Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Explaining NLP Model Predictions for Fact-Checking Pipeline

Eliška Kopecká

Supervisor: Ing. Jan Drchal, Ph.D.
Field of study: Open Informatics
Subfield: Data Science
May 2022

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kopecká Eliška**	Personal ID number:	**491858**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Specialisation:	**Data Science**		

II. Master’s thesis details

Master’s thesis title in English:

Explaining NLP Model Predictions for Fact-Checking Pipeline

Master’s thesis title in Czech:

Vysvětlování výstupu modelů zpracování přirozeného jazyka pro úlohu ověřování faktů

Guidelines:

The task is to compare methods for explaining model predictions focusing on NLP, more precisely on models used in Czech language fact-checking pipelines, i.e., document retrieval and natural language inference.
1) Familiarize yourself with the fact-checking pipeline and state-of-the-art methods for explaining machine learning model predictions. Explore algorithms specialized for the NLP domain.
2) Work with the Czech Wiki FEVER and ČTK datasets, both supplied by the supervisor.
3) Select and evaluate appropriate methods using standard metrics. Analyze the algorithms also from the user's point of view. For example, experiment with visualizations of important words (sentences or other textual features) as designated by the algorithm and their ability to aid human fact-checkers.

Bibliography / sources:

[1] Thorne, James, et al. "FEVER: a large-scale dataset for fact extraction and verification." arXiv preprint arXiv:1803.05355 (2018).
[2] Thorne, James, et al. "The fact extraction and verification (fever) shared task." arXiv preprint arXiv:1811.10971 (2018).
[3] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why should I trust you? Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.
[4] Lundberg, Scott, and Su-In Lee. "A unified approach to interpreting model predictions." arXiv preprint arXiv:1705.07874 (2017).

Name and workplace of master’s thesis supervisor:

Ing. Jan Drchal, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **27.07.2021** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **19.02.2023**

Ing. Jan Drchal, Ph.D.	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

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

.	
Date of assignment receipt	Student’s signature

Acknowledgements

First, I want to thank my supervisor Jan Drchal for his guidance, for his kind approach and for giving me the freedom and space to explore and work on my own while always finding time to answer my questions and providing guidance when needed.

Next, I want to thank my annotators for making the experiments possible.

Finally, I want to use this opportunity to thank my family and friends for their unwavering support not only throughout this thesis but throughout my whole studies and life.

Specifically, I want to mention Radek Kopecký, Marie and Břetislav Kopečtí, Zuzana Chalupová, Prokop Černý, *soon to be* Bára and Ondřej Placzkovi, Lucie Tanistrová and especially Petr Houška.

Thank you for finding the balance between calm support that helped me sleep at night and gentle panic that kept me going by day.

Declaration

I hereby declare that I have developed the presented thesis independently and I have cited all literature, publication and other sources of information.

In Prague, 20. May 2022

Abstract

This thesis explores interpretability methods and the possibilities of their application to natural language processing (NLP) models used within a fact-checking pipeline. More specifically, it focuses on the application of two local, model-agnostic interpretability methods LIME and SHAP to natural language inference (NLI) models used to infer a veracity label from a claim and a context.

In this work, we modify and apply SHAP and LIME interpretability methods to the NLI models and develop a text-augmented version for LIME. Later, we test various parameter settings to find the optimal parametrization for each method which we then compare in a binary forced-choice experiment with human-grounded evaluation. For both datasets used within the project, SHAP is evaluated to produce more helpful explanations.

Keywords: NLP, Interpretability, Explainability, LIME, SHAP

Supervisor: Ing. Jan Drchal, Ph.D.

Abstrakt

Tato práce zkoumá interpretační metody a jejich aplikovatelnost na modely pro zpracovávání přirozeného jazyka (NLP) použitých v rámci úlohy ověřování faktů. Konkrétněji se zaměřuje na aplikaci dvou lokálních, modelově agnostických interpretačních metod LIME a SHAP na modely pro inferenci přirozeného jazyka (NLI). Cílem těchto NLI modelů je odvození pravdivostní hodnoty výroku z kontextu.

Metody LIME a SHAP upravujeme a poté aplikujeme na NLI modely. Rovněž navrhujeme a implementujeme vlastní verzi LIME, rozšířenou o generování nových vstupů na základě podobnosti textu. Následně tyto interpretační metody testujeme s různým nastavením parametrů, abychom našli nejvhodnější parametrizaci pro každou z nich. Poté metody s nejvhodnější parametrizací porovnáváme proti sobě v binárním experimentu s nucenou volbou, pro něž využíváme lidské hodnocení. Pro obě, v práci použité, datové sady SHAP vytváří užitečnější vysvětlení.

Klíčová slova: NLP, Interpretovatelnost, LIME, SHAP

Překlad názvu: Vysvětlování výstupu modelů zpracování přirozeného jazyka pro úlohu ověřování faktů

Contents

Figures

Tables

Chapter 1

Introduction

The decisions we make are based on the information we have and it is therefore not at all surprising that many political and industrial subjects have long lived by the credo *"If you can control information, you can control people."*[1] Fake news is not a new phenomenon, it has been used as a form of political and industrial propaganda for centuries (OW19). However, with the rise of internet and social media its speed of spreading and effect multiplied (Bur17), (BB19).

As a result, we live in the age of misinformation, which floods our public space.

A lifebuoy comes in the form of high-quality journalism and well-resourced articles which bring verified information to the public discourse. But the verification process, essential for quality journalism, is time-consuming and mentally demanding.

Therefore, we build on recent experiments of automated fact-checking (TVCM18) and bring a tool which, using state-of-the-art natural language processing (NLP) models, helps journalists verify information.

But even the best models are sometimes wrong. Therefore our tool is not meant to be an omniscient black box. Instead, it should guide the user through the quanta of text to the vital information.

And that is where the contribution of this thesis lies. We will research different explainability methods, explore how they could be applied to interpret NLP models and test which of them are most suitable to to ease the work of human fact-checkers.

[1]Quote by American novelist Tom Clancy (Sch95)

1.1 Fact-checking

Fact-checking refers to the assessment of the truthfulness of a claim (VR14). It has traditionally been linked with journalism. As argued in the Introduction, it is now more needed than ever.

