give the formal definition of them.

The **Poisson** distribution is a discrete distribution on the positive integers [17], first introduced by S. Poisson himself in [18]. The Poisson distribution has one parameter, usually called λ , that describes its shape and the

probability mass function is defined in Eq. (2.1):

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad (2.1)$$

The **Dirichlet** distribution is a continuous multivariate distribution parameterized by a vector of positive reals [19], usually called α . The Dirichlet distribution is a multivariate generalization of the beta distribution [20]. The probability density function can be seen in Eq. (2.2):

$$f(\mathbf{x}, \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^K x_i^{\alpha_i - 1}, \quad (2.2)$$

where $B(\alpha) = \prod_{i=1}^K \Gamma(\alpha_i) / \Gamma(\alpha_0)$.

Finally, the **Multinomial** distribution is the generalization of the **Binomial** distribution to a multivariate output set [21]. For n independent trials and $p_1 \dots p_k$ event probabilities, the probability mass function is given in Eq. (2.3):

$$p(\mathbf{x}) = \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x_k}. \quad (2.3)$$

The Model In the LDA model, two latent variables describe how to generate a document. The first variable, α , describes the probabilities for each topic to appear in a document. The second variable, β , describes the probabilities for each word to appear in a document, given that the document contains a specific topic. The document generation starts by sampling the number of words that will make up the document, N . N is typically distributed like a Poisson distribution with parameter η . Then γ is sampled from a Dirichlet distribution with parameter α . The value of γ represents the distribution of the topics for the document. Then for each of the N words in the document, a topic z_n is chosen from a Multinomial distribution with parameter γ . Finally, the word w_n is chosen from a Multinomial probability of β conditioned on the topic chosen z_n . The algorithmic form of the generative process can be seen in Algorithm 1.

Algorithm 1: LDA document generation algorithm

- (1) Choose $N \sim \text{Poisson}(\eta)$.
 - (2) Choose $\gamma \sim \text{Dir}(\alpha)$.
 - (3) For each of the N words w_n :
 - (a) Choose a topic $z_n \sim \text{Multinomial}(\gamma)$.
 - (b) Choose a word w_n from $p(w_n | z_n, \beta)$.
-

Note that the Poisson assumption is not critical to any step that follows step 1 in Algorithm 1 and other distributions can be used as necessary, additionally, N is independent of the other generated variables (γ and z). It is thus an ancillary variable and its effect will be ignored in the following analysis.

Now that we have a general idea of how a document is constructed we can give proper mathematical definitions:

- N_i is the number of words in document i ;
- α is a K -dimensional vector, the parameter of the Dirichlet prior of the per-document topic distribution;
- γ_i is a K -dimensional vector, topic distribution for document i sampled from $\text{Dir}(\alpha)$;

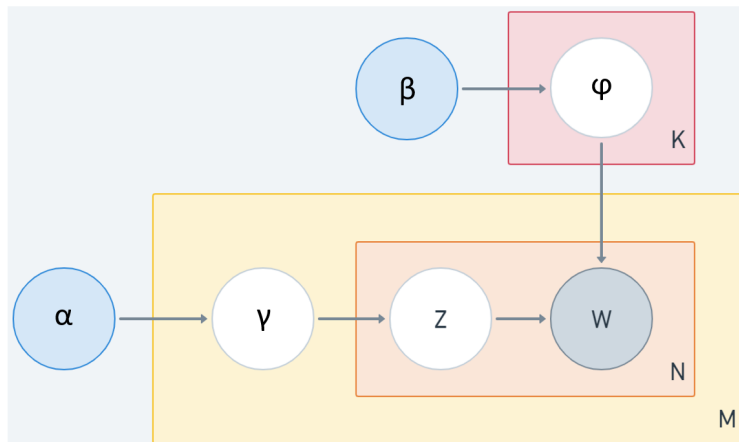


Figure 2.2: Plate Notation for the full LDA model

- β is a $K \times V$ matrix, where every row is the parameter of the Dirichlet prior of the per-topic word distribution;
- φ_k is a V -dimensional vector, word distribution for topic k sampled from $Dir(\beta_k)$;
- z_{ij} is a value between 1 and K , the topic for the j -th word in document i sampled from $Multinomial(\gamma_i)$;
- w_{ij} is a value between 1 and V , specific word in position j for document i sampled from $Multinomial(\varphi_{z_{ij}})$.

The LDA generation model can be represented in plate notation as in Figure 2.2. Instead of drawing each repeated variable individually, a plate or rectangle is used to group variables into a subgraph that repeats together, and a number is drawn in the bottom right of the plate to represent the number of repetitions of the subgraph in the plate. The assumptions are that the subgraph is duplicated that many times, the variables in the subgraph are indexed by the repetition number, and any links that cross a plate boundary are replicated once for each subgraph repetition. Structures such as the ones shown in Figures 2.2 and 2.3 are referred to as hierarchical models [22] or conditionally independent hierarchical models [23]. Such models are also often referred to as parametric empirical Bayes models, a term that refers not only to the model structure but also to the methods used for estimating parameters in the model [24].

Out of all the variables in Figure 2.2, w is the only variable that is observable from the documents. All other variables are called "hidden" since they are not observable by looking at the document. Additionally, α and β are called "latent" because they are the variables from which all others are generated through the sampling process described above.

Inference and Parameter Estimation The key inferential problem that we need to solve to use LDA is that of computing the posterior distribution of the hidden variables given a document:

$$p(\gamma, z | w, \alpha, \beta) = \frac{p(\gamma, z, w | \alpha, \beta)}{p(w | \alpha, \beta)}. \quad (2.4)$$

Unfortunately, this distribution is intractable to compute in general [25]. There exists, however, a wide variety of approximate inference algorithms. In the following, a convexity-based variational algorithm for inference in LDA based on [11] is described, a similar method is used by the Python package *scikit-learn* [26], used in the project for the calculations. The implementation is based on [27] and [28].

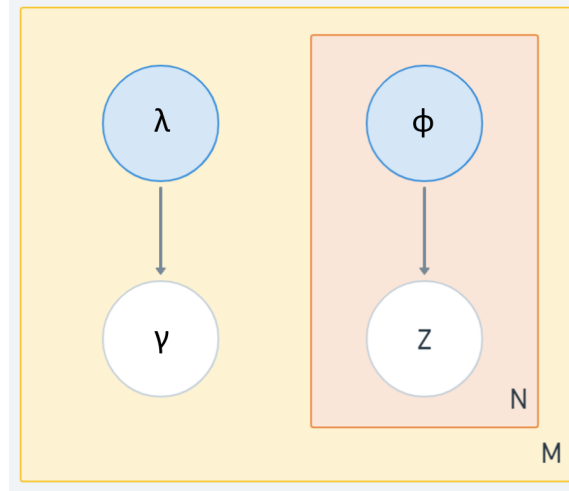


Figure 2.3: Plate Notation for the variational LDA model

The basic idea of this algorithm is to make use of Jensen's inequality to obtain an adjustable lower bound on the log-likelihood [29]. A way of obtaining such a lower bound is to approximate the intractable distributions with variational parameters that are chosen to be restricted to a simpler family of distributions. The variational parameters are then computed by minimizing the difference with the original distributions. In our case, we simplify the LDA model to what is shown in Figure 2.3 and thus add the variational parameters: λ and ϕ , where λ is from a Dirichlet distribution and every (ϕ_1, \dots, ϕ_N) is from a Multinomial distribution.

Finding the values of the variational parameters is equivalent to the following optimization problem:

$$(\lambda^*, \phi^*) = \arg \min_{(\lambda, \phi)} D((q(\gamma, z | \lambda, \phi), p(\gamma, z | w, \alpha, \beta)), \quad (2.5)$$

where $D(\mathbf{X}, \mathbf{Y})$ is the Kullback-Leibler (KL) divergence and it is used to measure the difference between the distributions of γ and z in the model with and without the variational parameters.

As shown in [11], the minimization problem in Eq. (2.5) can be solved by the iterative fixed-point method. Eqs. (2.6) and (2.7) are the update equations for ϕ and λ respectively:

$$\phi_{ni} \propto \beta_{i w_n} \exp\{E_q[\log(\gamma_i) | \lambda]\}, \quad (2.6)$$

$$\lambda_i = \alpha_i + \sum_{n=1}^N \phi_{ni}. \quad (2.7)$$

The summary of the variational inference procedure can be found in Algorithm 2, with appropriate starting points for λ and ϕ . From the pseudocode, it is clear that each iteration of variational inference for LDA requires $O((N + 1)k)$ operations.

The goal of the inference is to find parameters α and β such that the log-likelihood of the data is maximized:

$$l(\alpha, \beta) = \sum_{d=1}^D p(w_d | \alpha, \beta) \quad (2.8)$$

Algorithm 2: LDA variational inference algorithm

```

(1)  $\phi_{ni}^0 \leftarrow 1/k \quad \forall i, n$ 
(2)  $\lambda_i \leftarrow \alpha_i + N/k \quad \forall i$ 
(3) repeat
(4)   for  $n = 1$  to  $N$ 
(5)     for  $i = 1$  to  $k$ 
(6)        $\phi_{ni}^{t+1} \leftarrow \beta_{iw_n} \exp(E_q[\log(\gamma_i)|\lambda])$ 
(7)       normalize  $\phi_{ni}^{t+1}$  to sum to 1
(8)    $\lambda^{t+1} \leftarrow \alpha + \sum_{n=1}^N \phi_n^{t+1}$ 
(9) until convergence

```

This can be done by iteratively computing (λ, ϕ) and (α, β) . We saw how to approximate (λ, ϕ) in Eqs. (2.6) and (2.7), so what is left to show is how to maximize (α, β) based on the values of (λ, ϕ) . This method is called *variational EM* procedure, due to the E(stimation) and M(aximization) steps.

Again, [11] shows that maximizing the log-likelihood in Eq. (2.8) results in Eq. 2.9 used to calculate β :

$$\beta_{ij} \propto \sum_{d=1}^D \sum_{n=1}^{N_d} \phi_{dni} w_{dn}^j, \quad (2.9)$$

while the equation for α does not have a closed-form solution since it is much more complicated. However, a solution can be found by the use of an efficient Newton-Raphson method in which the Hessian is inverted in linear time. This method is often used for maximum likelihood estimation of the Dirichlet distribution as in [30] and [31].

Outputs To conclude this section on LDA, these are the outputs of the algorithm that are of the highest interest to us:

- φ_{kw} : Probability of word w appearing in topic k , the topic-word distribution (approximation of β);
- γ_{kd} : Probability of topic k being present in document d , the document-topic distribution;
- θ_k : Probability of topic k appearing in a document, the topic distribution (approximation of α).

2.2.2. Gibbs Sampling for Dirichlet Multinomial Mixture Model

Gibbs Sampling algorithm for Dirichlet Multinomial Mixture, or GSDMM for short, is a clustering technique that can be used to infer topics from textual data. The algorithm was first introduced in [12]. The LDA algorithm described in Section 2.2.1 is designed to work on long-form documents, while the GSDMM algorithm described in this section is best when used on short texts. This is because the GSDMM algorithm assumes that every document is associated with a single topic. In our application for service visits, the document length is usually pretty short so GSDMM could be the preferred algorithm.

As stated in [32], there are several issues with clustering: 1) Setting of the number of clusters; 2) Ability to work with high-dimensional data; 3) Interpretability of the results; 4) Scalability to large datasets. Short-text clustering has all of the above challenges and has the additional challenge of sparsity, since most, if not all, the words tend to appear only once per document, thus rendering some measures useless. GSDMM attempts to solve them by changing some of the assumptions of other methods.

Movie Group Process First, an explanation of how the GSDMM algorithm works is given so that we have a base for discussing how the algorithm solves the above problems. As explained in the original paper [12], the GSDMM algorithm is equivalent to the Movie Group Process (MGP). MGP is very intuitive to understand and since it is equivalent to GSDMM, MGP is presented first.

We can imagine that the professor of a movie discussion course has a class of students. He plans to divide the students into several groups so that they can have smaller discussions where everyone can participate. He expects the students in the same group to have watched similar movies, such that they would have more movies to discuss. The professor asks the students to write down a list of movies they have watched and it only gives them a short period to finish the task. This is to reduce the number of movies on the list and increase the likelihood of the students writing down movies they have watched recently or movies they particularly like. In both cases, those will be the movies that the students are more interested in discussing. After the students finished writing their lists, each student can be represented by the list of movies they produced. The professor needs to find a way to cluster the students into several groups according to his initial requirements: students in the same group will share similar movie lists, while students in different groups will have different interests.

The professor then invites all students to a big restaurant with K tables and randomly assigns the students to each of these tables. Then he asks the students to choose a different table in turn if the following two rules are satisfied:

Rule 1: The new table has more students.

Rule 2: The students at the new table have watched more similar movies.

As this process goes on, students will continue to move to tables according to the above rules. Thus some tables will grow larger and others will become empty. When the process stops (either because the professors decides the groups are good enough or because there are no more students that can move), we can expect that only some tables will be populated and the students in each table will share similar interests.

We can see that the above two natural rules are related to the two goals of clustering: Completeness and Homogeneity [33]. Completeness represents the objective that all members of a ground true group are assigned to the same cluster. Rule 1 of MGP tends to result in high completeness, as it leads popular tables to be more popular, and students in the same ground true group are more likely to be in the same table. Homogeneity represents the objective that each cluster contains only members of a single ground true group. Rule 2 of MGP tends to result in high homogeneity because it leads the students in the same table to be more similar and thus more likely to be in the same ground true group.

To put it in a formal context with matching terms used in Section 2.2.1, the singular students represent **documents** and the entirety of them form the **corpus**, so D is the number of students. Every unique movie represents a **word** in the **vocabulary** and the total number of unique movies is V . N is then the number of movies in a specific student's list. The sparse characteristic of short text means that V is really large (often larger than 10^4), while the average number of words (\bar{N}) in each short text is small (often less than 10^2). For GSDMM K is not the number of clusters (or topics), but an upper bound on the number of clusters. This is because some of the initial tables can become empty by the end of the process and thus will not count as clusters in the end.

As mentioned at the beginning of this section, GSDMM attempts to solve the main clustering problems in the context of short-text datasets. Here are some properties of GSDMM that contribute to the attempt:

1. GSDMM can infer the number of clusters. The initial number of clusters, K , is an input to the algorithm, but it is only an upper bound leaving the model free of selecting the actual number of output clusters.
2. GSDMM can easily cope with sparse and high-dimensional data. Common similarity-based models like K-means [34] for text clustering usually represent the documents with the Vector Space Model (VSM)

[35]. Each **document** is represented with a vector of length V , where each element is the weight of the corresponding **word**. This way of storing data wastes a massive amount of memory due to the sparsity of short-text documents. GSDMM solves this problem by working directly on lists of **words** thus storing a minimal amount of data.

3. GSDMM has fast convergence. For every iteration, the GSDMM algorithm has a time complexity of $O(KD\bar{N})$, whereas other clustering algorithms such as K-means have a time complexity of $O(KDV)$ per iteration. This difference is much more pronounced when the algorithms are applied to short-text clustering since the value of \bar{N} is orders of magnitude less than the value of V .

Mathematical Model With the idea of the MGP in mind, let's introduce now the Dirichlet Multinomial Mixture (DMM) model used first in [36]. The model makes two assumptions about the generative process: (1) the documents are generated by a mixture model [37], and (2) there is a one-to-one correspondence between mixture components and clusters. When generating document d , DMM first selects a mixture component (cluster) k according to the mixture weights (weights of clusters), $p(z = k)$. Then document d is generated by the selected mixture component (cluster) from distribution $p(d|z = k)$. The likelihood of a document d can be characterized by the sum of the total probability over the mixture components:

$$p(d) = \sum_{k=1}^K p(d|z = k)p(z = k). \quad (2.10)$$

With the assumption of exchangeability, we can define $p(d|z = k)$ as:

$$p(d|z = k) = \prod_{w \in d} p(w|z = k). \quad (2.11)$$

The assumption made in [36] is that each cluster is a multinomial distribution over words, such that $p(w|z = k) = p(w|z = k, \phi) = \phi_{kw}$, where $w = 1, \dots, V$ and $\sum_w \phi_{kw} = 1$. They assume a Dirichlet distribution as the prior for each cluster and that the weight of each cluster is sampled from a multinomial distribution. finally, they assume a Dirichlet prior for this multinomial distribution.

The graphical model of DMM is shown in Figure 2.4. In this short text clustering problem, we need to estimate the mixture component (cluster) z for each document d .

Algorithm 3 describes the collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model, which is equivalent to the Movie Group Process, and the meaning of its variables is shown in Table 2.2.

In the initialization step, the documents are randomly assigned to K clusters. The following information is recorded: \mathbf{z} (cluster labels of each document), m_z (number of documents in cluster z), n_z (number of words in cluster z), and n_z^w (number of occurrences of word w in cluster z). Then the documents are traversed for I iterations. In each iteration, the documents are in turn re-assigned to a cluster according to the conditional distribution: $p(z_d = z|\mathbf{z}, \mathbf{d})$. Each time a cluster z is re-assigned to document d , the corresponding information in \mathbf{z} , m_z , n_z , and n_z^w are updated accordingly. GSDMM is also a soft clustering model like the Gaussian Mixture Model (GMM) [38] since we can get the probability of each document belonging to each cluster from $p(z_d = z|\mathbf{z}, \mathbf{d})$.

2.2.3. Combine and Split Algorithm

It is reasonable to expect that if item A is always present with item B, then they belong to the same topic. The simple algorithm presented in this section attempts to create topics based on this intuitive rule.