There is a range of projects, which focus on the fact-checking activity, e.g. PolitiFact[2], FactCheck.org[3] or The Washington Post Fact Checker[4]. In the Czech Republic, there is a successful project Demagog.cz[5].

However, all these projects perform manual fact-checking - a process in which a person (usually a journalist) manually verifies claims. And manual fact-checking is a time-consuming and tedious process. Moreover, it is expensive and hard to scale because it cannot keep up with the pace, at which news is created and spread online.

Therefore, in recent years, automated fact-checking has been gaining an increased amount of attention(ZAZ21). Full or partial automation of the fact-checking process would be very beneficial for the journalism community. It would also broaden the horizons of fact-checking use e.g. to real-time verification during political debates.

There are multiple automated fact-checking projects, which provide a different definition of the task and present various pipeline designs (ZAZ21). This work is a part of a Czech fact-checking project, which was based on pipeline from (TVCM18) and presented in (DUR$^+$22). We are going to use datasets and explain models described in (DUR$^+$22) and therefore we are going to consider the fact-checking pipeline used in the project.

1.1.1 Fact-checking pipeline

The fact-checking pipeline (also verification pipeline) in (DUR$^+$22), consists of two stages - document retrieval (DR) and natural language inference (NLI) and is visualized in fig. 1.1

In the beginning, we have a *claim* – a statement to verify, and a *ground-truth corpus* - database of verified texts, which we accept as the ground truth.

[2]https://www.politifact.com
[3]https://www.factcheck.org
[4]https://www.washingtonpost.com/news/fact-checker
[5]https://demagog.cz

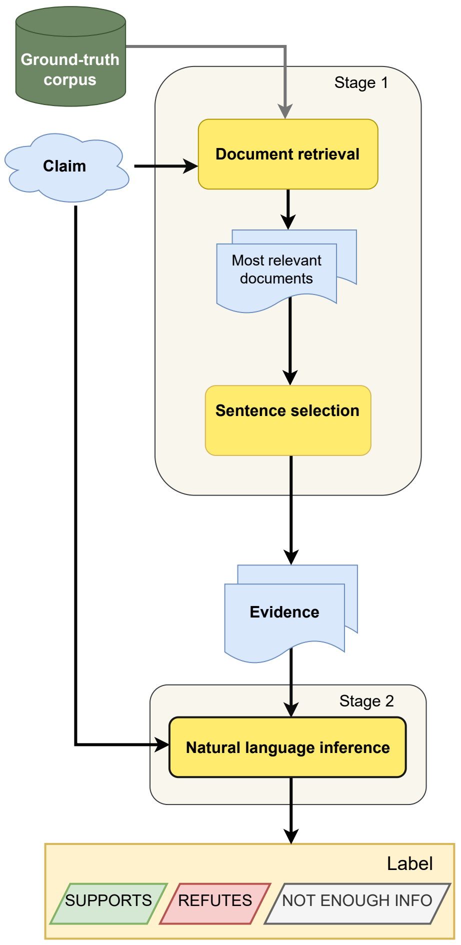


Figure 1.1: Fact-checking pipeline

- In the first stage, the DR component selects documents from the ground-truth corpus which are most relevant (semantically closest) to the claim. From these documents, the most relevant sentences are selected and formed as *evidence*. In (TVCM18), sentence selection is defined as a separate stage.
  DR is typically implemented using traditional approach of numerical methods (e.g. TF-IDF weighting scheme), or neural approach, which currently means using large-scale Transformer architecture.
- In the second stage, the NLI model matches the claim against the evidence and determines whether the claim is supported, refuted or whether there is not enough evidence to decide.
  NLI is also implemented using Transformer architecture and is further described in section 3.3.

1.1.2 Models

Throughout this work, we shall be focusing mainly on interpretability methods and so our approach to the explained models will be pragmatic. Therefore, unless necessary for the explainability method, we will not be describing the architecture in detail, parameter settings, or inner workings of the model.

However, as can already be seen in the fact-checking pipeline description, the fact-checking project heavily relies on Transformer architecture. And since it is such a crucial component of the project and every model which we are going to interpret is based on it, we are now going to devote a short section to the Transformers as well as to the training process used to train all the models that we are going to encounter.

Transformer architecture

The Transformer architecture was introduced in (VSP$^+$17) and in the 5 years since its release, it has dominated nearly every Natural Language Processing task.

It is a neural network architecture originally invented for sequence-to-sequence tasks which, due to better parallelization, outperformed at that time popular recurrent neural networks(RNN). Because unlike RNNs, Transformers do not need to process the data in order. Instead of recurrence, they use solely Attention, which provides arbitrary context for a token at any position in the input sequence.

Attention. *Attention* is the key mechanism of Transformers. Its main task is to handle long-term dependencies and provide context. For example in the sentence "The Law will never be perfect, but its application should be", attention links word "its" with word "Law". (example from Appendix of (VSP$^+$17))

Architecture. Transformers use Encoder-Decoder architecture (previously known from RNNs), containing an encoder unit and a decoder unit. Each unit consists of N repeatedly stacked identical blocks - encoder block and decoder block. The whole architecture can be seen in fig. 1.2.

BERT. Transformers enabled the creation of many popular models. One such famous example is BERT - Bidirectional Encoder Representations from Transformers (DCLT18) which is a stack of Transformer encoders pre-trained on a large corpus of unlabelled text (including entire Wikipedia).

BERT and countless number of its variations are accessible through a library called *Hugging Face Transformers*[6] (WDS$^+$20) as pre-trained models, which can be retrained at relatively small costs to yield admirable results for various NLP tasks. This retraining process is called transfer learning.

Transfer learning

Transfer learning means applying the knowledge of an already trained model to a different but related task. In practice, we distinguish two training stages - pre-training and fine-tuning.

Pre-training. Pre-training is a computationally expensive process during which the network is trained for a generic task on large corpora. (E.g. training BERT from scratch.)

[6]`https://huggingface.co/docs/transformers/index`

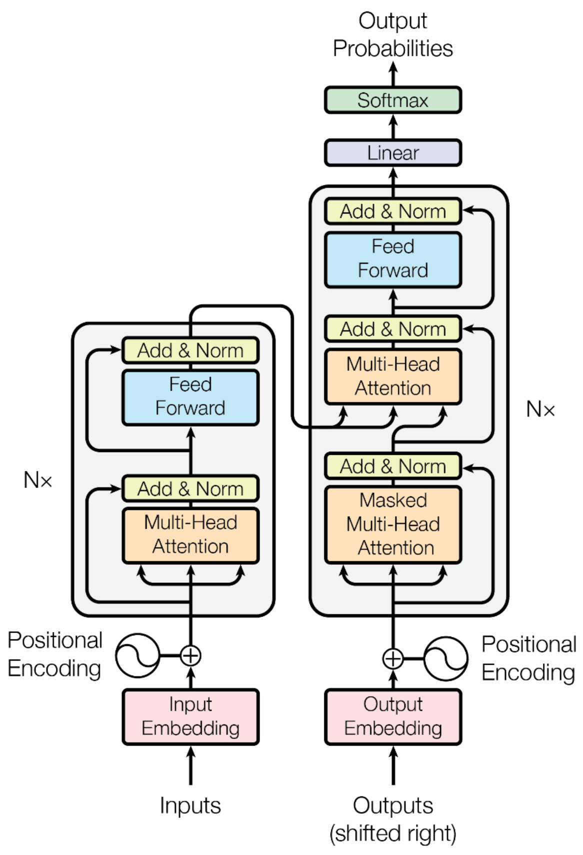


Figure 1.2: Transformer model architecture. Figure from (VSP+17)

Fine-tuning. Fine-tuning means taking the pre-trained model with its learnt weights and retraining it for the specific task which we want to use it for with a new, usually smaller dataset. (E.g. retraining BERT to fit Czech-English translation.)

If the task's output varies from what the original model was trained to return or if the new dataset is very small, instead of retraining the whole model, we can retrain only its final layers. We add or replace the final layers of the original model and retrain those.

1.2 Thesis outline

- **Chapter 1** states the motivation for this work and provides a broader context in terms of the project, which this work is a part of.
- In **chapter 2** we define the concept of interpretability and introduce different interpretability techniques and interpretability assessment methods.

- In **chapter 3** we define the problem, that we trying to solve. First, we analyze, where best to apply the interpretability methods, then introduce the chosen task and describe the models and datasets that we work with.
- **Chapter 4** presents the solution to the problem - the interpretability methods, their parametrization, implementation details and results. In this chapter, we pre-select the methods and parameter settings which produce the most helpful explanations.
- In **chapter 5** we compare and statistically evaluate the best methods and parameter settings from chapter 4.
- Finally, **chapter 6** concludes the thesis and provides a brief summary of the work done and the results achieved.

Chapter 2

Background

In this chapter, we shall introduce the concept of interpretability, its taxonomy, as well as selected applicable methods. Then, we will move to an overview of current evaluation measures. This chapter is designed, such that it provides all the theoretical background that will be used in the search for a suitable solution.

2.1 Terminology

In literature, the terms *explainability* and *interpretability* are frequently either used interchangeably (CPC19) or the authors only use one of the terms and disregards the other (DVK17). However, there are also sources which recognize or even describe the differences and because it is the main subject of this work, we will differentiate between them also.

Before we clearly define the vocabulary used in this work, let us remark, that there is not an agreement within the machine learning community on the terminology (MV20) and there are several different definitions of *interpretability* and/or *explainability* - e.g. (Lip16), (DVK17) or (Rud19) are worth mentioning. In this work, we will use terminology from (Lip16), whose description of nuances between the terms is well suited for this thesis. Let us now clarify the vocabulary used in this work.

The term *interpretability* (in some papers synonymous with *understandability* and *comprehensibility* (LCGH13)) refers to the degree to which cause and effect can be observed within the system. In other words, it is describing what effect changes in parameters or input will have on a system.

On the other hand, the term *explainability* means the extent to which the internal mechanics of a machine or deep learning system can be explained in human terms. To rephrase it, it tries to explain the inner workings of the model such that it can be understood by a non-expert.

One way of looking at it would be, that interpretability is enabled through explainability.

The goal of this work is to explain the model output with respect to the input. We are not trying to explain the inner mechanics of the model but rather to show which exact values in the input resulted in the particular model outcome. Therefore in this project, we are interested in interpretability.

2.2 Classification of interpretability

Interpretability can be described and categorized based on many different criteria - described for instance in (DVK17) or (CPC19). In different sources, there are different categories, we selected the most frequently used ones, which are also used later in this chapter to describe individual explanation techniques.

- **Intrinsic vs post-hoc** Intrinsic, or sometimes also inherent, interpretability is achieved through a lack of complexity in the model structure. Intrinsic interpretability can be attained before model training because the models are self-explanatory. For instance, sparse linear models or decision trees belong to this category. Throughout the work, we will refer to these self-explanatory, inherently intuitive models as interpretable. Post-hoc interpretability, on the other hand, refers to using an explainability method (a tool) after the model training, which provides us with an explanation of the model or interpretation of the prediction.
- **Model-specific vs model-agnostic** Model-specific interpretability methods can only be applied to a specified set of model classes because such a method typically works by examining model internal structures or parameters. Intrinsic interpretability methods are by definition always model-specific. Model-agnostic methods allow interpretation of any model, independently of implementation or structure because they approach the model as a black box. They do not have access to model internals or parameters and derive explanations by alternating input and analysing input-output pairs. They are by definition always post-hoc.
- **Local vs global** Another way, how we can classify interpretability is by its scope. Global methods try to explain the overall behaviour of the model - over the whole population, therefore they provide an assessment of the quality of the model. In practice, however, a global explanation is difficult to obtain. The model will be very complex and it is not within the mental capacity of an average human to comprehend so many aspects and dimensions. Local explanations interpret one individual prediction. It can therefore provide a better explanation of feature contribution in less significant groups, which could be left unnoticed by global methods. Since holistic global interpretability is so hard to

achieve, it is worth mentioning that for some model structures (e.g. naive Bayes) it is possible to achieve global interpretability on a modular level - only for a certain part of a model. Or we can apply a global method only for a subgroup of predictions.