The algorithm is based on the computation of the **cross-frequency matrix**, C . C is a $V \times V$ matrix where C_{ij} corresponds to the probability of item j to be present in documents where item i is present, thus making every

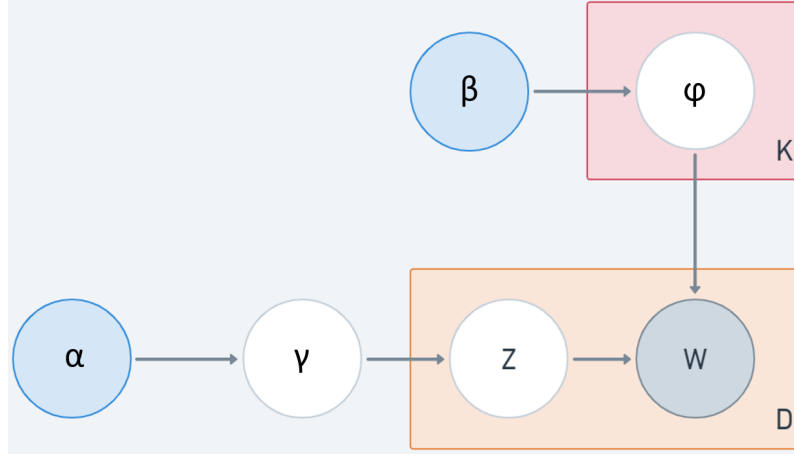


Figure 2.4: Plate notation for the DMM model

Algorithm 3: GSDMM**Data:** Documents in the input, \mathbf{d} .**Result:** Cluster labels for each document, \mathbf{z} .initialize m_z , n_z and n_z^w as zero for each cluster z **for** each document $d \in [1, D]$ **do**sample a cluster for d : $z_d \leftarrow z \sim \text{Multinomial}(1/K)$ $m_z \leftarrow m_z + 1$ $n_z \leftarrow n_z + N_d$ **for** each word $w \in d$ **do** $n_z^w \leftarrow n_z^w + N_d^w$ **for** $i \in [1, I]$ **do****for** each document $d \in [1, D]$ **do**record the current cluster d : $z = z_d$. $m_z \leftarrow m_z - 1$ $n_z \leftarrow n_z - N_d$ **for** each word $w \in d$ **do** $n_z^w \leftarrow n_z^w - N_d^w$ sample a cluster for d : $z_d \leftarrow z \sim p(z_d = z | \mathbf{z}_d, \mathbf{d})$ $m_z \leftarrow m_z + 1$ $n_z \leftarrow n_z + N_d$ **for** each word $w \in d$ **do** $n_z^w \leftarrow n_z^w + N_d^w$

V	number of words in the vocabulary
D	number of documents in the corpus
\mathbf{d}	documents in the corpus
\mathbf{z}	cluster labels of each document
I	number of iterations
m_z	number of documents in cluster z
n_z	number of words in cluster z
n_z^w	number of occurrences of word w in cluster z
N_d	number of words in document d
N_d^w	number of occurrences of word w in document d

Table 2.2: GSDMM notations

C_{ij} a number between 0 and 1. In simple examples and ignoring very low values, the cross-frequency matrix can be visualized in a graphical way as in Figure 2.5, where every node represents an item and every edge from node i to node j represents the value of C_{ij} .

In the case where item A is (almost) always present with item B, item B is not necessarily always present with item A. Therefore the algorithm goes through the following rules iteratively until it cannot make progress anymore while recomputing the cross-frequency matrix at every iteration to accommodate the changes:

1. Rule 1: Combine items A and B if both C_{AB} and C_{BA} are higher than a threshold;
2. Rule 2: Split item A if there is at least one item B for which C_{BA} is higher than a threshold;
3. Rule 3: Remove item A if it has no occurrences (this sometimes happens when splitting an item from Rule 2).

When item A and item B are combined, then they become item A + B, and their occurrences are combined. When splitting item A against item B, item A is deprecated in favor of two new items: A (B) and A - (B). Every occurrence of the original item A is transferred to either item A (B), if item B appears in the same document, or item A - (B), if item B does not appear in the same document. At the end of the process, the cross-frequency matrix will hold the information regarding the topics and can be used to compute all the outputs as for the LDA algorithm.

Let's demonstrate how the algorithm works with a simple example with a vocabulary size of $V = 10$. Some ground true topics are chosen such that they can be sampled to generate the input data. Every topic is represented by a list of items, in this case, every item is given a unique number between 0 and 9. The topics picked for this example are shown in Table 2.3.

Topic	Elements	Frequency
1	1, 2, 3	0.1
2	1, 2, 4, 5	0.1
3	6	0.1
4	7, 8, 9	0.4
5	5, 7	0.05
6	0, 8	0.25

Table 2.3: Combine and Split algorithm example topics

A **corpus** of $D = 10000$ **documents** is simulated by sampling one topic per document according to the frequencies listed above. Moreover, every document had additional topics sampled according to a Poisson distribution with $\lambda = 0.4$ to simulate multiple topics per document. Most of the documents still had only a single topic, but about a third of the documents were made up of more than one topic. The result was B , a 1000×10 matrix ($D \times V$).

Now let's examine the steps of the process by visualizing the cross-frequency as a graph. Figure 2.5a shows the cross-frequency matrix of the input data, the links between item 2 and all other nodes (except 1) were omitted for clarity as they have the same values as the links from and to item 1. As expected from the list of topics there seem to be 3 almost separate groups of nodes: (1, 2, 3, 4, 5), (0, 5, 7, 8, 9), and (6); with item 5 linking the two big groups.

The first step finds items to combine according to Rule 1, this resulted in the merging of item 1 with item 2 and item 7 with item 9. The results of the merge are shown in Figure 2.5b.

After that, we find nodes to split according to Rule 2, and this results in the split of: node 1+2 between nodes 3, 4, and none; node 5 between nodes 4, and none; node 8 between nodes 0, 7+9, and none. The recomputed cross-frequency matrix is visualized in Figure 2.5c. Of all the new splits, the node 1+2 - (3,4) and the node 8 - (0, 7+9) had a zero count so they are removed in the next step according to Rule 3. Additionally, new nodes can be combined: 7+9 with 8 (7+9), 0 with 8 (0), 3 with 1+2 (3), 4 with 1+2 (4) and 5 (4), as expected. The resulting graph is shown in Figure 2.5d.

This leaves only a single connection in the graph that can be split (Figure 2.5e) and combined to make up the final topics division of Figure 2.5f. In this example, the topics were "discovered" almost perfectly and it indicates that the method has merit at least for a small sample of topics and items.

2.2.4. Correction Based Algorithm

This algorithm is based on the idea that parts, corrections, and symptoms should be treated differently to create topics. In particular, every correction code starts as being its own topic, and all the parts and symptoms are appended to these topics based on the cross frequencies between them. The steps are shown in Algorithm 4.

Algorithm 4: Correction Based Algorithm

- (1) Create one topic per unique correction code.
 - (2) Combine topics if the correction codes in them have cross frequencies higher than a threshold on both directions.
 - (3) Set threshold t and loop over:
 - (a) Find all parts and symptoms with cross-frequency higher than t to a correction code.
 - (b) Add the parts and symptoms to the topics containing the correction codes.
 - (c) Lower t and go back to 3.a. Only consider parts and symptoms that have not been assigned to a topic yet.
-

This algorithm takes some ideas from the Combine and Split algorithm and additionally utilizes knowledge about the difference between correction codes, parts, and symptoms to achieve more accurate and desirable topics.

Computing the topic-word and document-topic distribution Since every correction code is only assigned to a single topic, the value of the topic word distribution for topic k and correction code c is simply the number of times c appears in the population.

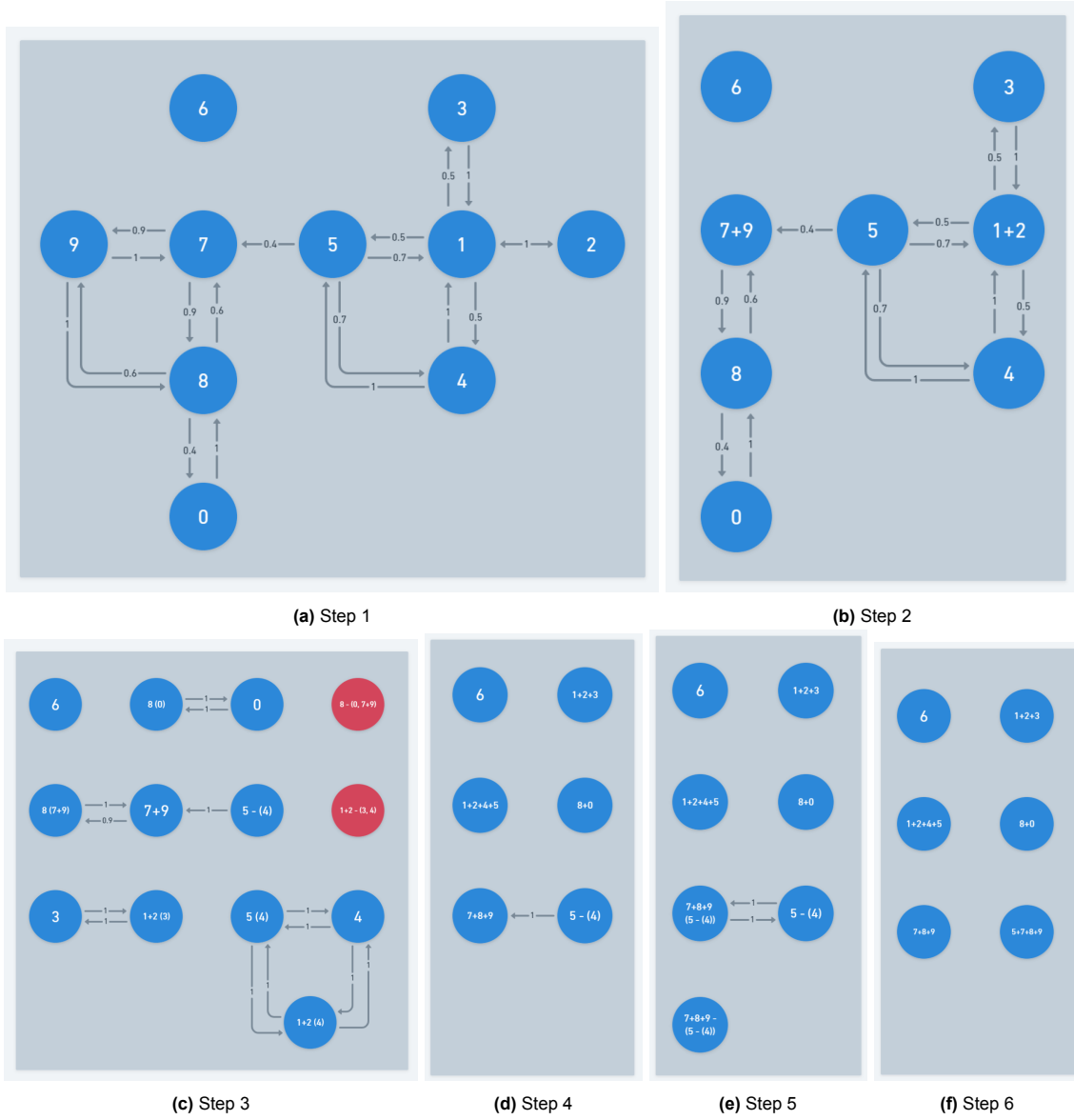


Figure 2.5: Steps of Combine and Split algorithm Example

Parts and symptoms can be assigned to different topics, so a more involved process is needed and it can be described in Algorithm 5.

Algorithm 5: Topic Word Distribution for Parts and Symptoms

- (1) Select a word w .
 - (2) Select all documents with w in it.
 - (3) For every document d :
 - (a) For every correction code c in d , select the topic k containing c .
 - (b) If w was associated with topic k , then: $\varphi_{kw} += \frac{1}{n_{cc}}$, where n_{cc} is the number of correction codes in d
-

Finally, Algorithm 6 describes how to compute the document-topic distribution.

Algorithm 6: Document Topic Distribution

- (1) Select a document d .
 - (2) Select all correction codes present in document d .
 - (3) For every correction code c :
 - (a) Select the topic k containing c
 - (b) $\gamma_{kd} += \frac{1}{n_{cc}}$, where n_{cc} is the number of correction codes in d .
-

2.2.5. Topics Generation Algorithms Conclusions

The algorithms presented above can all be used for topic generation. The results for all of them applied to the same data set can be found in Section 3, where also a comparison of the strengths and weaknesses of the algorithm is discussed. Since the algorithms perform well in different situations and make different assumptions it is important to test them all and compare them to select the best one for our context.

2.3. Novelty Scores

As roughly explained in Section 2.1, the process for computing the novelty of every topic first has to go through the computation of novelty for every word in the vocabulary. The goal of this computation is to assign a novelty score to each individual word at every timestep by looking at the word frequency over time. The algorithm should assign a high novelty score where the frequency graph has a peak, a neutral score when the frequency graph is stationary or linear, and a low (negative) score when the frequency graph has a dip.

Words Novelty Score We could use different methods for achieving the desired novelty scores that satisfy our requirements, the most straightforward would be to compare the frequency of the word at time t with the historical average. Let's say we want to compute the score for the word w at time t , this will result in computing a sort of z-score of the form:

$$z_w(t) = \frac{p_w(t) - \mu_w}{\sigma_w}, \quad (2.12)$$

where μ_w and σ_w are respectively the historical mean and standard deviation of the frequency of w , and $p_w(t)$ is the frequency of w at time t .

The above approach could work in some circumstances, however, it seems to be too simplistic. In many cases, frequencies tend to be seasonal or the words have multiple consecutive spikes that we would like

to capture individually. Both of these scenarios can't be handled properly by only comparing the individual frequencies with the historical statistics. Therefore, as proposed in [5], we make use of a Locally Weighted Linear Regression (LWLR) [39].

"Locally", means that we do not look at the full historical data, but we restrict the scope to the $S - 1$ observations before the current timestep. We will call S the **horizon**. "Weighted" means that data points closer to the current timestep are considered more important and thus get a higher weight, we will define the weight function later. Finally, a "Linear Regression" for w at timestep S is of the form:

$$h_{wS}(x) = a_{wS}x + b_{wS} \quad x = 0, \dots, S. \quad (2.13)$$

Other methods are proposed to evaluate word novelty based on the average frequency of historical data such as in [40] and [41], but this method ensures new-coming words stand out from existing words more significantly.

For simplicity, from now on we will be omitting the S and the w in our notation since all the computations before are valid for every timestep and every word without loss of generality.

To compute the coefficients a and b for a LWLR, we have to minimize the following:

$$J(a, b) = \frac{1}{2} \sum_{x=1}^{S-1} w(x)(h(x) - p(x))^2, \quad (2.14)$$

where $w(x)$ is an arbitrary weight function.

Substituting Eq. (2.13) into Eq. (2.14) we get:

$$J(a, b) = \frac{1}{2} \sum_{x=1}^{S-1} w(x)(ax + b - p(x))^2, \quad (2.15)$$

Taking the partial derivatives of J with respect to a and b gives the following:

$$\frac{\partial J(a, b)}{\partial a} = \sum_{x=1}^{S-1} xw(x)(ax + b - p(x)) \quad (2.16)$$

$$\frac{\partial J(a, b)}{\partial b} = \sum_{x=1}^{S-1} w(x)(ax + b - p(x)) \quad (2.17)$$

Setting the partial derivatives equal to 0 results in the following set of linear equations in a and b :

$$\begin{cases} a \sum_{x=1}^{S-1} x^2 w(x) + b \sum_{x=1}^{S-1} xw(x) = \sum_{x=1}^{S-1} xw(x)p(x) \\ a \sum_{x=1}^{S-1} xw(x) + b \sum_{x=1}^{S-1} w(x) = \sum_{x=1}^{S-1} w(x)p(x), \end{cases} \quad (2.18)$$

which are easily solved.

Finally, after having computed a and b , and thus the function h , we can compute the novelty scores in the following way:

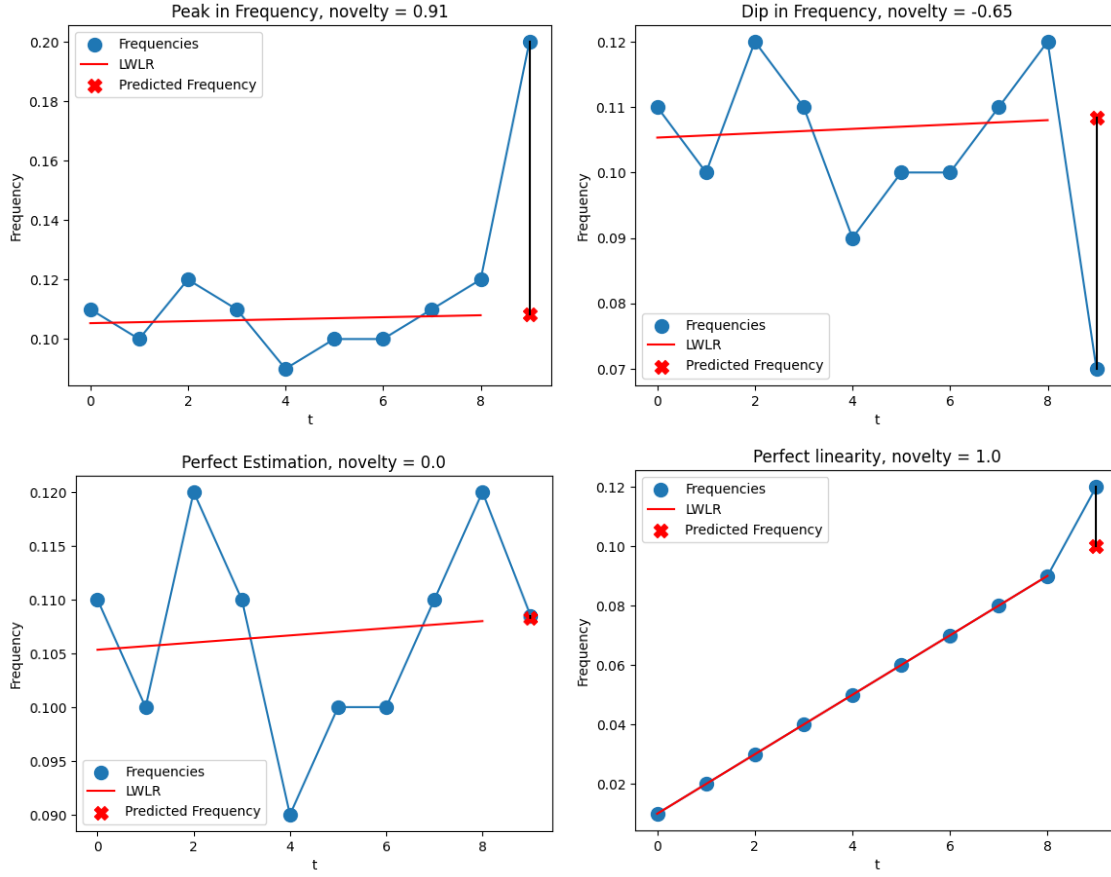


Figure 2.6: Visual examples of novelty scores in different situations

$$novelty = \begin{cases} \frac{w(S)(p(S) - h(S))^2}{\sum_{x=1}^S w(x)(p(x) - h(x))^2} & \text{if } p(S) > h(S); \\ \frac{-w(S)(p(S) - h(S))^2}{\sum_{x=1}^S w(x)(p(x) - h(x))^2} & \text{if } p(S) < h(S); \end{cases} \quad (2.19)$$

Figure 2.6 shows the variables we worked with in different situations. The blue dots are the frequencies $p(x)$, the red line is the representation of the linear regression h , and the red "X" is the estimated value for the frequency at time S . In all the examples the horizon S was 10 and the weight function was constant ($w(x) = 1$).