2.3 Intuitively interpretable models

First, we shall have a brief look at intrinsic interpretability, because many complex interpretability techniques rely on an intuitive understanding of simple models.

As Lundberg and Lee write in (LL17) "the best explanation of a simple model is the model itself". In this section, we will not introduce any interpretability methods. Instead, we shall characterize some of the models which are simple enough that they are easy to understand without further processing.

We also need to keep in mind that the goal of interpretability is to provide understanding of the relationship between the input and the prediction. Therefore, we must also take into consideration the expertise and limit of the target audience. For instance, for a machine learning professional, gradient vector might be understandable, while laymen would find a short list of weighted features more enlightening.

For our use case, the explanation should help journalists find key parts of a text. Hence we are striving for an explanation model, which will assign weight to a text unit. This can be further presented, for example, in a form of highlighted text.

Therefore, even though there are of course many more interpretable models, we shall limit the overview to the several most significant ones. Moreover, since the following models are generally known machine learning models, we shall not describe them in detail, we will only briefly outline their explainability utility.

Linear regression

Linear regression is possibly the most famous simple model. Many complex post-hoc methods are trying to emulate it because thanks to linearity, the model is easy to interpret. The prediction is a weighted sum of the feature inputs and therefore we can understand the model outcome just by reviewing model weights and inputs.

If we then want to measure the importance of a feature, we can do so by calculating its t-statistics as the wight β_i divided by the weight's standard error ($SE(\beta_i)$) which represents variance.

$$t_{\beta_i} = \frac{\beta_i}{SE(\beta_i)} \tag{2.1}$$

What is however crucial for the interpretability of a linear regression model is sparsity. Because while it is easy to explain (and visualize) a prediction made with two features, it is harder to understand the model as the number of features is growing. Probably no one can imagine regression with hundreds of dimensions. Therefore, for models with many features, we often want to introduce sparsity by applying some feature selection algorithm, such as forward selection, backward selection or Lasso (least absolute shrinkage and selection operator).[1]

Logistic regression

Logistic regression is used for binary classification and we can view it as an extension of linear regression for classification. Since the prediction of linear regression is not a probability, but a linear interpolation, it can be hard to find a substantial trash-hold between classes. Therefore we often use logistic regression, which maps linear prediction into interval $[0, 1]$.

The interpretability of the logistic regression prediction is not as straightforward as with linear regression, since the weights no longer influence the outcome linearly.

We can however calculate odds ratio for each feature, which tells by which factor the odds of the result change with the feature value increasing by 1 and all other feature values remaining the same.

Decision tree

Decision trees are well suited for situations when features are not in a linear relationship with the predicted value and when they interact with each other. Tree models repeatedly split the data according to cutoff values in selected features and thereby distribute the data into different subsets. The prediction for each node is then achieved by averaging the value of the predicted feature over training data belonging to the subset of the node.

Decision tree predictions are understandable because we can track the decision through the tree and in every node explain the change in a predicted value.

[1] An overview of feature selection algorithms used in this work can found in (Wan18).

2.4 Interpretability methods

Intuitively interpretability has many perks. However, models, which are currently best at automated fact-checking, are large and complex neural language models (TVC$^+$18). And not even a machine learning expert can comprehend a network with multiple layers and thousands of nodes and weights. In order to explain such a model's prediction, we need to use a post-hoc explainability method, some of which, we shall introduce in this section.

Because the goal of this work is to help users understand individual predictions rather than to provide insight into the model, we will limit our overview to local interpretability methods, which are more suitable for this task.

2.4.1 Local Interpretable Model-agnostic Explanations (LIME)

As the characteristics in the name suggest, Local Interpretable Model-agnostic Explanations (LIME) is applied for a single prediction of any machine learning model. It is based on the following technique: For a single prediction of any black-box model, we locally approximate the original, usually complex model by a different, simpler and usually self-explanatory model, such that it fits the original model's outcome in the neighbourhood of the target prediction. The simple, intuitively-interpretable model is called the local surrogate model and it can be used to intuitively interpret the original model's prediction.

LIME, introduced by Ribeiro, Singh and Guestrin (RSG16) is a proposition of a specific implementation of the local surrogate model.

As was already stated, LIME's main objective is to explain why the model made a certain prediction for a particular input instance. The main principle is, that it generates random samples in the neighbourhood of the prediction's input by perturbing the input. Then, it uses the original black box model to predict output for each of the newly generated input samples. Finally, LIME trains an interpretable model on the newly created dataset of generated samples, annotated by the original model, while each sample is weighted by the proximity of the sample to the target instance, we are aiming to explain. The learnt model is then a good approximation of the black-box model locally, although it does not have to be a satisfactory approximation globally, e.g. features which are critical locally might be negligible in the global scope and the other way around. This criterion is called local fidelity.

The learnt interpretable model can be any intuitively interpretable model (section 2.3). However, in practice, both, the current implementation of LIME and later chapters of the (RSG16) paper, only offer/consider linear regression models.

Formally, the explanation ξ is obtained as:

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.2}$$

G is a class of interpretable models (section 2.3).

f is the original true model.

We use $\pi_x(z)$ as a proximity measure between target instance x and a generated instance z. It is a local kernel function which weights input z based on its proximity to target instance x and hereby specifies how broad the neighbourhood of instance x should be. π_x then represents locality around x.

$\mathcal{L}(f, g, \pi_x)$ loss function, which represents infidelity - how unfaithful approximation g is to true model f.

$\Omega(g)$ stands for the complexity of model g, which in (RSG16) stands opposite to interpretability and it can vary for different model types - for instance for decision tree it could be its depth.

Hence we can state that LIME's explanation is produced as a fidelity-interpretability trade-off. However, in practice LIME does not optimize complexity $\Omega(g)$ (in the paper it is suggested as future work), it has to be set as a constant in the development stage and LIME only optimizes infidelity.

Sparse Linear Explanation. Let us now consider an alternative, where G is a class of linear models, such that $g(z') = w_g * z'$, as it is the case with the current LIME implementation. The explanation returned will have a form of list of feature weights w_g and it is obtained as defined in eq. (2.2), but the paper closer specifies loss function $\mathcal{L}$ as a locally weighted square loss

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{2.3}$$

$\mathcal{Z}$ is a dataset of perturbed samples.

$\pi_x(z)$ is kernel function, which takes distances and kernel width and returns weights. By default an exponential kernel is used, denoted as:

$$\pi_x(z) = exp(-D(x, z)^2/\sigma^2) \tag{2.4}$$

The exponential kernel uses any defined distance function D and a kernel width σ, which is provided during the development phase and determines how close the instance must be to the original input instance in order to affect the local model.

LIME for text. Since the goal of this work is to explain text data, we decided to also elaborate on specifications used for textual input. LIME can also be used for images, however, it is not relevant for this work.

The algorithm works as described above, using cosine distance as a distance function for exponential kernel in eq. (2.4). The main distinction is in the sample generating process. It splits the text into words (each word becomes a feature) and then randomly erases some words from the original text. Each dataset is expressed with a binary value for each word. 1 if the word is present and 0 if it was removed from the text. The explanation will be interpretable, because the interpretable representation is a bag of words, and we limit the number of words returned in the explanation with K, which is specified during the development phase.

Advantages and drawbacks. LIME method is frequently used thanks to its many good aspects - it results in short and therefore often readable and contrasting explanations, it allows us to use comprehensible features derived from the ones, the model was trained with and with implemented libraries it is very easy to use. However, there are several drawbacks which are worth considering.

Its instability and lack of robustness can be a serious deficiency. Due to the random data sampling process, the result - the explanation is not stable and repeating the explanation process with the same prediction instance and the same model can lead to different feature weights. In (AMJ18) the authors measured the robustness of different interpretability methods and showed that LIME is profoundly sensitive to subtle perturbations in the input data, which have minimal effect on the model's predicted class probabilities. It also showed that for two close instances, the explanation varied significantly in different experimental settings.

This could be connected with another issue, which is the problem of locality setting π_x - how major impact it has on the explanation while being insufficiently defined (Mol22). One of the characteristics of LIME is that it is local, but the method itself does not define what the neighbourhood should be. With the default implementation alone, kernel width is a game-changing parameter, which can literally turn the explanation around, yet it is set to a constant unexplained value without any guidance on what a good kernel width could be for different problems.

Finally, especially in the context of textual data, imperfect sampling is worth mentioning. Even for tabular data, the sampling is derived from Gaussian distribution ignoring inter-feature relationships, therefore the generated samples might be very implausible. For textual data, the effect is enhanced because the possible value of each feature (word) is reduced to present and absent. Therefore perturbed texts contain a subset of words from the original text but when read by a human, they might not make sense because the semantics might get lost during the perturbation.

2.4.2 Shapley values

Shapley values originate in cooperative game theory and they were introduced in 1953 by Lloyd Shapley (Sha16) as a technique for fair distribution of a payout among a team of n players.

Imagine we have a team of n players, who together win a payout. The question arises - how to fairly distribute the payout among the n players according to their individual contributions?

This idea can be applied to explain a particular prediction p with the following mapping:

- *payout* (shared by all players) is the difference between the prediction p and the full-dataset-average prediction. It is the predicted value (for regression) or probability (for classification), predicted for the explained instance, reduced by the average predicted value over all instances.
- *players* are feature-value pairs of the instance that cooperatively won the payout (resulted in a prediction p).

For each feature value, we are trying to tell, with what value the feature contributed to the predicted value of the instance in comparison with the average prediction. And that is a Shapley value.

Shapley value for one feature-value pair of the target instance can be defined as the average marginal contribution of the feature-value pair across all possible coalitions.

$$\phi_i(x) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!}(f(x_{S \cup \{i\}}) - f(x_S)) \tag{2.5}$$

We want to compute $\phi_i(x)$ - Shapely value of the i-th feature for instance x. We have a model f, predicted instance x and a set of all features F used by model f to predict instance x.

We take a set of all model features F except for feature i (which we are computing Shapley value for) and we create a set of all possible coalitions (subsets) of the features.

We further denote $f(x_S)$ as a prediction of model f made with features in the set S with their according values from instance x.

For every coalition S of features we calculate the contribution of the feature i for the coalition. The contribution is the predicted value for features in the set S and the feature of interest i, and their according values from instance x reduced by the predicted value for features S without the feature i. The Shapely value $\phi_i(x)$ is a weighted average of all these contributions.

There is however a problem in the approach above, in step $f(x_S)$. How do we remove a feature from the input? For tabular data, if the model was trained with certain features, we need to provide all of them for obtaining a prediction. There are several solutions.

Shapley regression values. For linear models, we can calculate Shapley regression values, which solve the problem by model retraining. For every feature subset S, we need a model $f_{S\cup\{i\}}$, trained with the feature present and model f_S, trained without the feature of interest. The equation eq. (2.5) then has the following form:

$$\phi_i(x) = \sum_{S\subseteq F \ \backslash\{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}(f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)) \tag{2.6}$$

That is however costly since we would need to train $2^{|F|}$ models - two for all possible feature subsets.

Shapley sampling values. A different approach is applied with Shapley sampling values, which approximate the effect of feature removal by integrating over samples from the training dataset. To calculate them, we randomly sample an instance y from the dataset, which we call the donor. Then as input to the model, we provide the value of explained instance x (x_i) for features in the coalition ($i \in S$) and the donor's value (y_i) for features, which we want to represent as missing ($i \notin S$). The result can be misleading because a) the process can generate unlikely instances, b) some donor's values can be the same as the values in the instance x, therefore there will be no difference for prediction with and without the feature. However, we can repeat the sampling and average the contribution to get a more precise estimate of the true value.

Shaply values for text. For text data, each token is one feature, its value is simply present, 1 (0 would represent a missing value). Depending on the tokenizer, this can be problematic, because it is common to tokenize on the level of words or even subwords (SHB16). And since the number of possible coalitions increases exponentially with the number of features, the computation of Shapely value is very expensive.