The novelty score is independent of the absolute frequency, as it gets normalized in Eq. (2.19). We call then $n_w(t)$ the novelty probability for word w at time t and is computed as follows:

$$n_w(t) = novelty_w(t)p_w(t). \quad (2.20)$$

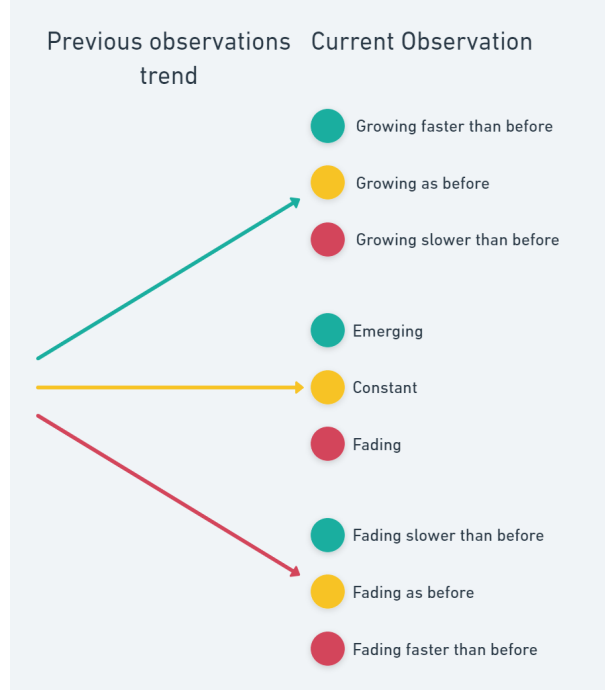


Figure 2.7: Qualitative visualization of the novelty scores

Topic Novelty Score The novelty score for topic k is the combination of the novelty score for the individual words. The words' novelty scores are weighted according to the topic-word distribution φ as in the following:

$$n_k = \sum_w n_w \varphi_{kw}. \quad (2.21)$$

The final value will be a number between -1 and 1 indicating the novelty of the topic. A value > 0 means the topic is growing in popularity at a higher rate than before (peak), a value of 0 means that the frequency of the topic is stationary or linear, and a value of < 0 means the topic frequency is descending faster than before (dip).

Computing the novelty probability allows us to not only figure out emerging topics but also to track their evolution over time and classify them as fading or stationary. This is not the case with other approaches such as the ones presented in [42] and [43].

2.4. Growing Scores

The way we defined the novelty scores encapsulates the deviation of the current frequency from the extrapolation of the previous timesteps. The useful visualization in Figure 2.7 can help understand this, here the arrows represent the trend of the observations previous to the current timestep and the colored dots represent possible current observations, one set for each possible trend line. The green dots represent observations with novelty > 0 , yellow ones a novelty of 0, and red ones a novelty < 0 .

So, if the frequency of a certain item is increasing linearly over time, apart from a spike at the beginning, the

novelty score will be close to 0 the whole time. This falls perfectly within our definition of the novelty score, but it might be useful to consider an additional score that only looks at the trend of the frequency graph. For example, looking again at Figure 2.7, a score that is > 0 in the case of the green arrow, 0 in the case of the yellow arrow, and < 0 in the case of the red arrow.

Computing the growing scores In Section 2.3, we already saw a way of computing the slope of a frequency curve through the means of a LWLR. Thus, we use Eq. (2.14) with one modification: we sum all values up to S instead of $S - 1$ since the current values don't need to be predicted, but it's part of the linear regression. Resulting in:

$$J(a, b) = \frac{1}{2} \sum_{x=1}^S w(x)(h(x) - p(x))^2, \quad (2.22)$$

that can be solved similarly to Eq. (2.14).

Now, a represents the slope of the graph and is related to the growing score we are trying to compute. Much like in Eq. (2.19), we then normalize the value of a to a value between -1 and 1. The maximum possible value of a (in absolute value) for any interval is defined by the following formula:

$$A = \frac{1.5}{S-1} \left(\max_{1 < x < S} p(x) - \min_{1 < x < S} p(x) \right), \quad (2.23)$$

A full derivation of Eq. (2.23) can be found in the next paragraph. The final growing score is then defined as:

$$growing = \frac{a}{A}. \quad (2.24)$$

Figure 2.8, shows some examples of frequency graphs and their growing scores. Finally, the growing probability for word w at time t is defined as:

$$g_w(t) = growing_w(t)p_w(t). \quad (2.25)$$

Topic growing scores The way we defined the novelty scores for each topic at the end of Section 2.3, can be easily extended to the definition of a topic growing score. Eq. (2.21) can be extended to any metric that is valid per item thanks to the topic-word distribution. The growing score of topic k can be computed as follows:

$$g_k = \sum_w g_w \varphi_{kw}. \quad (2.26)$$

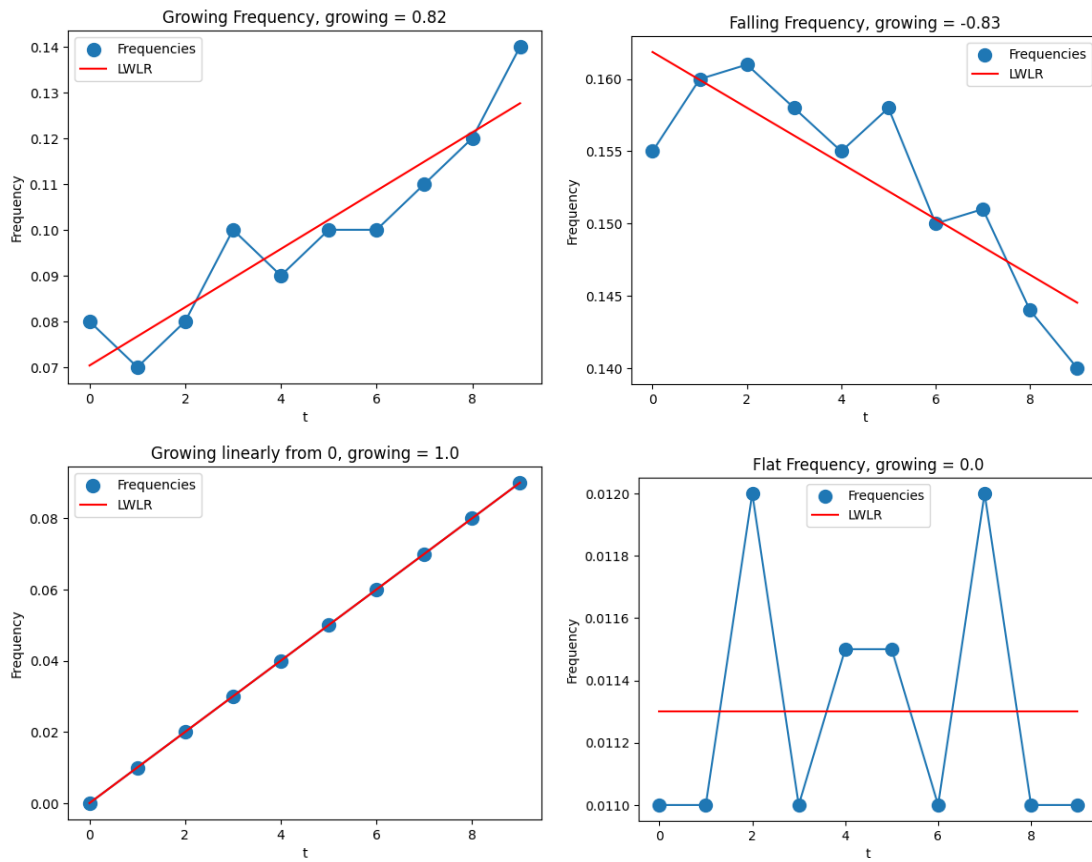
2.4.1. Max Slope

The equation for a linear regression solution computed with least squares in a 2D plane is the following:

$$a = \frac{Cov(x, y)}{Var(x)} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}. \quad (2.27)$$

The problem of finding the maximum slope possible for n points where the x s are equidistant and $y_i \in [0, 1]$ $\forall i$ can be expressed as follows:

$$\max_{0 \leq y_i \leq 1} a = \max_{0 \leq y_i \leq 1} \sum_{i=1}^n (i - \bar{x})(y_i - \bar{y}) = \max_{0 \leq y_i \leq 1} J(\mathbf{y}), \quad (2.28)$$

**Figure 2.8:** Visual examples of growing scores in different situations

since the denominator is independent of the y_i values and we assume that $x_i = i$ as the absolute value of the distance between the x values can be ignored without loss of generality. Substituting the values for the averages of both x and y results in the following equation for J :

$$J = \sum_{i=1}^n \left(i - \frac{n+1}{2} \right) \left(y_i - \frac{1}{n} \sum_{k=1}^n y_k \right). \quad (2.29)$$

Taking the partial derivative of J with respect to the individual y_j we have the following:

$$\begin{aligned} \frac{\partial J}{\partial y_j} &= j - \frac{n+1}{2} - \frac{1}{n} \sum_{i=1}^n \left(i - \frac{n+1}{2} \right) \\ &= j - \frac{n+1}{2} - \frac{1}{n} \left(\sum_{i=1}^n i - \sum_{i=1}^n \frac{n+1}{2} \right) \\ &= j - \frac{n+1}{2} - \frac{1}{n} \left(\frac{n(n+1)}{2} - \frac{n(n+1)}{2} \right) \\ &= j - \frac{n+1}{2}, \end{aligned} \quad (2.30)$$

which means that:

$$\frac{\partial J}{\partial y_j} \begin{cases} < 0 & \text{if } j < \frac{n+1}{2}. \\ = 0 & \text{if } j = \frac{n+1}{2}. \\ > 0 & \text{if } j > \frac{n+1}{2}. \end{cases} \quad (2.31)$$

This result means that having the first half of the $y_j = 0$ and the second half of the $y_j = 1$. If n is odd, the value in the middle won't matter as the $\partial J / \partial y_{(n+1)/2} = 0$. What remains to compute is the value of a and the computations are split into two: for n odd and for n even.

n Odd Computing the Variance:

$$\begin{aligned} \text{Var}(x) &= \frac{1}{n-1} \sum_{i=1}^n \left(i - \frac{n+1}{2} \right)^2 \\ &= \frac{1}{n-1} \sum_{j=-\frac{n-1}{2}}^{\frac{n-1}{2}} j^2 = \frac{2}{n-1} \sum_{j=1}^{\frac{n-1}{2}} j^2 \\ &= \frac{2}{n-1} \frac{\frac{n-1}{2} \left(\frac{n-1}{2} + 1 \right) \left(2 \frac{n-1}{2} + 1 \right)}{6} = \frac{n(n+1)}{12}. \end{aligned} \quad (2.32)$$

Computing the Covariance:

$$\begin{aligned}
 Cov(x, y) &= \frac{1}{n-1} \sum_{i=1}^n \left(i - \frac{n+1}{2} \right) (y_i - \bar{y}) \\
 &= \frac{1}{2(n-1)} \left[\sum_{i=1}^{(n-1)/2} \left(\frac{n+1}{2} - i \right) + \sum_{i=1+(n+1)/2}^n \left(i - \frac{n+1}{2} \right) \right] \\
 &= \frac{1}{2(n-1)} \left[\sum_{j=1}^{(n-1)/2} j + \sum_{j=1}^{(n-1)/2} j \right] = \frac{1}{n-1} \sum_{j=1}^{(n-1)/2} j \\
 &= \frac{1}{n-1} \frac{\frac{n-1}{2} \left(\frac{n-1}{2} + 1 \right)}{2} = \frac{n+1}{8}.
 \end{aligned} \tag{2.33}$$

Finally, computing a :

$$a_{max} = \frac{Cov(x, y)}{Var(x)} = \frac{n+1}{8} \frac{12}{n(n+1)} = \frac{3}{2} \frac{1}{n}. \tag{2.34}$$

n Even Computing the Variance:

$$\begin{aligned}
 Var(x) &= \frac{1}{n-1} \sum_{i=1}^n \left(i - \frac{n+1}{2} \right)^2 = \frac{2}{n-1} \sum_{i=1}^{n/2} \left(\frac{n+1}{2} - i \right)^2 \\
 &= \frac{2}{n-1} \sum_{j=1}^{n/2} \left(j - \frac{1}{2} \right)^2 = \frac{2}{n-1} \left[\sum_{j=1}^{n/2} j^2 - \sum_{j=1}^{n/2} j + \sum_{j=1}^{n/2} \frac{1}{4} \right] \\
 &= \frac{2}{n-1} \left[\frac{n}{2} \frac{n+2}{2} \frac{n+1}{6} - \frac{n}{2} \frac{n+2}{4} + \frac{n}{8} \right] \\
 &= \frac{n^3 - n}{12(n-1)} = \frac{n(n+1)}{12}
 \end{aligned} \tag{2.35}$$

Computing the Covariance:

$$\begin{aligned}
 Cov(x, y) &= \frac{1}{n-1} \sum_{i=1}^n \left(i - \frac{n+1}{2} \right) (y_i - \bar{y}) \\
 &= \frac{1}{2(n-1)} \left[\sum_{i=1}^{n/2} \left(\frac{n+1}{2} - i \right) + \sum_{i=1+n/2}^n \left(i - \frac{n+1}{2} \right) \right] \\
 &= \frac{1}{2(n-1)} \left[\sum_{j=1}^{n/2} \left(j - \frac{1}{2} \right) + \sum_{j=1}^{n/2} \left(j - \frac{1}{2} \right) \right] \\
 &= \frac{1}{n-1} \left[\sum_{j=1}^{n/2} j - \sum_{j=1}^{n/2} \frac{1}{2} \right] = \frac{1}{n-1} \left[\frac{n(n+2)}{8} - \frac{n}{4} \right] \\
 &= \frac{n^2}{8(n-1)}
 \end{aligned} \tag{2.36}$$

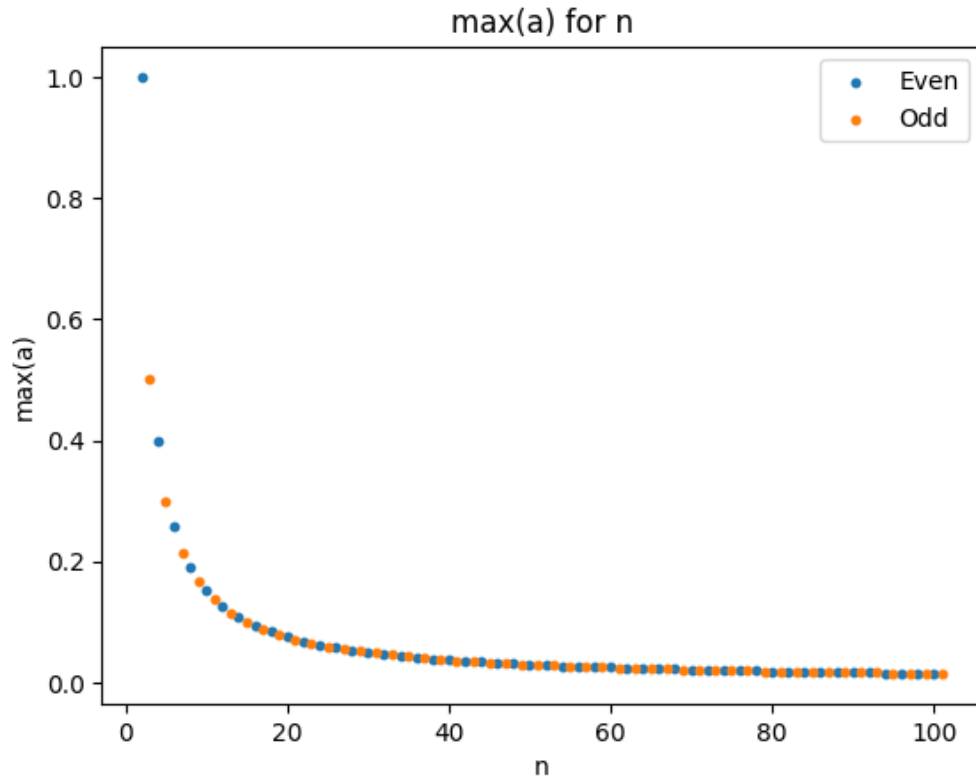


Figure 2.9: Plot of a_{max} for the first values of n

Finally, computing a :

$$a_{max} = \frac{Cov(x, y)}{Var(x)} = \frac{n^2}{8(n-1)} \frac{12}{n(n+1)} = \frac{3}{2} \frac{n}{n^2 - 1} \quad (2.37)$$

Combining the results Figure 2.9 shows the max value of a for every value of n , and it shows that a_{max} can be approximated by either Eq. (2.34) or Eq. (2.37). Since Eq. (2.34) is simpler, that is the chosen equation. Additionally, if the interval within which the y_i are constrained is not $[0, 1]$, but is $[y_{min}, y_{max}]$ then all the analysis remains the same, and the values of a_{max} is:

$$a_{max} = \frac{1.5(y_{max} - y_{min})}{n} \quad (2.38)$$

2.5. Computing Cost Metrics

For every repair order job data points can be collected to compute the cost for each repair. The following are the most influential items to the overall cost, let's recall their definitions first introduced in Section 1.4:

- **Pay type:** were the parts replaced in warranty or not, who ended up paying for the visit;
- **Parts Cost:** the cost of the individual parts replaced during the service visit;

- **Flat Rate Time:** "hands-on" of time spent by the technician working directly on the car;
- **Days in Service:** amount of days the car has spent in the Service Center.

The warranty type can distinguish between repairs that were paid by the customer and repairs that were in warranty and thus paid by Tesla. The other three are the factors that most contribute to the cost of a service visit. In this section, a way to combine all of the above to compute cost information per topic is presented.

2.5.1. FRT Cost

If FRT_d represents the Flat Rate Time associated with the repair job d , then the value of the Flat Rate Time associated with a topic k (FRT_k) can be computed by using the document-topic distribution as follows:

$$FRT_k = \sum_{d=1}^D \gamma_{kd} FRT_d. \quad (2.39)$$

To compute the FRT associated with only Tesla-paid repairs, then it is sufficient to have the sum go over only those service visits, likewise for customer-paid repairs.

2.5.2. Parts Cost

Similarly to the FRT, we can compute the total cost of parts for every topic k as follows:

$$total_part_cost_k = \sum_{w=1}^V cost(w) \sum_{d=1}^D \gamma_{kd} B_{dw}, \quad (2.40)$$

and the sum over d can be restricted to customer-paid or Tesla-paid repairs depending on the quantity computed.

2.5.3. Days in Service Cost

Both FRT and parts are associated with a repair order job directly. The number of days the car spends in the Service Center, however, is associated with the overall service visit, which can be (and usually is) split into several repair order jobs. So the number of days in service queried from the database needs to first be divided by the number of repair order jobs associated with the service visit. After that, the computation for the days in service for each topic are very similar to the one for the FRT in Eq. (2.39):

$$DiS_k = \sum_{d=1}^D \gamma_{kd} DiS_d. \quad (2.41)$$

2.6. Car Configuration analysis

As explained in Section 1.2, vehicles have properties such as the vehicle model, the number of drive units, which factory it was built at, and when. Some properties are static and some of them dynamic, and for every parameter there are many options available. When analyzing failure modes, parameters such as car model, factory, and firmware version are almost always investigated to see if there are any discernible patterns. There are, for example, failure modes that are unique to some specific model or that are introduced by specific firmware versions. Due to the vast number of options, however, it is a hard task to pull and compare every parameter, especially if the useful ones are non-obvious.

Additionally, every Service Visit can be connected to one or more of the topics computed in Section 2.2, which means that lists of cars can be created per topic. It would be useful to know which configurations are the most prevalent in those lists since that can help identify the core issue behind the repair or point out where to focus efforts to reduce the cost of the repair.

For the reasons above, in this section, a simple method for calculating outstanding configurations from a group of cars is proposed.

As seen in Figure 1.2, the data is stored in databases with one row per car. Every row has many columns, one for each **configuration**, and every configuration has many different **options**.

Let us consider the case of comparing the configurations of a group of cars with the population. We start by calculating the overall probability for every configuration c and every option o by simply doing the following:

$$p_{co} = \frac{\text{\#cars with option } o \text{ for config } c \text{ in the population}}{\text{\#cars in the population}}. \quad (2.42)$$

If the population is large enough, we expect the number of cars with option o for config c in the group to be distributed like a Binomial distribution: $X_{co} \sim \text{Bin}(N, p_{co})$, where N is the number of cars in the group. If n_{co} is the number of cars in the group with option o for config c , then we can compute the following:

$$z_{co} = \mathbb{P}(X_{co} > n_{co}) = \mathbb{P}\left(\frac{X_{co} - Np_{co}}{Np_{co}(1 - p_{co})} > \frac{n_{co} - Np_{co}}{Np_{co}(1 - p_{co})}\right) \quad (2.43)$$

If N is large enough the left-hand side of the inequality can be approximated with a normal distribution.

$$z_{co} \approx \mathbb{P}\left(Z > \frac{n_{co} - Np_{co}}{Np_{co}(1 - p_{co})}\right) = 1 - \Phi\left(\frac{n_{co} - Np_{co}}{Np_{co}(1 - p_{co})}\right). \quad (2.44)$$

Computed as above, z_{co} is a value between 0 and 1 that represents a measure of the difference between the distribution of the overall population and the distribution of the group. A low z_{co} means that the specific option is present more in the group than in the population, while a high value of z_{co} means that the opposite is true.

Sorting the z_{co} for all c s and all o s from lowest to highest results in an ordered list of configuration options that are more unique to the group compared to the overall population. This method can be used to generate additional useful information about the topics.

2.7. Software Used

In this section, an overview of the software tools that were used for this project is given, together with their applications.

2.7.1. Tableau

Tableau [44] is a data visualization software platform that allows the user to define helpful and interactive visualizations of all kinds of data types. Tesla makes use of Tableau extensively and in many different contexts, from keeping track of failures over time to displaying information about the fleet. Almost all technical divisions at Tesla create and maintain numerous **dashboards**. Those dashboards are then hosted on a server and are accessible for everyone to interact with.

During my internship at Tesla, I interacted with Tableau significantly and learned how to use it in the process. Many of the data produced by this project were presented to other people through different Tableau dashboards. All the different types of graphs are equipped with a lot of functionality, such as sorting, tool-tips, and selections. These features make it intuitive and effective to navigate large amounts of data. Tableau also encourages the use of dynamic elements that can filter or select different data to show depending on the interests of the person interacting with the dashboard. Additionally, it was really easy to share the visualizations due to Tesla servers dedicated to hosting them.

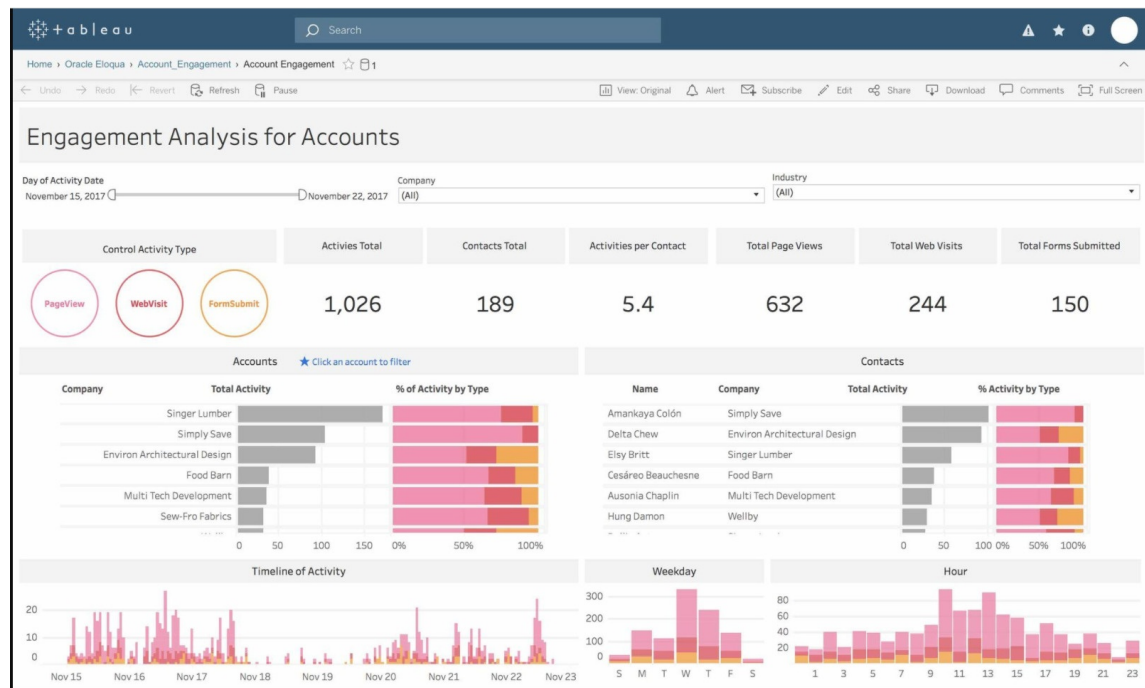


Figure 2.10: Example of Tableau dashboard

Another convenient feature of Tableau is the ability to get data from a variety of data sources. Tableau supports local files, spreadsheets, relational and non-relational databases, and more. Any of Tableau's data sources may be readily connected and combined with data from other sources to generate visuals. For my use case, the ability to query data from Tesla databases as well as files on my computer was crucial.

2.7.2. SQL

The majority of the data needed for this project was stored in multiple different Tesla databases and a substantial amount of time was spent querying and organizing the data from said databases. The queries were done in the SQL (Structured Query Language), which is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system introduced first in 1970 in [45]. It is particularly useful in handling structured data.

SQL, or slight modifications of it, is the default for most databases and it is one of the most used programming languages in the world. SQL has 2 main advantages over other APIs. Firstly, many records can be accessed with a single command. Secondly, it eliminates the need to specify how to reach a record, for example, with the use of an index. An example of a SQL query used for this project can be seen in Figure 2.11.

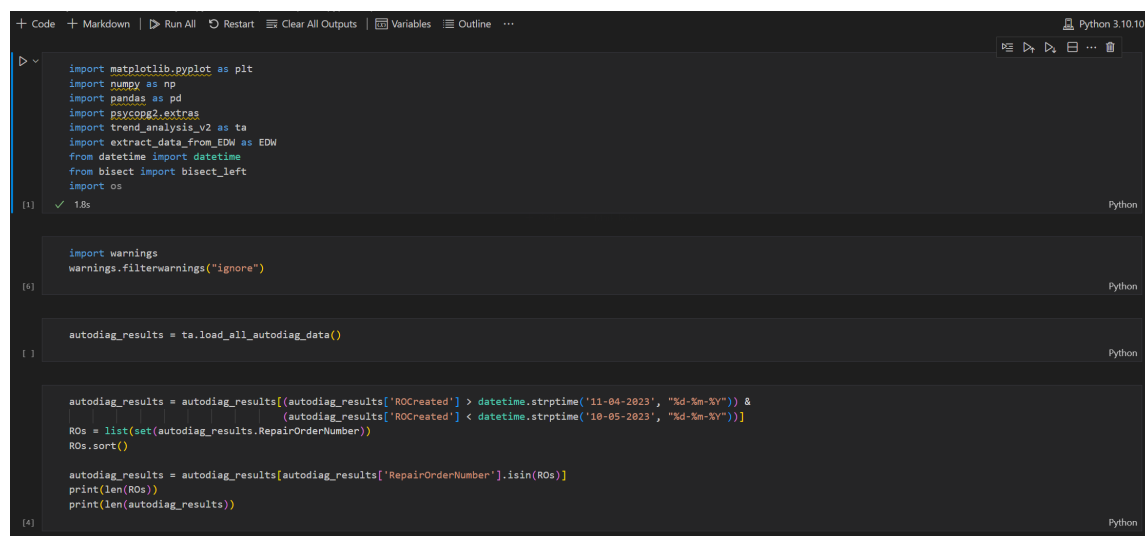
2.7.3. Python

All the code written for the project was written in Python with the use of Jupyter Notebooks [46], an example can be seen in Figure 2.12. Python offers easy-to-use libraries for connecting to all types of databases with simple and intuitive APIs. The handling of the results of the SQL queries was done using the package *pandas* [47] as well as the data pre-processing. The package *numpy* [48] was used for most of the computations, as it is a

```
1 WITH employee_ranking AS (  
2   SELECT  
3     employee_id,  
4     last_name,  
5     first_name,  
6     salary,  
7     RANK() OVER (ORDER BY salary DESC) as ranking  
8   FROM employee  
9 )  
10 SELECT  
11   employee_id,  
12   last_name,  
13   first_name,  
14   salary  
15 FROM employee_ranking  
16 WHERE ranking <= 5  
17 ORDER BY ranking
```

Figure 2.11: Example of an SQL query

very efficient and effective library for array programming. The LDA algorithm implementation was taken from the package *scikit-learn* [26], in particular the class: *sklearn.decomposition.LatentDirichletAllocation*. Finally, the graphs were made using the package *matplotlib* [49].



The screenshot shows a Jupyter notebook with a dark theme. The top bar includes tabs for 'Code' and 'Markdown', and buttons for 'Run All', 'Restart', 'Clear All Outputs', 'Variables', and 'Outline'. The Python version '3.10.10' is displayed on the right. The notebook contains four code cells:

```
[1]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import psycog2.extras
import trend_analysis_v2 as ta
import extract_data_from_EDM as EDM
from datetime import datetime
from bisect import bisect_left
import os
```

[2]: import warnings
warnings.filterwarnings("ignore")

[3]: autodiag_results = ta.load_all_autodiag_data()

[4]: autodiag_results = autodiag_results[(autodiag_results['ROCreated'] > datetime.strptime('11-04-2023', "%d-%m-%Y")) &
(autodiag_results['ROCreated'] < datetime.strptime('10-05-2023', "%d-%m-%Y"))]
ROs = list(set(autodiag_results.RepairOrderNumber))
ROs.sort()

autodiag_results = autodiag_results[autodiag_results['RepairOrderNumber'].isin(ROs)]
print(len(ROs))
print(len(autodiag_results))

Figure 2.12: Beginning of a Jupyter notebook used during the project

3

Topic Generation Results

In this chapter, the results for the different topic generation algorithms are presented. In Section 3.1 the results for the LDA algorithm are shown and discussed. In Section 3.2 the results computed with the GSDMM algorithm are shown. Section 3.3 shows the results for the Combine and Split algorithm and, finally, Section 3.3.1 shows the results for the Correction Based Algorithm.

3.1. Latent Dirichlet Allocation Results

The results presented in the following sections for the LDA algorithm were computed with data for a single month. The dataset used for this analysis was separated as follows:

- 800,000 parts replaced, of which 10k are unique;
- 2.1 million correction codes applied, of which 4k are unique;
- 300,000 symptoms reported, of which 500 are unique.

It is easy to see that the amount of data for just a single month of service records is quite large. To put into perspective, the matrix of 64-bit floating point numbers (float64) needed to compute the topics by the LDA algorithm (Algorithm 1) would be almost 100 GB, a value that is at the limit of the very best computers' RAM and greatly surpassed the capabilities of the 32GB of RAM in normal computers. Simply storing the input data is a challenge that cannot be blindly handed to Python or the *numpy* package. The default type in *numpy* is a float64 and, as we saw above, storing a full matrix of float64 in memory is not possible. So the input matrix had to be represented in memory with a datatype that used less than 8 bytes, for this reason, the input matrix was stored using the *numpy* type *i1* (8-bit signed integer) that only occupies 1 byte of memory. Thus the input matrix had a memory footprint of a little over 12GB.

3.1.1. Data Pre-processing

From the rough numbers given above it seems clear that there should be a pre-processing step to filter the repair order jobs and the items. As explained in [50], for LDA applied to Natural Language Processing, it is common practice to apply the following three pre-processing techniques to the input data:

1. **Document deduplication:** Working with textual documents, it is possible to encounter very similar or identical copies in a dataset. It is usually pretty straight-forward to detect duplicated documents due

Original	The mission of Tesla is to accelerate the world's transition to sustainable energy and in order to achieve that, the goal is to produce and sell 20 million vehicle per year and reach level 5 autonomy.
Stop words highlighted	The mission of Tesla is to accelerate the world's transition to sustainable energy and in order to achieve that, the goal is to produce and sell 20 million vehicle per year and reach level 5 autonomy.
Stop words removed	mission Tesla accelerate world transition sustainable energy order achieve that, goal produce sell 20 million vehicle per year reach level 5 autonomy
Stemming highlighted	mission Tesla acceler ^{ate} world's transit ^{ion} sustain ^{able} energy order achiev ^e that, goal produc ^e sell 20 million vehic ^{le} per year reach level 5 autonomy
Final	mission Tesla acceler world transit sustain energy order achiev that, goal produc sell 20 million vehicl per year reach level 5 autonomy

Table 3.1: Pre-processing steps visualized for an example phrase

to the length of text-based data. If the amount of duplicated documents is significant, it could lead to reduced performance of the model. So this technique aims to remove such duplicated documents.

2. **Stopword removal:** Extremely common words like: "the", "an", "and", "are", "is" are called **stop words**. The name derives from the fact that these words are part of a **stop list**, the words in this list are usually filtered out before the processing of the data due to them having little to no significance since they don't have a specific meaning. The stop list can be different depending on the context and the analysis being performed.
3. **Stemming:** In almost every language, some words are derived from others. For example, the noun "attachment" is derived from the verb "attach" and it is common to see these words be used in similar contexts. With stemming, every word is remapped to its root word (or stem), thus reducing the total number of words in the vocabulary.

Table 3.1 shows the ideas of stopwords removal and stemming applied in the natural language context. The phrase taken as an example is the very first phrase of Section 1.

For our purposes, **document deduplication** is not relevant since the data is already free of any duplicates, every repair order job is tied to an actual event in the real world. This does not mean that there are no repair order jobs that display the same parts/corrections/symptoms, but it means that identical repair order jobs are significant to the analysis and should not be removed since they represent useful information that cannot be omitted.

In our context **stopword removal** is very relevant and translates to removing items with high frequency and low information. The items with really high frequency are only a handful so the stop list can be created manually. For this purpose, the 100 most frequent items were reviewed and the ones that were too general and would not add useful information to the analysis were added to the stop-list. Some examples are the following correction codes: *Perform General Inspection*, *General Diagnosis*, and the symptom: *Courtesy Service Provided*.

Since we are not dealing with language, **stemming** cannot be applied in the same way as in NLP. However, a similar concept can be used to combine some of the parts. The part numbers used to identify parts are

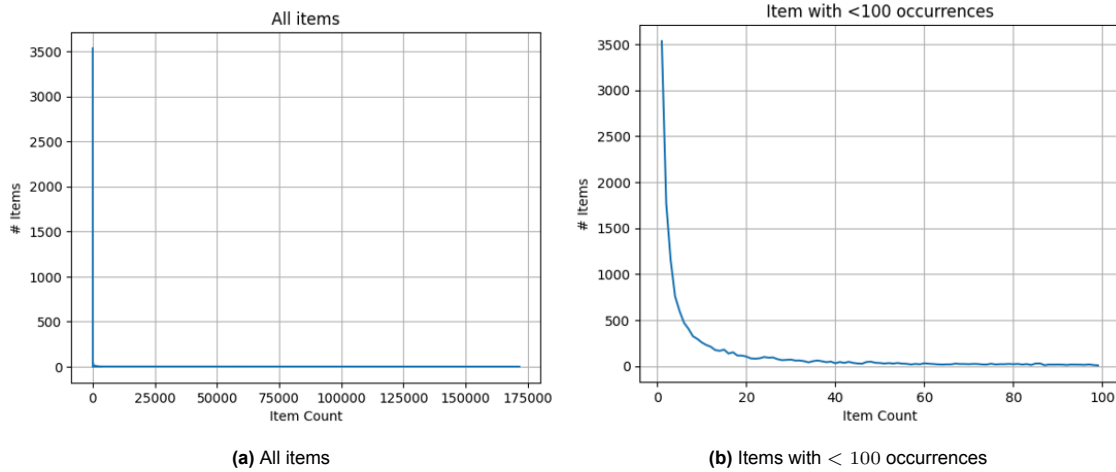


Figure 3.1: Item counts distribution before pre-processing

separated into 2 codes, the first part is a 7-digit number (base part number) and the second is a 3-character identifier (suffix). For some base part numbers, there could be multiple parts with different suffixes that are related, but slightly different. For example, they could be different revisions of the same part, be fitted for different countries, or be different colors or materials. It is useful to combine parts with the same base part number into a single part for the same reason that stemming is done on words, the parts are very closely related. Additionally, in our application, most of the time parts with the same base and different suffixes are mutually exclusive, which makes it simple to combine them with little repercussion to the rest of the data.

Applying the techniques above helped to reduce the number of items by around 350, roughly 340 "stemmed" parts, and 10 high-frequency low information correction codes and symptoms. So the quality of the inputs increased, but the problem with the amount of data persists. By looking at the frequency distributions of the items in Figure 3.1, we can see that the majority of the items have a very low frequency, with more than 3,500 occurring only once in a full month of Service Records.

It is then reasonable to focus our data reduction on these low-frequency items. Here is where we need to use more domain knowledge to guide us. We could remove all items with a frequency lower than a specified number, but by doing so we risk removing parts that are valuable to the analysis. For example, there are a lot of types of HV batteries and some of them are replaced rarely (less than 20 times a month), those batteries are, however, some of the most expensive parts for Service so removing them would remove very valuable items for determining the total costs involved.

With the above in mind, I decided to remove items from three different categories:

1. **Fasteners.** Fasteners are the combination of bolts, screws, and studs. They are really cheap and frequently used during replacements and maintenance. Fasteners are usually paired with more specific parts and do not convey as much information as other parts. Fasteners could be seen as an extension of the **stopwords** list, although not as ubiquitous as the items in that list. In total, of the 10k unique parts, 650 of them were classified as fasteners. The cost of all the fasteners amounts to a very small percentage of the total parts costs.
2. **Low cost parts.** This group is defined by the parts that have < 100 occurrences and < \$1,000 of total cost. Total cost here refers to the cost of the specific part times the occurrences. The 100 occurrences and the \$1,000 are numbers that can be changed and could be tuned to obtain better performance.