On the other hand, the conceptual simplicity is an advantage when using Shapely values for textual data. Since the model input is defined as a string sequence, we do not need to bother with the problem of feature removal as described above and we can simply mask the feature (token) from the sequence. Therefore we can calculate Shapley values as described in 2.5, with the original model, without integrating over the dataset.

Advantages drawbacks. The main advantages of the Shapley method are the guarantee of fair distribution among features and a strong theoretical groundwork. The fundamental problem of the technique is its complexity. The computational time increases exponentially with the number of features, therefore for more than a few features, calculation of the full exact solution becomes problematic and for real-life in bulk use cases potentially infeasible. Therefore instead of computing Shapley values, we often chose to estimate them.

Shaply value estimation. There are several estimation approaches and algorithms. For instance, instead of calculating the marginal contribution of a feature for all possible coalitions, we could only compute it only for a sampled subset of coalitions. Alternatively, there is a popular algorithm by Štrumbelj and Kononenko proposed in (ŠK14), which approximates Shapley values based on Monte-Carlo integration. Or we can use SHAP with its approximation methods.

2.4.3 SHapley Additive exPlanations (SHAP)

SHapely Additive exPlanations (SHAP) is a local, model-agnostic interpretability technique based on Shapley values (section 2.4.2). It was first introduced by Lundberg and Lee in 2017 in (LL17), where the authors propose the main concept, describe its properties and introduce six SHAP approximation methods.

Additive feature attribution method. Before we dive into SHAP itself, it is worth pointing out an innovative view on Shapley value explanations, that Lundberg and Lee pose, which combines Shapley value with LIME (will be further elaborated in Kernel SHAP). They represent Shapley value explanation as an additive feature attribution method (method whose explanation model is a linear function with binary variables), which in is defined as:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{2.7}$$

where g is explanation model. M is a number of simplified input features. And $z' \in \{0,1\}^M$ corresponds to the coalition of simplified input features present.

Simplified input features. The simplified input features x' map to the original model features x given simplified input mapping h_x; $x = h_x(x')$. Therefore they can but do not have to be the same as the original features. They provide an option to reduce the feature space for explanation purposes. An example could be image input, the original model might process the image on the pixel level, however explanation with pixel granularity is too complicated and hard to read for most users. Therefore for the sake of clarity, for explanations, pixels might be grouped into superpixels, which would then become the simplified input features and an input of the explanation model g. Another, for our project more appropriate, example is text processing.

The original text processing model might require text tokenized (split) into word pieces and relies on abstract embeddings. That would be too expensive to calculate and complicated to explain. Therefore, we might decide to set simplified input features as words or even sentences and calculate Shapley values for them.

Properties. Shapley values are the only additive possible additive feature attribution method with the following properties:

1. **Local accuracy** - The explanation model $g(x')$ matches the original model $f(x)$. ϕ_0 represents model prediction with all features missing.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{2.8}$$

2. **Missingness** - If a feature is missing, it has no impact to the model.

$$x'_i = 0 \Rightarrow \phi_i = 0 \tag{2.9}$$

3. **Consistency** - If a contribution of feature i is higher or the same in model f' than in model f,

$$f'_x(z') - f'_x(z'_{\setminus i}) \geq f_x(z') - f_x(z'_{\setminus i}) \tag{2.10}$$

then Shapley value ϕ_i should also be higher or equal in model f' than in model f.

$$\phi_i(f', x) \geq \phi_i(f, x) \tag{2.11}$$

SHAP. Now let us turn our attention to SHAP itself. We can look at SHAP as an alternative approach to Shapley values computation. Its goal is to preserve desirable properties of Shapely values while applying several approximations which enable more feasible computation time.

Since SHAP is based on Shapley value computation, its narrative is similar. As for all local interpretability methods, SHAP's purpose is to explain the prediction of a selected instance x. It does so by computing feature importance - the contribution of each feature to the prediction. In the previous section we described how we can measure feature importance with Shapley values and the SHAP paper proposes to measure feature importance with SHAP values.

SHAP value is a Shapley value of a conditional expectation function of the original model. Since for most models, it is impossible not to provide features value to emulate them missing, SHAP approximates $f(z_S)$ with $E[f(z)|z_S]$, where z_S are features in the set S with the corresponding value from instance z.

As visualized in fig. 2.1, ϕ_i (SHAP value of a feature i) is the change in the expected model prediction when conditioning on the feature i. We define a base model value $E[f(z)]$ as a value which would be predicted if there were no features present in the coalition $z \in \{0\}^M$. SHAP values explain how to get from the base value to the predicted output value $f(x)$.

For non-linear models or models where input features are not independent, the order of the features matters, therefore SHAP value is then the average across all possible orderings.

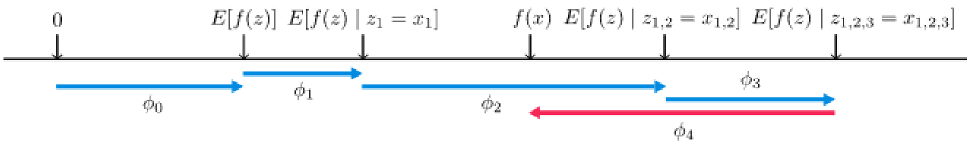


Figure 2.1: SHAP values depict how to get from base value $E[f(z)]$ to the actual predicted value $f(x)$. Figure from (LL17)

Besides approximating $f(z_S)$ by conditional expectation as $E[f(x)|z_S]$, SHAP offers two further optional approximations, which simplify the computation of the expected values - assumption of feature independence and model linearity. These two assumptions combined allow SHAP to replace expensive integration over samples by directly inserting the mean value of the feature in the dataset.

Since the exact computation of SHAP values is challenging, the authors in (LL17) introduce six approximation methods to estimate SHAP values. Two model-agnostic - Shapley sampling values (already known for Shapley values) and Kernel SHAP (novel method) and four model-specific - Linear SHAP, Low-Order SHAP, Max SHAP and Deep SHAP. Later in (LEL18), there was another model-specific approximation method - Tree SHAP. From the perspective of this project, we want to explain neural network models. Therefore only sampling SHAP values, Kernel SHAP and Deep SHAP methods are applicable.