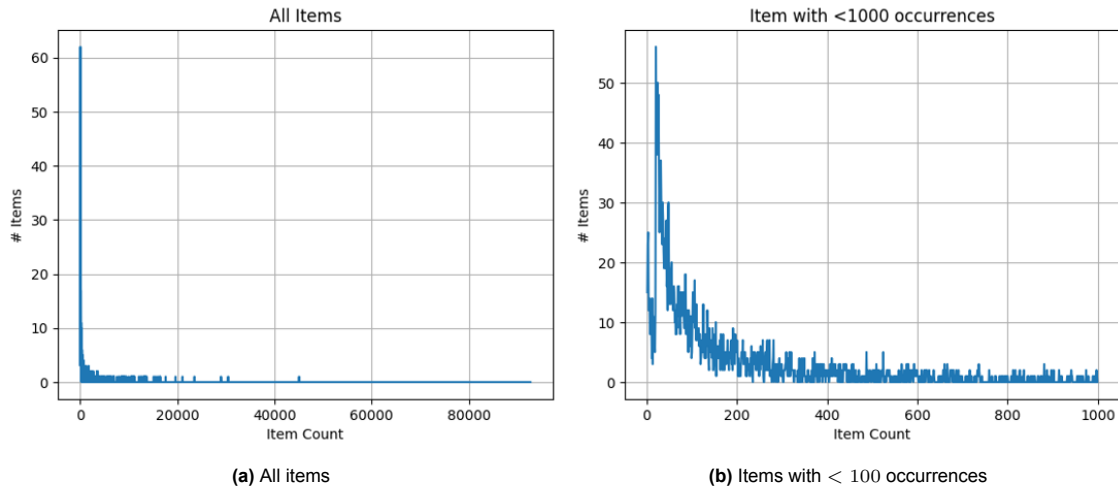


Figure 3.2: Item counts distribution after pre-processing

Much like the fasteners category above, this group of parts has low impact and can be safely removed. This group totaled 7,500 parts and the total cost of them was higher than that of the fasteners, but still in a range that could be ignored.

3. **Low frequency items.** This group is defined by any item with less than 20 occurrences, with an exception for parts of which total cost is greater than \$3,000. Much like for the group above, the \$3,000 threshold can be changed and adapted for better results. This group totaled 3,500 items.

Finally, some jobs were discarded due to their non-interesting nature. The most frequent example was that of *Merchandise / Over-The-Counter Sale*. These are technically saved in the databases as separate job lines, but they do not interest the car or are relevant for any repair. Most of these are linked to the sale of accessories (like key batteries) and charging adapters or equipment. All repair order jobs with the *Merchandise / Over-The-Counter Sale* symptom or any of the relevant correction codes were removed.

After all the filtering was applied, the resulting dataset was made up of roughly 650 thousand repair order jobs and 3,500 items. As we can see from comparing Figure 3.1 and Figure 3.2, the majority of the low-frequency items were removed as well as some of the highest-frequency ones. To compare data size, the 100GB matrix needed for computation before pre-processing turned into a 16GB matrix, which can be easily stored in memory.

3.1.2. Topics Generated

As explained in Section 2.2.1, the two inputs to the LDA algorithm are: the bag-of-words representation of the corpus and the number of topics to generate K . The former is computed from the pre-processed data from Section 3.1.1, while the latter is still a free parameter.

The usual approach for selecting the optimal K is done through the help of a quantity called **perplexity**. The perplexity can be computed on the results generated by the LDA algorithm run with different values of K , the results with lower perplexity will, in general, be the ones most closely matching the true topics distributions. Various models for perplexity have been proposed over the years, starting with the original LDA paper [11] in 2003 and continuing with [51] and [52].

The LDA algorithm was run on the input data with the following values of K : 700, 1000, and 1200. Usually, the

penalty for using too large of a value of K is having repeated and/or diluted topics that have low recognition. In our application, however, the effect was much more dramatic. The results for $K = 1000$ and $K = 1200$ showed many hundreds of "empty" topics (293 for $K = 1000$ and 474 for $K = 1200$). By **empty**, it is meant that the topic-word distribution φ for that specific topic shows the same low probability for every word in the dictionary. The reason for this phenomenon is hypothesized in later sections.

The results for $K = 700$, did not show a significant proportion of empty topics, and since the number of nonempty topics seen by increasing the value of K resulted in numbers close to 700 the following results and examples are taken from running the LDA algorithm with $K = 700$.

The run times of the algorithm on the machine (Lenovo 20Y4 - Intel i7 @2.5GHz, 8 Cores - 32GB of RAM) used to get the results are displayed in Table 3.2.

K	Run time [min]
700	64
1000	72
1200	78

Table 3.2: LDA run time

Example Topics The topics shown in Table 3.3, 3.4, and 3.5 are the first few in the list generated, so they serve as "random" topics.

Topic 0		
Item	Type	φ
Door difficult to close	Symptom	0.57
Exterior Adjustment - 0.4	Correction	0.23
Exterior Adjustment - 0.6	Correction	0.20

Table 3.3: LDA Topic 0

Topic 0 from Table 3.3 seems to be a very good topic. "Door difficult to close" is paired most often with the two corrections in the table and no other item has similar correlations. The values of φ also represent well the dynamic, as only 30% of the time the two correction codes appear when the "Door difficult to close" is present. Both "Exterior Adjustment" corrections are more popular than the symptom and so they appear in more than just this topic. In fact, they appear in 11 and 19 more topics respectively.

Topic 1		
Item	Type	φ
Reverse camera inoperative	Symptom	0.74
Trunk Handle	Part	0.13
Rear Camera (Remove & Replace)	Correction	0.13

Table 3.4: LDA Topic 1

Topic 1 from Table 3.4 appears to be another good topic. When the "Trunk Handle" part is replaced, almost always there is a "Reverse camera inoperative" symptom and a "Rear Camera" correction, no other item is even close to the same cross frequencies. The "Rear Camera" correction is also associated with a specific rear

camera part, but there is another topic dedicated to this interaction. Additionally, "Reverse camera inoperative" is much more common than the other two items, so the values of φ make sense.

Topic 2		
Item	Type	φ
Service Bulletin	Symptom	0.24
Inspect Vehicle Firmware Version	Correction	0.20
Seat heater does not produce heat	Symptom	0.15
Fog lights have poor performance	Symptom	0.10
Front left seat Trim	Part	0.06
Power Strut O-ring	Part	0.06
Driver Seat Trim Cushion	Correction	0.06

Table 3.5: LDA Topic 2

Topic 2 from Table 3.5 is the worst topic of the first three shown. The "Service Bulletin" and the "Inspect Vehicle Firmware Version" are always paired together in the input data, however, there are no occurrences of "Inspect Vehicle Firmware Version" with any of the other items. "Seat heater", "Front left seat Trim" and "Power Strut O-ring" do appear together. "Fog lights have poor performance" only appears a handful of times with "Service Bulletin" and with no other item, same for "Driver Seat Trim Cushion". The "Service Bulletin" symptom is associated with different types of repair and thus the topic is the aggregation of multiple ones. In this case, it seems that this topic is the aggregation of multiple ones, thus being uninformative and not too useful. Also, the φ values do not seem to map to anything meaningful, as the relative frequencies of the items in the data do not correspond to those values.

Topic 3		
Item	Type	φ
HV Battery (Remove & Replace)	Correction	0.24
Discharge HV Battery	Correction	0.24
Decrease in estimated range	Symptom	0.16
HV Battery 75kWh	Part	0.12
Battery Alert	Symptom	0.10
Coolant	Part	0.06
Penthouse seal	Part	0.05
Refill Coolant	Correction	0.02
Fuse HV Battery (Remove & Replace)	Correction	0.06

Table 3.6: LDA Topic 3

Topic 3 from Table 3.6 is quite a large topic, incorporating seemingly completely different repairs such as coolant refill and Hv battery replacement. However, these repairs in practice are very much linked. The HV battery is water-cooled, so when replacing it the cooling lines are inevitably affected and thus need to be refilled.

Most Popular Topics The following topics shown in Table 3.7, 3.8, and 3.9 were the most frequent according to the results.

In Topic 523 from Table 3.7, the "Perform Vehicle Inspection" symptom is one of the most common items in the

Topic 523		
Item	Type	φ
Perform Vehicle Inspection	Symptom	0.21
Paint Protection Film - Rear Doors	Correction	0.14
Paint Protection - Right	Part	0.14
Paint Protection - Left	Part	0.14
Mud Flap	Part	0.13
Retrofit Request	Symptom	0.12
PDI	Correction	0.12
Retrofit Paint Protection Film	Symptom	0.12
Retrofit Mud Flap	Correction	0.12

Table 3.7: LDA Topic 523

data, and 35% of the time it is present with "PDI". The "Paint Protection Film - Rear Doors" correction is also quite frequent and almost 100% of the time it comes with both "Paint Protection" parts and the "Retrofit Paint Protection Film" symptom. About 20% of the time "Mud Flap" and "Paint Protection" are replaced together and almost always when the "Mud Flap" is replaced there is a "Retrofit Request" symptom and a "Retrofit Mud Flap" correction. This topic seems to be the combination of a couple of popular repairs that a low percentage of the time are done together. The fact that "Perform Vehicle Inspection" is present in this topic makes the frequency higher than the single repairs would imply.

Topic 400		
Item	Type	φ
Perform Vehicle Inspection	Symptom	0.64
PDI	Correction	0.36

Table 3.8: LDA Topic 400

As said for Topic 523, the items of Topic 400 from Table 3.8 are seen together quite often in the data and are relatively frequent. This topic by itself is not particularly interesting as the items are very general, but it is to be expected that some of the topics would be of this type even with some of the most general items removed.

Topic 559		
Item	Type	φ
Hinge Support Grommet	Part	0.20
Service Bulletin	Symptom	0.20
Arrowhead Domed Tie	Part	0.20
Cable Tie	Part	0.20
Inspect And Retrofit Trunk Lid Harness	Correction	0.20

Table 3.9: LDA Topic 559

Topic 559 from Table 3.9 represents basically a perfect topic. When the "Hinge Support Grommet" was replaced almost 100% of the time all other four items are also present. This hints at a very specific repair that was performed several times in the timeframe of the data.

Some Bad Topics These final topics shown in Table 3.10 and 3.11 are examples of bad topics.

Topic 30		
Item	Type	φ
Service Containment Activity	Symptom	0.65
Falcon Door noisy	Symptom	0.22
Fan Shroud	Part	0.07
Fan Condenser Left (Remove & Replace)	Correction	0.06

Table 3.10: LDA Topic 30

Topic 30 from Table 3.10, much like Topic 2, is messy and does not have any engineering meaning. The "Fan Shroud" and the "Fan Condenser Left" are coupled in the data, but are not linked in any way with the other two items. "Service Containment Activity" and "Falcon Door noisy" are also not linked.

Topic 266		
Item	Type	φ
Radio will not change stations correctly	Symptom	0.51
Wiper Arm Left (Remove & Replace)	Correction	0.23
Wiper Arm Pair (Remove & Replace)	Correction	0.17
Panoramic roof makes noise	Symptom	0.09

Table 3.11: LDA Topic 266

In Topic 266 from Table 3.11, even though "Wiper Arm Left" and "Wiper Arm Pair" are associated with similar items, they are rarely seen together and never with the other two symptoms in this topic.

Figure 3.3 shows the frequency distribution of all the topics. The majority of the topics have very low frequencies and the distribution is very skewed towards 0, with more than 600 topics with a frequency of less than 0.0025.

3.1.3. Conclusions

These example topics show some of the strengths of LDA and its potential in this application, but they also expose some of the weaknesses it has in this specific context. The main strength of the algorithm is its ease of use and its general applicability to many situations. However, LDA is not designed for short-text applications, such as this, and thus fails at generating consistent topics. Going over all the topics, most of them are in the "good" category, but there were enough "bad" topics (like 2, 30, and 266). Some topics are unfocused and others are a mix of multiple repairs with low interpretability. Additionally, even after extensive pre-processing, about 36% of the 3,500 items were not attached to any topics. All of these drawbacks make LDA not a suitable enough solution by itself for the topic generation problem.

Let us then expand on the assertions above and discuss the benefits of the LDA algorithm in our application. The topics generated are mostly good, as they combine items that are seen in the field to be correlated and retain the engineering meaning of single repair (topics 0, 1, and 559 are examples of this).

The other advantage is the generality of the algorithm. It can be applied with little set-up and if more types of data get introduced at a later date, the process can be easily extended to accommodate for it.

However, while the majority of the topics have a good engineering meaning, there is still a significant portion of them that seem to be either the mix of multiple individual repairs or just unfocused and random. This inconsistency makes it hard to trust the topics generated.

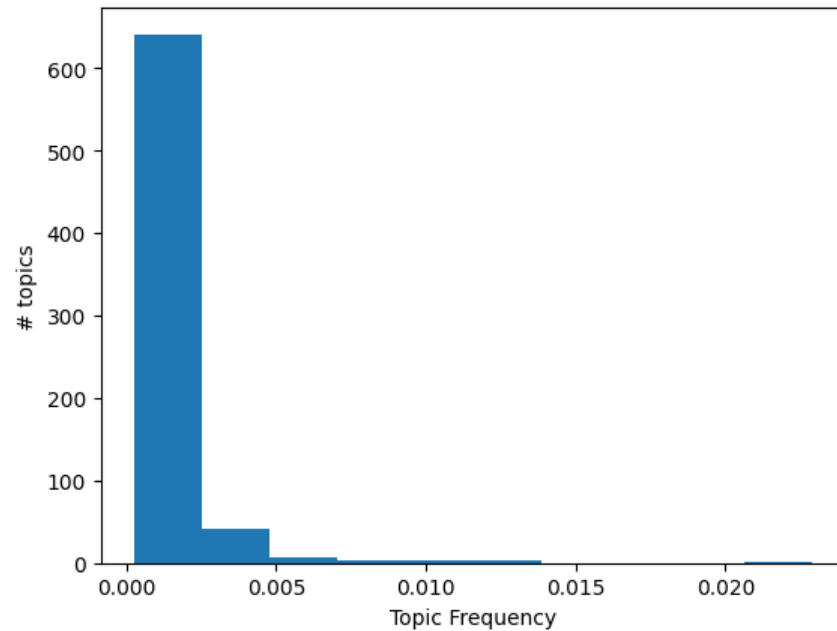


Figure 3.3: Topics' Frequencies Distribution

Another major problem is the fact that more than a third of the items are not categorized in any topic according to the topic-words distribution φ . Even though these items have lower frequency on average, there are plenty of examples of replaced parts with low frequency but very high impact on the overall cost. For example, the most expensive part of an electric vehicle is the HV battery. Fortunately, the frequency of HV battery replacements is relatively low compared to most other replacements, but, due to the high price of these parts, the cost is very significant and absolutely cannot be ignored if the goal is to have an accurate picture of all service costs.

Even though the pre-processing of the data was done carefully to avoid removing useful information from the inputs, it is still an extra step that is not strictly needed. It also adds a point where human supervision is required and increases the possibility of mistakes.

Another aspect that is not ideal is the fact that increasing the number of topics does not change the outputs much. This means that LDA believes the 700 topics it generates are close to the ground truth value and makes the algorithm consistent, however, this also makes it static and not tunable. Looking at the topics generated, we would like to have more topics that are more distinct, but this is not possible due to the lack of input parameters that influence the result. It is possible to reduce the number of input topics and this will generate larger and more aggregate topics, but that does not line up with our goals.

The reasons for most of the problems with LDA are not in the algorithm but in the application. LDA is designed to be best when documents have a large number of words and multiple topics can be attributed to the same document. Our input data, however, has very low words per document and there are rarely repeated words. This results in unfocused and merged topics as the algorithm is not good at separating such sparse data. The problem of items not being categorized can also be attributed to this, as most of the items have low frequency and thus the algorithm has trouble figuring out the correlations between them.

Another major problem with LDA is that it does not make a distinction between parts, corrections, and symptoms, treating them all as words. This simplifies the process drastically, but perhaps too much as a lot of context is lost this way. For example, the majority of repair order jobs are associated with a single symptom and the relation between a symptom and the actual issue with the car can be thin as the symptoms are selected by the owners and can contain mistakes. On the other hand, the relationship between a correction code and the underlying issue (topic) is much stronger, as the technician selects the corrections and has to make sure that they solve the problem. This discrepancy is not an input to the LDA algorithm and thus cannot be used to generate better topics. The rigidity of the inputs also makes it difficult to adjust the topic generation towards more meaningful results. Overall LDA performs an alright job in this application, but it can be improved by considering the differences between Natural Language Processing and our problem domain.

3.2. GSDMM Results

The input data to GSDMM was the same as for the LDA algorithm, with the pre-processing described in Section 3.1.1. The number of initial topics was chosen to be 700 and Table 3.12 shows why no other experimentation was done regarding this.

GSDMM is an iterative algorithm that tends to converge to the optimal solution. The process can be, however, stopped at any step and the algorithm would generate an approximate solution. The run time of the algorithm on the machine (Lenovo 20Y4 - Intel i7 @2.5GHz, 8 Cores - 32GB of RAM) used to get the results was: 8 hours and 1 minute. Due to its heavy run-time, the algorithm was stopped after 10 iterations. The step-by-step statistics of the modifications and the run time can be seen in Table 3.12.

Step	Items moved	# of Clusters
0	652k	700
1	332k	700
2	177k	689
3	139k	342
4	123k	248
5	114k	235
6	109k	235
7	104k	226
8	100k	226
9	97k	222

Table 3.12: GSDMM run time

It is clear that the number of topics generated is much less than 700, so starting with a higher number of topics would only increase the already high run-time. Since we saw in the LDA section that 700 topics were not enough to describe the dataset as per our requirements, we expect the 222 topics generated by GSDMM to also be insufficient. Figure 3.4 shows the distribution of the number of items in each topic clustered by the GSDMM algorithm.

Example Topics Since GSDMM is a clustering algorithm, the value of φ has to be computed from the frequency of every item present in every cluster. In the following tables, the φ was substituted with **frequency** to be more clear about this relationship. In Table 3.13 and 3.14 the first two topics generated by the algorithm are shown.