Kernel SHAP

Kernel SHAP is a method which combines Shapley values and LIME. Or more precisely, it is an extended method from LIME, which due to its kernel function approximates SHAP values with much fewer coalitions than required for their full computation.

Since it is an approximation method for the calculation of SHAP values, we shall use notations and definitions which were defined and explained in SHAP sections section 2.4.3 - e.g. coalition of simplified input features $|z'|$, or simplified features mapping $h_x(x')$.

Same as LIME, Kernel SHAP uses linear explanation model g to locally approximate the original model f. And the formulation is also the same as in LIME eq. (2.2) with the following specifications:

There is no penalisation for the complexity of the explanation model g.

$$\Omega(g) = 0 \tag{2.12}$$

Loss function is as in LIME eq. (2.3) weighted square loss, weighted by kernel $\pi_{x'}(z')$.

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_{x'}(z') \tag{2.13}$$

Here however comes the biggest difference compared to LIME - in the kernel function, which assigns weight to each sample. While in LIME the kernel weights samples according to their proximity to the instance of interest x (based on a heuristically chosen kernel width), SHAP chooses the kernel analytically. It assigns weight to the sample according to the Shapley value estimation of the coalition z'.

$$\pi_{x'}(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)} \tag{2.14}$$

$|z'|$ corresponds to the number of non-zero elements in coalition z' and M is a number of simplified input features.

The flow of the method is following:

- Randomly sample coalitions Z.
- For each sample coalition (z') :
  - Map simplified input features into original model feature space
  - Predict sample with original model f
  - Weight sample with SHAP kernel $\pi_{x'}(z')$
- Fit linear model
- Return the coefficients of the linear model as SHAP values

2.5 Interpretability assessment methods

While explainable AI seems to be on the rise with a ceaseless stream of new papers introducing explainability methods, or claiming that their model is interpretable, the situation is not so bright with interpretability assessment methods. In fact, according to (AB18), only 5% of the papers, that they have examined, covered evaluation of interpretability methods and quantification of their relevance.

Many papers we encountered approached interpretability as a qualitative evaluation of the model and not as an independent development stage with its objectives and the need for measurement. They would simply assume the explanation is faithful and use it to gain insight into the model and control how robust the model is. And while this approach can be helpful to attain a robust and non-discriminatory model, it is highly unsuitable when the examined subject is the interpretability method itself, such as in this work.

And even papers which focus on the evaluation of interpretability methods usually define desiderata and properties of interpretability evaluation or describe its taxonomy, rather than proposing a specific, well-defined and measurable metric. Such finding is not surprising given the subjective character of interpretability, however, it makes rigorous comparison and evaluation of interpretability methods difficult.

Doshi-Velez and Kim in (DVK17) present following taxonomy of interpretability evaluation:

- **Application-grounded evaluation** is the most expensive to perform but arguably provides the strongest evidence of success, especially for human-computer interaction tasks. It requires conducting a human experiment, in which the domain experts (target audience) use the explanations during the exact task where the explanation method should be applied.
- **Human-grounded evaluation** requires a human experiment where human subjects perform a simplified task, which evaluates the explanations. The experiment could be for instance *binary forced choice* when subjects choose better of two explanations or *forward simulation* when subjects need to correctly simulate the predicted output of the model when presented with the explanation.
- **Functionally-grounded evaluation** does not require a human experiment and instead specifies a *proxy task* based on a specified mathematical definition of interpretability, which it then measures. Since it is the simplest to implement, it is possibly most suitable for feasibility studies (MV20). The key problem of functionally-ground evaluation is the selection of proxies and a specification of criteria, which generally stays an open problem and according to (DVK17) is important for future research.

Doshi-Velez and Kim in their paper define interpretability as *"the ability to explain or to present in understandable terms to a human."* and their evaluation for interpretability defined as such is measuring whether the explanation brings the desirable benefits to the target audience performing a specific task (in a real or emulated environment)[2]. This makes their

[2]Of course, in their taxonomy, there is space for a different, mathematical definition of interpretability with functionally-grounded evaluation, but it is not defined closer in the paper.

evaluation approach well suited for human-computer interaction, such as ours. And so, we are going to use this definition while designing our solution and experiments.

However, from (not only) our point of view, there is more to explainability than just how understandable it is to a human. The above approach to explainability is not complete because it omits one crucial property of explainability and that is fidelity - how faithful the explanation is to the actual model's behavior.

Rudin in his paper (Rud19) writes: *"Explanations must be wrong. They cannot have perfect fidelity with respect to the original model. If the explanation was completely faithful to what the original model computes, the explanation would equal the original model, and one would not need the original model in the first place, only the explanation."*

And while that is true, fidelity to the model is often stated as desiderata of explanation - e.g. in (ZC18). Or even LIME (section 2.4.1) in its core tries to minimize infidelity from the original model - see eq. (2.2).

The important aspect for us and arguably the greatest weakness of human-based interpretability evaluations is that we are unable to observe the boundary between the model's correctness and the interpretability fidelity. Consider a scenario when the model predicts the correct output but our chosen interpretability method shows that it was chosen due to nonsensical input values. How do we decide if this occurred due to the infidelity of the explanation, or if the explanation is reliable but it is the model that lacks robustness and predicts based on chance and noise?

In other words, people evaluate, whether they like what the explanation says that the model decided upon, but not whether it is consistent with the actual model logic. Nevertheless, this weakness is hard to overcome. If we want to assess the helpfulness of the explanation for the target user, we cannot segregate it from the prediction itself.

That being said, the evaluation measures described above are, despite these shortcomings, still frequently used in various forms (overview in (MZR18)). And we are going to use them as well in this work.

Chapter 3

Problem specification

In this chapter, we shall define the problem this work is going to solve. We shall first introduce the datasets which we will be working with. Then we will analyze which part of the fact-checking pipeline is most suitable for explaining. Finally, we will describe the chosen task in more detail - outline its structure, present the models, which we will be explaining, and clearly define their input and output.