Topic 0 from Table 3.13 seems to be related to various camera-related repairs. This topic is quite focused on this kind of repairs, but it still combines several individual repairs as can be seen by the relatively low

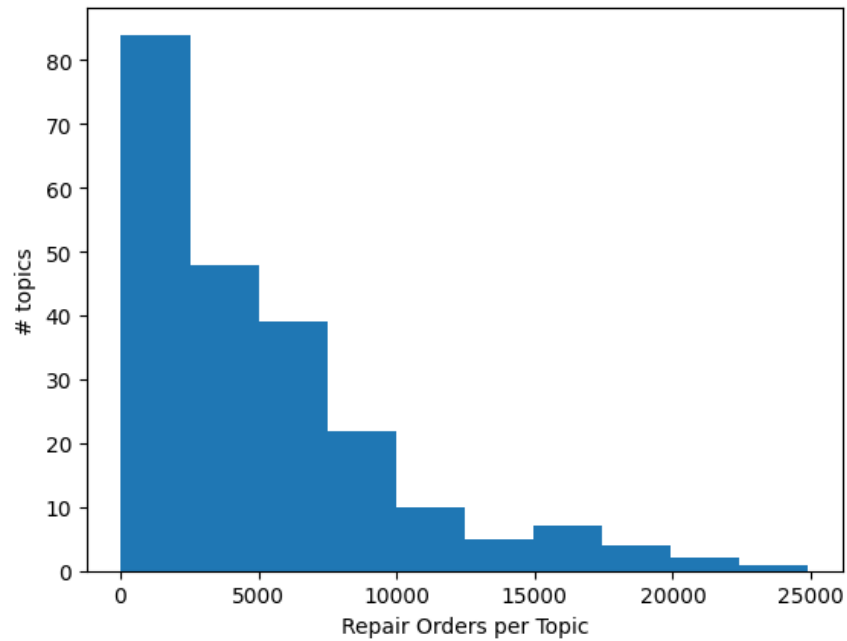


Figure 3.4: GSDMM items per topic distribution

Topic 0		
Item	Type	Frequency
Validation Test Drive	Correction	0.30
Camera Alerts	Symptom	0.24
Autopilot Camera Calibration	Symptom	0.21
Forward Camera Pitch Verification	Correction	0.19
Forward Camera (Remove & Replace)	Correction	0.14
Occupant Camera (Remove & Replace)	Correction	0.14
Driver Assistance Issues	Symptom	0.12
Low Voltage Circuit Check	Correction	0.08
Security Camera	Part	0.07

Table 3.13: GSDMM Topic 0

frequencies of all the items.

Topic 1 from Table 3.14 perfectly captures the 12V battery repair with a clear symptom, correction, and parts replaced.

Most Popular Topics In Table 3.15 and 3.16 the two most popular topics according to the GSDMM results are shown.

Topic 22 from Table 3.15 is grouping the majority (60%) of the "Service Bulletin" repair order jobs. The result

Topic 1		
Item	Type	Frequency
12V Battery Alert	Symptom	0.88
12V Battery (Remove & Replace)	Correction	0.88
12V Battery Model S	Part	0.78
12V Battery Model X	Part	0.14

Table 3.14: GSDMM Topic 1

Topic 22		
Item	Type	Frequency
Service Bulletin	Symptom	0.99
Power Strut O-ring	Part	0.30
Install Power Strut O-Ring	Correction	0.30
Inspect And Update Vehicle Software	Correction	0.20
Inspect Secondary Hood Latch	Correction	0.07
Inspect Vehicle Software Version	Correction	0.06

Table 3.15: GSDMM Topic 22

is analogous to Topic 0, a few similar, but different, repairs clustered together.

Topic 178		
Item	Type	Frequency
Exterior Adjustment - 0.6	Correction	0.35
Exterior Adjustment - 0.4	Correction	0.30
Door difficult to close	Symptom	0.13
Exterior Adjustment - 0.2	Correction	0.08
Door Handle Does Not Function	Symptom	0.05
Liftgate has poor alignment	Symptom	0.05

Table 3.16: GSDMM Topic 178

In Topic 178 from Table 3.16, a lot of different repairs related to door adjustments are combined together.

3.2.1. Conclusions

As the topics shown demonstrate, the GSDMM algorithm is working as intended by the authors, but it is not suitable for this application. The number of topics is too low to capture the individual repairs as intended. Additionally, the high run time and the lack of control over the topics make it hard to experiment with to improve these results.

The low number of topics mainly comes from Rule 1 of the Movie Group Project: the new table as more students. This rule tends to generate bigger clusters at the cost of precision. Additionally, GSDMM assumes one topic per document and, while this assumption may hold for most service visits, it probably does not hold for every single one. This could have caused some confusion in the clustering, combining topics that otherwise would have been separated.

The high run time of the algorithm is due to the chosen implementation used for this project. Due to the lower popularity of the GSDMM algorithm, few implementations are available. The one used is a pure Python implementation and does not make use of fast numerical libraries like *numpy* to do the computations.

Much like for the LDA algorithm, it is hard to influence the result by changing the inputs and the lack of tunable parameters does not help.

Since this algorithm seems to be good at clustering related repairs, it could be used as a pre-processing step to divide the repair order jobs into subgroups and then another algorithm could be used to extract the final topics. As interesting as this application might be, it was not experimented with in this project.

3.3. Combine and Split Results

The Combine and Split algorithm was applied to the same data as the LDA and GSDMM in the previous sections, with the same pre-processing step described in Section 3.1.1. The total run time of the algorithm was 3 min and 12s. The number of topics outputted by the algorithm was 2896.

Of these 2896 topics, 1929 were made up of single items. This means that only a third of the topics were made up of more than one item and that about 55% of all the items were not assigned to any topic. Table 3.17 shows the distribution over the different item types. As can be easily seen, the majority of the symptoms fall in the "Alone in a Topic" category and the total amount is dominated by correction codes.

Item Type	Total	Alone in a Topic	Percentage Alone in a Topic
Symptoms	420	358	85%
Parts	1053	376	36%
Correction Codes	1751	897	51%

Table 3.17: Items not associated to a topic

The other multi-item topics generated by this method were, of course, in line with the requirements as the rules wouldn't allow for groupings of items that are not correlated. Examples can be seen in Table 3.18, 3.19, and 3.20. All these topics represent clear relationships between a part and a correction code, this is exactly the objective of the algorithm. Most of the topics generated this way are pretty satisfactory.

Topic 1	
Item	Type
Primary Seal Front Left Door	Part
Primary Seal FL (Remove & Replace)	Correction

Table 3.18: C&S Topic 1

Topic 2	
Item	Type
Front Left Door Trim	Part
FL Door Trim (Remove & Replace)	Correction

Table 3.19: C&S Topic 2

Figure 3.5 shows the distribution of the number of items per topic, it can be seen that, apart from the single-item topics, the majority of the topics were made up of only 2 items. Most of the 2-item topics were the

Topic 3	
Item	Type
Security Camera	Part
Occupant Camera (Remove & Replace)	Correction

Table 3.20: C&S Topic 3

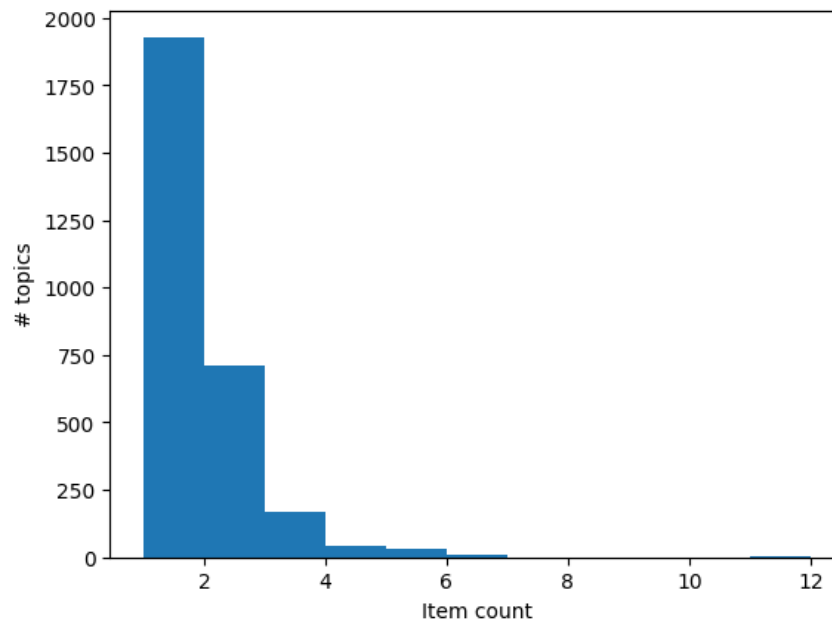


Figure 3.5: Item counts distribution

combination of a correction code and a part number. Additionally, Figure 3.6 shows the distribution of the topics' frequencies.

3.3.1. Conclusions

The algorithm did not perform as well as expected with the majority of the topics being single-items. The quality of the rest of the topics, however, shows that the approach of looking at the cross frequencies has some merit.

The reason for the high number of single-item topics is that the algorithm only combines items if it sees high cross frequencies between them. The symptoms, most of the time, are quite general and associated with a variety of different items and thus have a low correlation to most items. Similarly, but to a lesser degree, this is also true for some correction codes. These insights regarding the practical differences between corrections, parts, and symptoms, inspired the creation of the next algorithm.

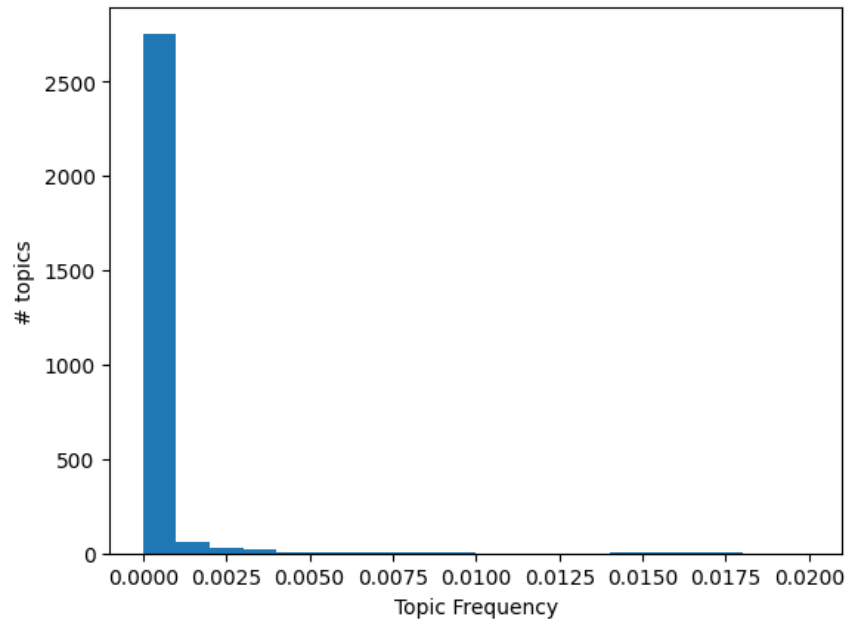


Figure 3.6: Topics' frequencies distribution

3.4. Correction Based Algorithm Results

Contrary to LDA and GSDMM, the Correction Based Algorithm was very efficient in the computations and didn't require any pre-processing of the data to reduce either memory footprint or computation time. The inputs was thus the full bag-of-words representation of the corpus without any filtering applied. The run time of the algorithm was (again on a Lenovo 20Y4 - Intel i7 @2.5GHz, 8 Cores - 32GB of RAM): 3 min 20s. The number of topics outputted by the algorithm was 4093.

Example Topics Due to the lack of pre-processing, the majority of the most frequent topics are made up of the high frequency, general correction codes such as "General Diagnosis" or "No Labor Performed" that were removed for other algorithms. Still, there are a couple of interesting topics out of the 10 most popular ones and they are shown in Table 3.21 and 3.22.

Topic 2912		
Item	Type	Frequency
Automated Tire Pressure Check	Correction	0.37
Tire Tread Depth Check	Correction	0.36
Courtesy Service	Symptom	0.26
Courtesy Inspection and top fluids	Symptom	0.01

Table 3.21: CBA Topic 2912

Figure 3.2a shows the item counts before any pre-processing is done, it is then expected that the majority of the topics will have very low frequencies. In fact, out of the 4093 topics, 2115 have less than 10 occurrences.

Topic 2641		
Item	Type	Frequency
Install Power Strut O-ring	Correction	0.33
Service Bulletin	Symptom	0.33
Power Strut O-ring	Part	0.33

Table 3.22: CBA Topic 2641

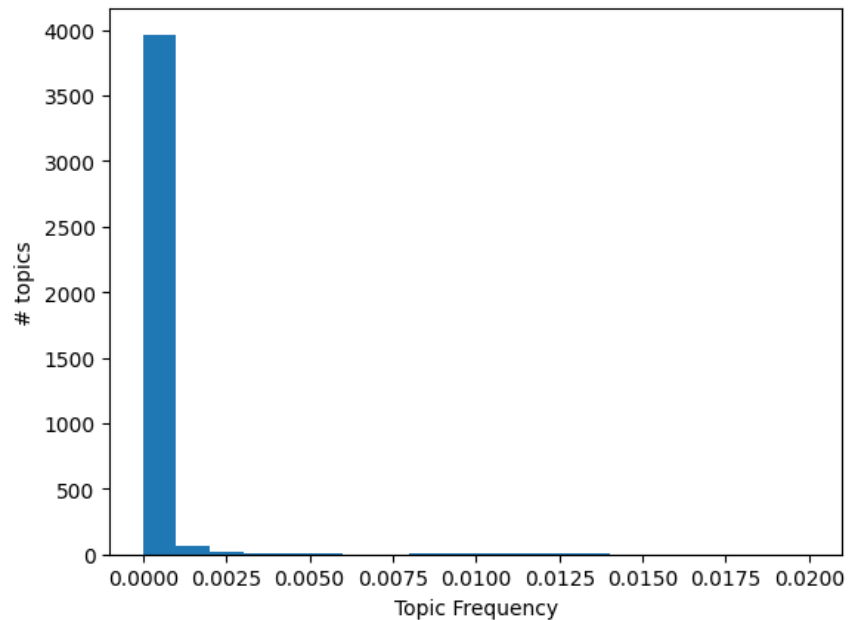


Figure 3.7: Topic's Frequencies Distribution

Figure 3.7 shows the distribution of the topics' frequencies. In Table 3.23, 3.24, 3.25, and 3.26 some examples of topics with medium frequencies among the first 100 are shown. All these topics seem reasonable and hold to further analysis of the data.

Topic 29		
Item	Type	Frequency
Replace all PT sensors	Correction	0.30
N/A	Symptom	0.24
PT Sensor, High Pressure	Part	0.23
PT Sensor, High Pressure	Part	0.23

Table 3.23: CBA Topic 26

The topics displayed in Table 3.27, 3.28, and 3.29 are less clearly defined and contain correction codes that don't have any other symptom or part associated with.

Topic 31		
Item	Type	Frequency
Coat Hook (Remove & Replace)	Correction	0.53
Model Y Coat Hook (a)	Part	0.25
Coat anger Issue	Symptom	0.09
Map light missing	Symptom	0.05
Model Y Coat Hook (b)	Part	0.02
Model Y Coat Hook (c)	Part	0.02
Model Y Coat Hook (d)	Part	0.02
Model 3 Coat Hook (a)	Part	0.01
Model 3 Coat Hook (b)	Part	0.01

Table 3.24: CBA Topic 31

Topic 96		
Item	Type	Frequency
Charge Port Door (Remove & Replace)	Correction	0.57
Motorized Charge Port Door	Part	0.23
Charge port doesn't open	Symptom	0.20

Table 3.25: CBA Topic 96

Topic 100		
Item	Type	Frequency
Replace VCM To Upgrade	Correction	0.34
Service Bulletin	Symptom	0.33
Tegra VCM	Part	0.33

Table 3.26: CBA Topic 100

Topic 102		
Item	Type	Frequency
HV Battery Isolation Inspection	Correction	1.0

Table 3.27: CBA Topic 102

Topic 104		
Item	Type	Frequency
Balance FR Tire	Correction	1.0

Table 3.28: CBA Topic 104

These correction codes are not often associated with any other parts or symptoms in any significant way. The frequency of the three examples is relatively low, so with more data it could be that new relations will emerge.

Topic 128		
Item	Type	Frequency
RL Brake Rotor (Remove & Replace)	Correction	1.0

Table 3.29: CBA Topic 128

3.4.1. Conclusions

The results show that this approach is promising for topic generation. The topics on both ends of the frequency spectrum could be improved, but that is a problem for all algorithms. The main strength of this Correction Based algorithm is the fact that it has many variables that can be tuned to output better results, something that is hard for the previously shown algorithms.

For the lower frequency topics, by definitions, there is less data to work with, and thus do not emerge properly in any algorithm. For the higher frequency topics and items, on the other hand, there is too much data and it gets mixed with a large variety of other topics and items thus obfuscating the ground true topics.

The benefits of the Correction Based algorithm compared to the ones shown previously are still valid as it eliminated some of their flaws. This algorithm does not require extensive pre-processing to run, it has fast computation time, and every item is categorized in at least one topic. This means that the pre-processing step could be used to improve the results by treating high-frequency and low-frequency items ad hoc, instead of being a required step to make the algorithm run at all as with LDA. Additionally, the algorithm has many more parameters that can be tweaked to achieve better performance, perhaps the most easily seen are the values of the thresholds that get decreased in the looping step.

4

Trend Analysis Results

In this chapter, the results of the trend analysis are presented. First, in Section 4.1 the results for the methods applied to a full month of service visits are shown. Then, in Section 4.2 the results for a subset of a year of service visits are shown. Finally, Section 4.2.1 draws some conclusions about the methods and their application to our data.

4.1. Results for one month of data

The weight function $w(x)$ defined in Section 2.3 can be chosen to be any function of x that is monotonically increasing until x_S . In [5], the following function is proposed:

$$w(x) = \exp\left(-\frac{(x - x_S)^2}{\rho}\right), \quad (4.1)$$

where ρ is a parameter that can be chosen and shapes the weight function as can be seen in Figure 4.1.

The dataset used for computing the novelty and growing scores was the same that was used for the LDA algorithm. The repair order jobs were organized based on the day they were open in order to compute the frequencies of each item over time. Since repairs are far more likely to be opened on a weekday, rather than during the weekend, a 7-day moving average was applied to the frequencies to smooth out this inconsistency.

In Figure 4.2 all of the topics have been plotted in a 2D graph showing on the x-axis the novelty scores and on the y-axis the growing scores. The size of the dot represents the frequency of the topic, a bigger dot corresponds to a higher frequency. The colors are there to help readability and are not tied to any quantity. The scores in the graph are taken from the last day of data. It can be seen that the majority of the topics have a novelty score of around 0, the growing scores, however, seem to be more spread out with one clump around 0 and another one around -0.001.

Figure 4.3, shows the data over time for frequency, word novelty, and growing scores for a few selected topics. Topic 640 has the highest novelty score, Topic 622 has the highest growing score, Topic 341 has a high novelty and high growing score, and finally, Topic 0 is the first in the sequence and was analyzed in Section 3.1. The values of the novelty scores and growing scores for the items in the first few days of data are not accurate as there are fewer data points to compare with. This is why the novelty scores at the beginning jump frequently between 0 and 1 for example.

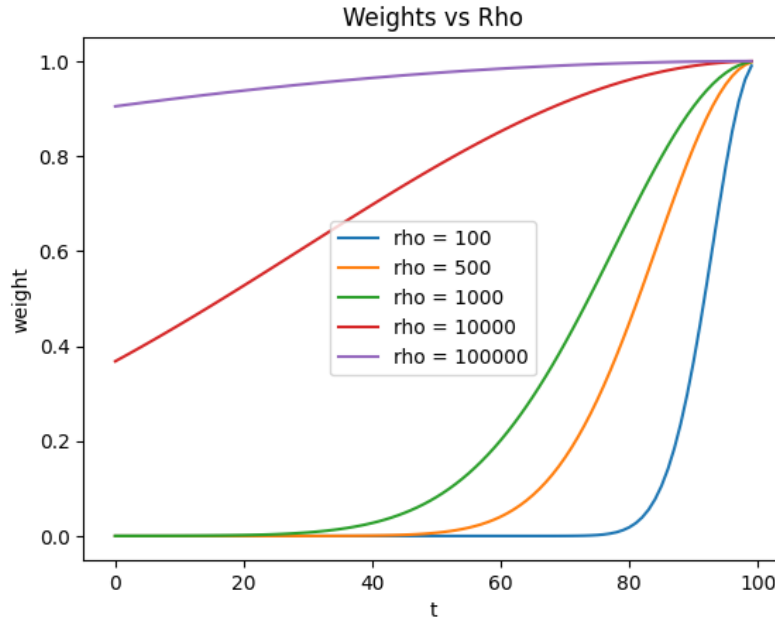


Figure 4.1: Shape of the weight function for different values of ρ

In the original application to microblogs, considering daily frequencies is a great way to find trending topics as the system reacts very quickly to external events. Service visits, on the other hand, are much more slow to react to the introduction of new issues or the scaling up of old ones. Mistakes introduced in the production line could take weeks if not months to show up in the field and failures due to environmental conditions (such as temperature) are as slow to change as the climate itself. With this in mind, it would be useful to see if the full procedure is suitable to apply to datasets that are significantly longer than a single month.

As explained in Section 3.1 the amount of data for a full month of repairs is already extremely high, collecting all the data for 6 or more months and processing it all together is not feasible. There are a few options to mitigate this, for example, it is possible to split the service visits, process them separately and then combine the results. The split can be done based on the individual months or by categorizing the service visits based on what items they contain. Alternatively, a subgroup can be selected and the rest ignored, provided that this subgroup is representative enough of the overall population.

4.2. Results for 1 year of data

To test the process on a longer time frame, data for a full year of "Service Bulletins" service visits was collected and analyzed. Service Bulletins are campaigns aimed at solving specific known issues. Only a small percentage of the service visits is due to Bulletins so it is possible to collect data for longer periods. Additionally, since every campaign is aimed at solving a specific issue, the data will consist of naturally coherent groups. Data was collected for all Service Bulletins from June 1st, 2022 until August 1st, 2023, resulting in 1500 unique correction codes, and 2,300 unique parts replaced.

The topics were generated with the LDA algorithm with $k = 500$. To reduce the volatility of the results, the repair order jobs were agglomerated by week instead of by day thus removing the need for a moving average. Figure 4.4 shows the novelty scores and growing scores for all the 500 topics, once again the size of the dots

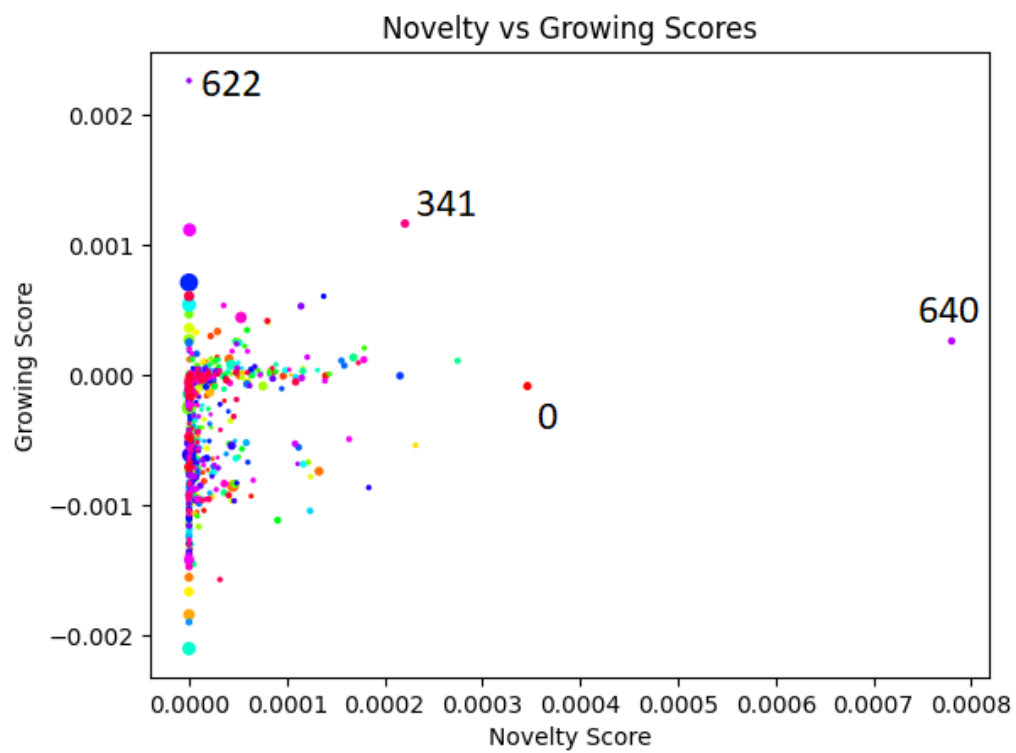


Figure 4.2: Novelty vs Growing scores

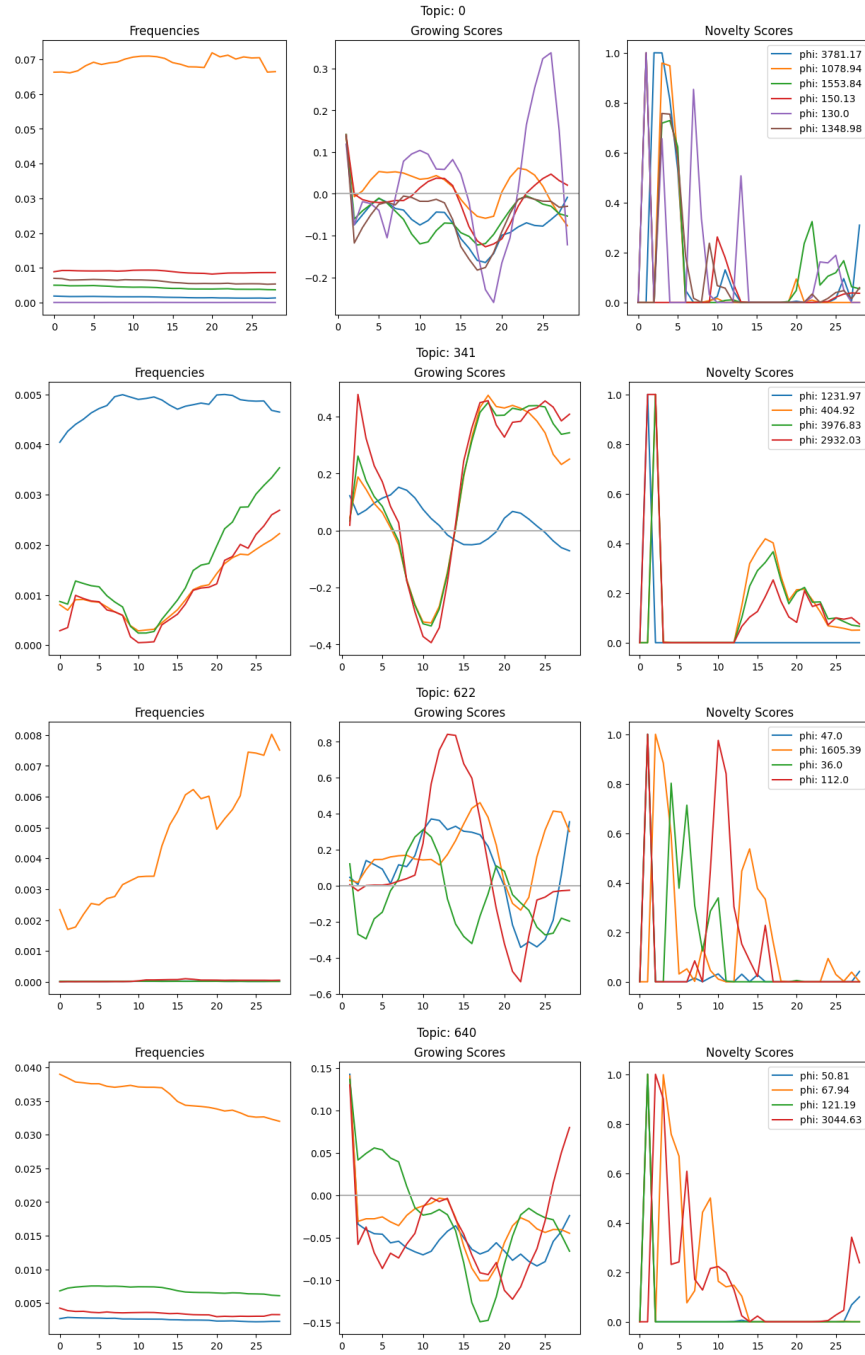


Figure 4.3: Time domain data for Topic 0 (Top), Topic 341 (Middle-Top), Topic 622 (Middle-Bottom), and Topic 640 (Bottom)

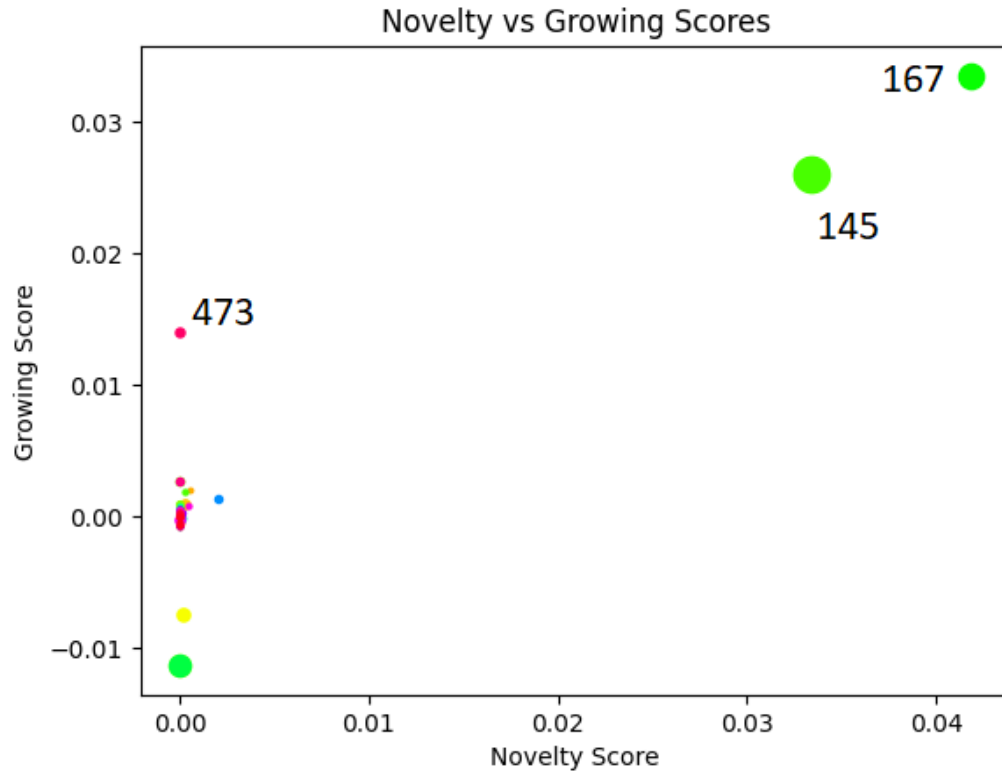


Figure 4.4: Novelty vs Growing scores for the Service Bulletins

represents the frequency of the topics and the colors are for clarity. The plotted scores refer to the last week of data.

Figure 4.5 shows the time data for some salient topics. Topic 167 and 145 are both novel and growing while being some of the most frequent, Topic 473 has a lower frequency, but still displays a high growing score. Due to the way the scores are computed, higher-frequency topics will be awarded higher scores if the shapes of the items' frequencies are the same. It is easy to see from the frequency plots of Figure 4.5 that all three topics have items with increasing frequencies in the last weeks. In particular, the frequencies of the items in Topic 473 have been growing since the 40th week and they seem to be plateauing or decreasing at the very end. This explains the high growing score and the close to 0 novelty score.

These graphs and scores show that extending the analysis to longer periods results in smoother curves and more interpretable results.

4.2.1. Conclusions

The approach of separating the growing score from the novelty score has proven to be effective since the scores seem to be independent. Overall, the results look promising and the trends seem to be captured well by the scores. The results for a 1 year of data, even though restricted to a specific type of repair, show almost impeccable correspondence with the frequency data. The methods for computing the scores are also very flexible and many parameters, such as the ρ and the horizon, can be changed to obtain more specific results.

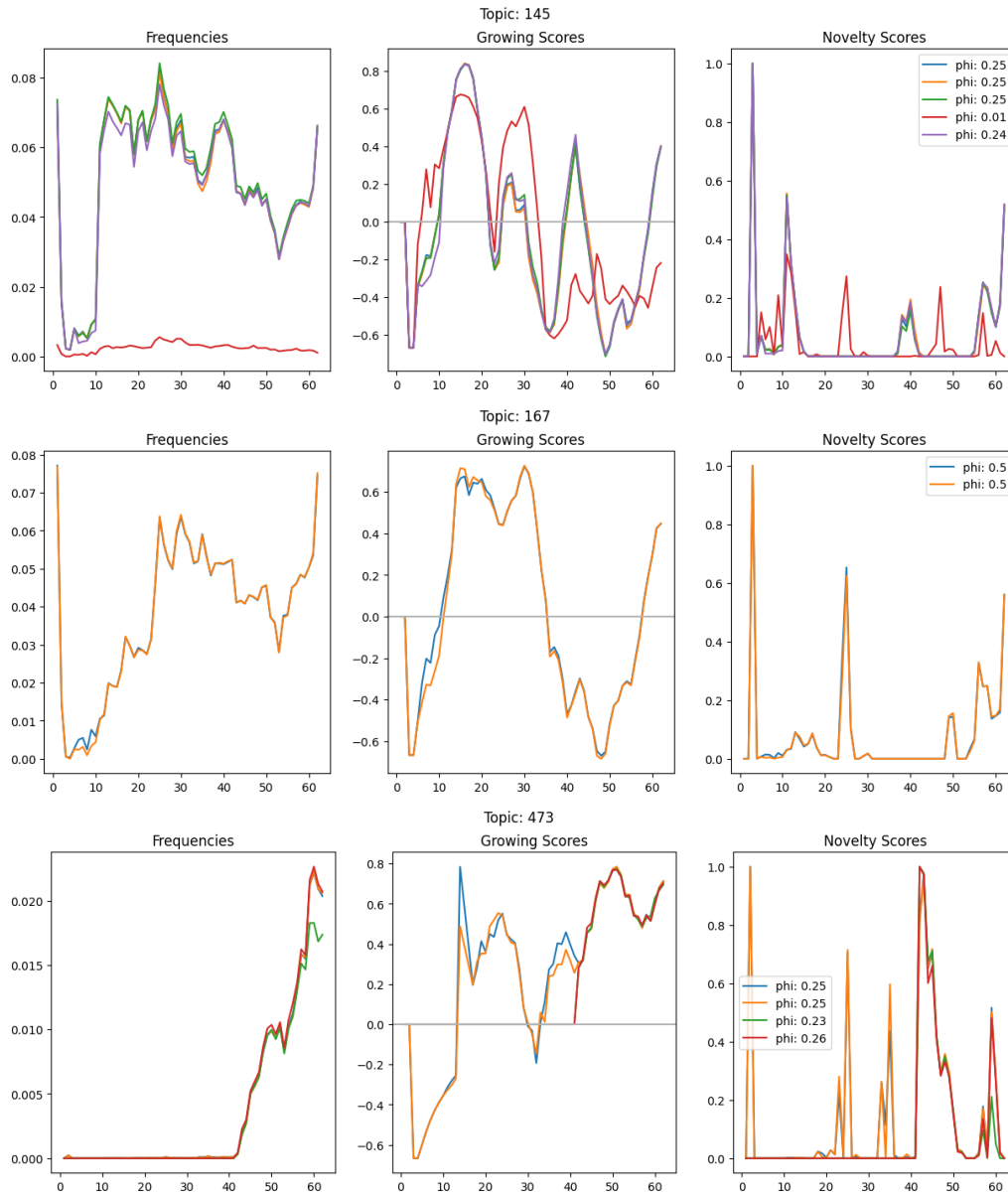


Figure 4.5: Time domain data for Topic 145 (Top), Topic 167 (Middle), and Topic 473 (Bottom)

As can be seen in both Figure 4.2 and 4.4, most novelty scores are very close to 0, while the growing scores are more spread out across the full spectrum. Additionally, there seems to be little correlation between the two scores, which justifies the separation of them into two different metrics.

The graphs in Figure 4.3 and 4.5 visually show that the correlation between the scores and frequencies. Looking at the frequencies of the items for all three topics shown in Figure 4.5, we expect high growing scores towards the end, since the frequencies are all increasing. And that is exactly what we see in Figure 4.4. Additionally, the novelty scores for Topic 145 and 167 should be high due to the sharp increase in the frequencies over the last few weeks. And indeed, that is also reflected in their novelty scores plotted in 4.4.

The computations of the scores have quite a few parameters that can be tuned to achieve better results. The ρ parameter in Eq. 4.1 controls the influence of the historic observations, a higher value of ρ pushes the weights closer to one making the LWLR less "local", while a lower value of ρ focuses the linear regression on the most recent observations. The other major parameter is the horizon S . In combination with ρ , S can be tuned to encompass more or less observations into the linear regression.

5

Conclusion

All the results put together can be seen in Figure 5.1, a screenshot of an interactive Tableau dashboard. The top left sheet shows the items present in a specific topic. The *Count* column indicates the number of times the item has been associated with the topic; the *Normalized* column shows the with which percentage the item has been associated with the topic; and the *Cost* column shows the cost contribution of the item to the parts cost of the topic.

The topic to be displayed in this sheet can be selected with the drop-down on the top or by selecting a data point from the other sheets. This makes it very convenient to analyze the data and check out the topics. The sheet in the middle left shows all the article IDs associated with the repair orders that showed the selected topic. Article IDs are the outputs of automated diagnostic tools.

The top right sheet is the combination of many metrics. The *Tesla* column shows the total cost to Tesla for the topic; the *Customer* column shows the total cost to the customer for the topic; and the *Cost* column shows the total cost for both customer and Tesla. The *S*, *X*, *3*, *Y* columns in red show the percentage of cars per model associated with the topic. Finally, the bottom sheet plots the different components contributing to the overall cost against the frequency of every topic. These graphs are computed also for the subset of repairs paid by the customer and the repairs paid by Tesla. The graphs can be swapped with the use of a toggle that is not visible in Figure 5.1.

Some information in this graphic has been censored for the publishing of this report. The values of the costs have been normalized to be in the range $[0, 1]$, and the same for all the quantities in the graphs in the bottom sheet. Additionally, exact part numbers and items have been hidden, only showing the systems and subsystems.

The dashboard described above is a good representation of the service costs and can be used to inspect the top items contributing to the total costs. Different metrics can be prioritized in different situations depending on the objective. For example, the people responsible for the supply chain could focus on topics with a high number of days in service and low FRT, this indicates that most likely the cars were waiting for parts to arrive. Something that is not shown in the dashboard above is that it is also possible to filter all the topics and keep only the ones related to specific systems or subsystems. This filter could be used by sub-departments to prioritize their work.

The dashboard shown in Figure 5.2 is meant to ease the use of the dashboard shown in Figure 5.1. Here, a specific item can be selected from a drop-down menu on the top right and all the information regarding the

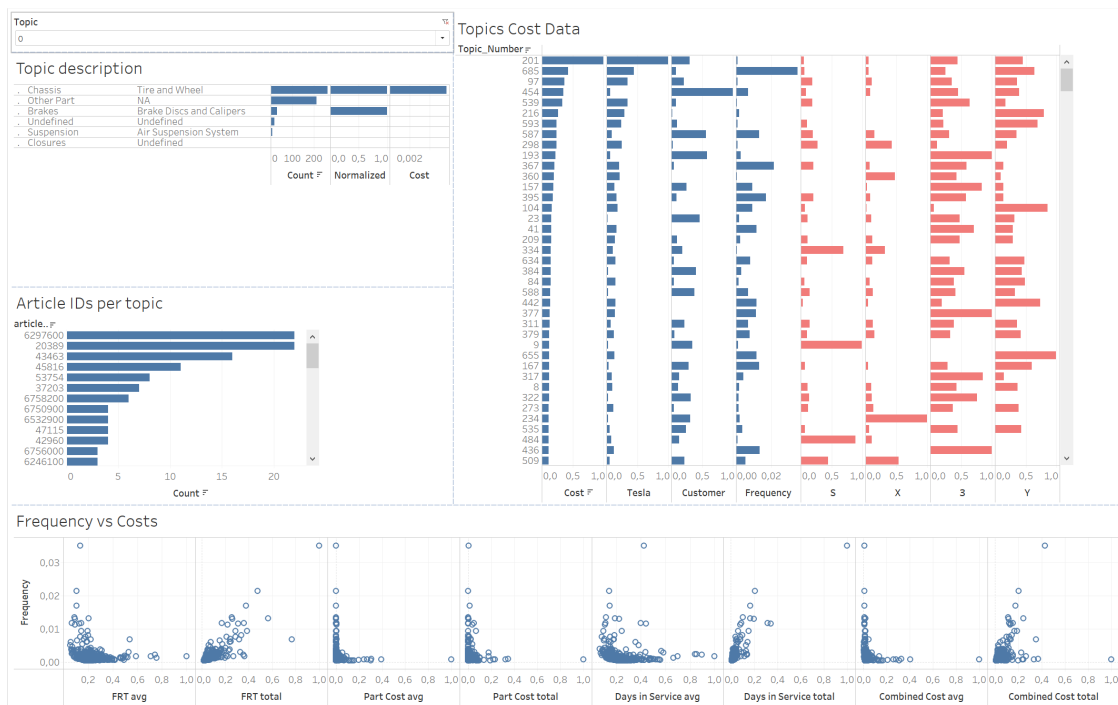


Figure 5.1: Tableau Dashboard showing most of the information computed in the previous sections combined

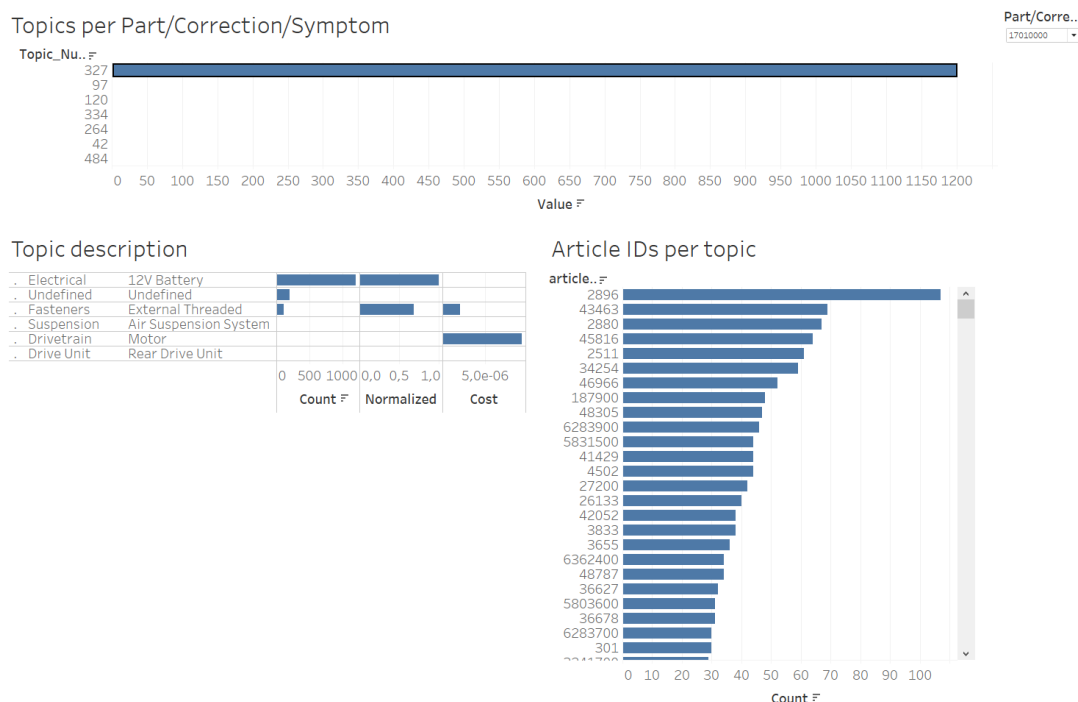


Figure 5.2: Tableau Dashboard showing the topics associated to a selected item with additional information for the articles

item is displayed. On the top is a list of all the topics associated with the item, ordered by highest value, while on the bottom is the combined description of those topics. This dashboard can be used to quickly assess specific items or groups of items to then compare with the other dashboards.

The final dashboard presented is possibly the most immediately useful. Figure 5.3 shows the average and total cost for every article together with their counts. This information is very valuable as the articles are very directly linked to specific issues. Having a clear sorted list of the most impactful articles to service can be easily acted upon by focusing efforts on reducing the issues linked to the top articles.

The data for this dashboard has been computed by aggregating parts, FRT, and days in service costs for every repair order. Then, all the costs for the repair orders were divided among the articles associated with it. The costs and counts displayed here are once again normalized to be in the range $[0, 1]$ so as to not show any sensitive information.

The goal of this project was to generate an actionable list of the top cost-impacting items for Service Engineering and create a system to identify new and growing issues in the field.

Firstly, a way to separate costs by repair type or issue needed to be implemented, and in this project, it was done through the use of topic generation algorithm borrowed from Natural Language Processing. Two known topic generation algorithms were tested, Latent Dirichlet Allocation and Gibbs Sampling for the Dirichlet Multivariate Mixture Model, and were found to be not satisfactory for our application. Then two more algorithms designed by the author, the Combine and Split algorithm and the Correction Based algorithm, were introduced and analyzed. The Correction Based Algorithm was found to produce the most satisfactory results out of all the algorithms tested.

Articles Cost Data

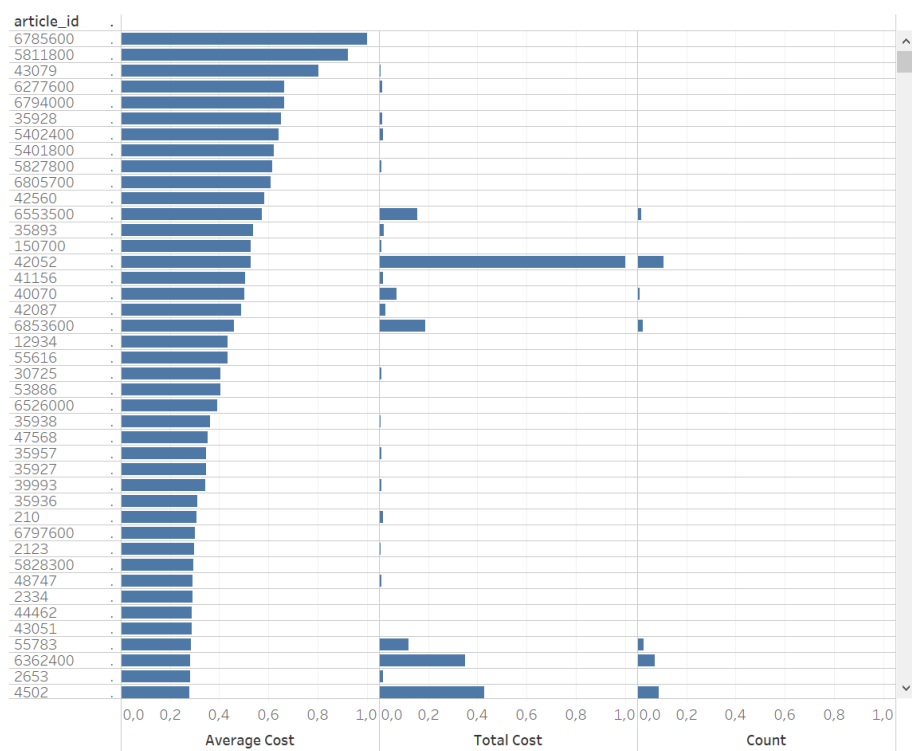


Figure 5.3: Tableau Dashboard showing the average and total cost for each article

Secondly, a way to access the data produced was needed. In particular, the following data points were the focus:

- contribution to the cost of a service visit: parts costs, flat rate time, and days in service;
- the division of costs between the customer and Tesla;
- the data of the car configurations;
- distribution of article IDs.

This was achieved with the use of the Tableau software through interactive dashboards such as the ones displayed in Figures 5.1, 5.2, and 5.3.

Finally, to identify trends in the repairs, a novel approach of computing two scores: a novelty score and a growing score, was introduced. It was shown in Section 4 that the approach produces results that are in line with the expectations from the data and it is viable for identifying new and growing issues.

Overall, the approach of using methods from Natural Language Processing in a completely different context has proven to generate mixed results. On the one hand, the ideas and the structure of the analysis proved to be useful, but on the other hand, many adjustments to the algorithms needed to be made to achieve decent results. This is not surprising as the structure of the data used for this project is very different from text-based data.

5.1. Discussion

While the model and the visualization are useful, they can be improved to generate a more actionable list of issues. At the moment, the main information used to derive possible improvements comes from looking at the list of article IDs associated with the topics or at the configurations of the cars. A better interface for navigating configuration information should be worked on. The display of the *Model* in separate columns is good, but it is not scalable for multiple configurations with a lot of options for each.

Another way to gather actionable information is to look at signals and alerts. Data can be collected in the hours/days/weeks before a failure to get more context. Some general signals are almost always relevant, such as km driven, ambient temperature, and vehicle speed, but more specific signals can be queried depending on the type of repair performed. This data could be used to further separate the cars within a topic and recognize trends.

Of course, the topic generation can also be improved. Here are a few points that can be looked into to improve the Correction Based Algorithm:

- Some generic and frequent correction codes absorb a large number of items. This results in massive and unfocused topics. Topics based on very frequent correction codes could be penalized for adding new items. These correction codes could be marked by hand or with some heuristics to produce better topics.
- The limits chosen for the threshold t could be optimized generally or chosen depending on the input dataset.
- More analysis should be done to attribute more fairly the cost of the days in service and FRT. At the moment it is divided equally between the topics for each repair order. However, some topics require more time and should take more of the costs.

In general, to improve the utility of the project, iterating with the users should be a priority to see what sort of information would be the most useful to collect and display. More dashboards could be made to improve the readability of the data or make the results easier to search or sort. In particular, a new dashboard should be

made to organize the novelty and growing scores in either 2D form such as Figure 4.4 or in a sorted list as in the top right sheet of Figure 5.1.

References

1. *Tesla Vehicle Production and Deliveries and Date for Financial Results and Webcast for Fourth Quarter 2022* <https://ir.tesla.com/press-release/tesla-vehicle-production-deliveries-and-date-financial-results-webcast-fourth-quarter>.
2. *Tesla Manufacturing* <https://www.tesla.com/manufacturing>.
3. *Tesla achieves its goal of 5,000 Model Ys a week at Giga Texas* <https://electrek.co/2023/05/09/tesla-achieves-goal-5000-model-y-week-giga-texas/>.
4. *Tesla Gigafactory Berlin finally reaches goal of 5,000 electric cars a week* <https://electrek.co/2023/03/25/tesla-gigafactory-berlin-goal-5000-electric-cars-week/>.
5. Huang, J. *et al.* A probabilistic method for emerging topic tracking in Microblog stream. *World Wide Web* **20**, 325–350 (2017).
6. Sumait, L. A., Barbar, D. & Domeniconi, C. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. *ICDM*, 3–12 (2008).
7. Lau, J. H., Collier, N. & Baldwin, T. On-line Trend Analysis with Topic Models: #twitter Trends Detection Topic Model Online. *Proceedings of COLING*, 1519–1534 (2012).
8. Chen, Y., Amiri, H., Li, Z. & Chua, T. Emerging topic detection for organizations from microblogs. *SIGIR*, 43–52 (2013).
9. Schubert, E., Weiler, M. & Kriegel, H. SigniTrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. *SIGKDD*, 871–880 (2014).
10. Huan, J., Peng, M. & Wang, H. Topic Detection from Large Scale of Microblog Stream with High Utility Pattern Clustering. *Proceedings of the 8th Workshop on Ph. D. Workshop in CIKM*, 3–10 (2015).
11. Blei, D. M., Ng, A. Y. & Jordan, M. I. Latent Dirichlet Allocation. *Journal of Machine Learning Research* **3**, 993–1022 (2003).
12. Yin, J. & Wang, J. *A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering* in (2014), 233–242.
13. Liu, T., Zhang, N. & Chen, P. *Hierarchical Latent Tree Analysis for Topic Detection in Machine Learning and Knowledge Discovery in Databases* (2014), 256–272.
14. Hofmann, T. *Probabilistic latent semantic indexing* in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), 50–57.
15. Aldous, D. Exchangeability and related topics. *Ecole d’été de probabilités de Saint-Flour* **13**, 1–198 (1985).
16. Pritchard, J. K., Stephens, M. & Donnelly, P. Inference of population structure using multilocus genotype data. *Genetics* **155**, 945–959 (2000).

17. Katti, S. K. & Rao, A. V. Handbook of The Poisson Distribution. *Technometrics* **10**, 412–412 (1968).
18. Poisson, S. D. Probabilité des jugements en matière criminelle et en matière civile, précédées des règles générales du calcul des probabilités. *Paris, France: Bachelier* **1** (1837).
19. Wilks, S. *Mathematical Statistics* (Read Books, 2008).
20. Kotz, S., Balakrishnan, N. & Johnson, N. L. *Continuous multivariate distributions, Volume 1: Models and applications* (John Wiley & Sons, 2004).
21. Evans, M., Hastings, N. & Peacock, B. *Statistical Distributions* 134–136 (Wiley, 2000).
22. Gelman, A., Carlin, J., Stern, H. & Rubin, D. *Bayesian data analysis* (Chapman and Hall/CRC, 1995).
23. Kass, R. & Steffey, D. Approximate Bayesian inference in conditionally independent hierarchical models (parametric empirical Bayes models). *Journal of the American Statistical Association* **84**, 717–726 (1989).
24. Morris, C. Parametric empirical Bayes inference: Theory and applications. *Journal of the American Statistical Association* **78**, 47–65 (1983).
25. Dickey, J. Multiple hypergeometric functions: Probabilistic interpretations and statistical uses. *Journal of the American Statistical Association* **78**, 628–637 (1983).
26. *scikit-learn* <https://scikit-learn.org/stable/index.html>.
27. Hoffman, M., Bach, F. & Blei, D. Online learning for latent dirichlet allocation. *advances in neural information processing systems* **23** (2010).
28. Hoffman, M. D., Blei, D. M., Wang, C. & Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research* (2013).
29. Jordan, M., Ghahramani, Z., Jaakkola, T. & Saul, L. Introduction to variational methods for graphical models. *Machine Learning* **37**, 183–233 (1999).
30. Ronning, G. Maximum likelihood estimation of Dirichlet distributions. *Journal of Statistical Computation and Simulation* **34**, 215–221 (1989).
31. Minka, T. *Estimating a Dirichlet distribution* 2000.
32. Berkhin, P. in *Grouping multidimensional data: Recent advances in clustering* 25–71 (Springer, 2006).
33. Rosenberg, A. & Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. *EMNLP-CoNL*, 410–420 (2007).
34. Jain, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters* **31**, 651–666 (2010).
35. Salton, G., Wong, A. & Yang, C. S. A vector space model for automatic indexing. *Communications of the ACM* **18**, 613–620 (1975).
36. Nigam, K., McCallum, A., Thrun, S. & Mitchell, T. M. Text classification from labeled and unlabeled documents using em. *Machine Learning* **39**, 103–134 (2000).
37. McLachlan, G. J. & Basford, K. E. *Mixture models: Inference and applications to clustering* (M. Dekker New York, 1988).
38. Bishop, C. & Nasrabadi, N. *Pattern recognition and machine learning* **4** (Springer, 2006).

39. Cleveland, W. S. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association* **74**, 829–836 (1979).
40. Weng, J. & Lee, B. Event detection in Twitter. *ICWSM*, 401–408 (2011).
41. Li, C., Sun, A. & Datta, A. Twevent: segment-based event detection from tweets. *CIKM*, 155–164 (2012).
42. Diao, Q., Jiang, J., Zhu, F. & Lim, E. Finding bursty topics from microblogs. *ACL*, 536–544 (2012).
43. Yan, X., Guo, J., Lan, Y., Xu, J. & Cheng, X. A probabilistic model for bursty topic discovery in microblogs. *AAI Conference on Artificial Intelligence*, 353–359 (2015).
44. Tableau <https://www.tableau.com/>.
45. Codd, E. F. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* **13**, 377–387 (1970).
46. jupyter <https://jupyter.org/>.
47. pandas <https://pandas.pydata.org/>.
48. NumPy <https://numpy.org/>.
49. Matplotlib <https://matplotlib.org/>.
50. Schofield, A., M. Magnusson, Thompson, L. & Mimno, D. *Understanding text pre-processing for latent Dirichlet allocation in Proceedings of the 15th conference of the European chapter of the Association for Computational Linguistics* **2** (2017), 432–436.
51. Huang, L., Ma, J. & Chen, C. Topic Detection from Microblogs Using T-LDA and Perplexity. *APSECW*, 71–77 (2015).
52. Gan, J. & Qi, Y. Selection of the optimal number of topics for LDA topic model—taking patent policy analysis as an example. *Entropy* **23**, 1301 (2021).