3.1 Datasets

During majority of this work, we worked with two datasets, which were both created by the research group of the Czech fact-checking project and are presented in (DUR+22). In this section we will introduce each of the datasets.[1]

3.1.1 CsFEVER

The dataset, presented in (DUR+22), is an automatically generated Czech localization of the large-scale Fever dataset (TVCM18). It contains 127,328

[1] In the early stages of the project, we also used the IMDb Movie Review Dataset (MDP+11), which is one of the standard benchmark datasets used for sentiment analysis and contains 25k positive and 25k negative film reviews. We used the dataset solely for exploration purposes.

The input to the model, in our case, is two texts (claim and context) (as is further explained in section 3.3.3). This often caused problems when using libraries for interpretability methods, which usually expect a single text as an input and we were often required to edit the library implementation. Therefore, when we wanted to first try out and observe an explainability method, we preferred to do so on a standard binary classification of English text. And that is when we used the IMDb dataset. Moreover, the binary outcome and the possibility to try different models made it easier to judge the quality of the explanation.

Since there is no outcome of this work regarding the IMDb dataset, nor did we use it for experiments, we shall not specify it any closer and in the rest of the work we will consider only the two datasets properly presented in this chapter.

claims annotated as *SUPPORTS, REFUTES* and *NOT ENOUGH INFO* (*NEI*), while the claims of the first two labels are *verifiable*.

	SUPPORTS	REFUTES	NEI
train	53,542	18,149	35,639
dev	3,333	3,333	3,333
test	3,333	3,333	3,333

Table 3.1: Label distribution in CsFEVER dataset

EnFEVER. The original English dataset - Fact Extraction and VERification (FEVER) (TVCM18) is a large scale dataset with 185 445 claims. It was created based on 50k most popular Wikipedia articles using an elaborate two-stage annotation methodology.

In the first, *claim generation*, stage, the annotators were asked to create claims based on an article's abstract (the first paragraph containing a brief summary of the article) and a dictionary (terms hyperlinked in the abstract with the first sentence from their corresponding Wikipedia article) and to generate various mutations of the original claim.

In the second, *claim labeling*, stage, the annotators labeled each claim generated in stage one as SUPPORTS, REFUTES or NEI and if the chose SUPPORTS or REFUTES, they were asked to select evidence for the decision from any Wikipedia page.

CsFEVER. The Czech localization of the original dataset was created with usage of interlanguage linking and machine translation. First, each article in the evidence was mapped to a corresponding Czech article while all evidence sets, where an article's localization was not available, are deleted. All verifiable data points with no evidence are deleted as well. Then, using a machine translation, claims were translated and eventually the dataset was re-split into its final form table 3.1.

id	167839
verifiable	VERIFIABLE
label	SUPPORTS
claim	Hmotnost ledovce může být ukazatelem klimatických změn.
claim_en	The mass of glacier can be an indicator of climate change.
evidence	[[193217, 203804, "Ledovec", 17, "Glacier"][1]]
	[[193225, 203814, "Ledovec", 17, "Glacier"][1]]

[1] The evidence refers to a paragraph in an article

Table 3.2: Data point example of CsFEVER dataset

Validity. CsFEVER is constructed with an assumption that the abstracts of English and Czech Wikipedia pages contain the same information. However, as the authors manually tested on a subset of data points, it is not always the case and for 28% of verifiable claim-evidence pairs, the information needed to infer the label is not provided in the Czech abstract. Further 5% of samples were invalid due to inadequate translation. Overall, only 66% of the claim-evidence pairs are valid.

3.1.2 CTKFacts

There are many parallels between the CTKFacts dataset and the CsFEVER dataset section 3.1.1 because the overall creation process as well as structure of the CTKFacts dataset were inspired by FEVER dataset (TVCM18).

Same as for the CsFEVER dataset, the CTKFacts dataset was presented in (DUR+22). It contains 3,097 claims annotated with the same labels as in CsFEVER - *SUPPORTS, REFUTES* (marked as *verifiable*) and *NOT ENOUGH INFO* (*NEI*). The label distribution can be seen in table 3.3

	SUPPORTS	REFUTES	NEI
train	1,104	556	723
dev	142	85	105
test	176	79	127

Table 3.3: Label distribution in CTKFacts dataset

The main difference is in the corpus and by extension in the collection process. CTKFacts was created using Czech News Agency[2] corpus with 2.2M articles as the ground-truth corpus instead of FEVER's Wikipedia corpus. This causes a range of differences - the articles do not contain an abstract, the communication style is different with less self-contained sentences and there are no hyper-links in the articles to create dictionary.

Dictionary creation. Since the dictionary has a key role in the annotation process, the authors had to construct it. Instead of hyper-linked articles, they used the most relevant articles, which were found using a TF-IDF document retrieval method combined with a two-tower retrieval model.

Annotation process. The annotation process is based on FEVER's annotation process, modified to fit the corpus specification.

Before the annotation itself, the authors preselected paragraphs with check-worthy information, which are used in the next stage.

Then in the *Claim Extraction* phase, the annotator is given a *knowledge scope* (a random paragraph and its generated dictionary) and is asked to produce a simple true initial claim, supported by the knowledge scope (while

[2] Česká Tisková Kancelář (ČTK) - `https://www.ctk.cz`

disregarding their own knowledge). In the next *Claim Mutation* stage, they are asked to generate various mutations from the initial claim, such as rephrasing, negating or generalizing.

Finally, in *Claim Labeling* step, the annotator is presented with a randomly sampled mutated claim m and a knowledge scope consisting of the knowledge scope of m's initial claim and of a newly generated dictionary for m. The annotator is asked to select a label and if the label is verifiable, then also select the minimum necessary evidence to deduct the label.

id	2292
verifiable	VERIFIABLE
label	SUPPORTS
claim	"Jozef Tiso byl odsouzen za válečné zločiny."
evidence	[T201604210827502_2^{t1}] [T201604210827502_6^{t1}]
source	T201604210827502_1^{t1}
mutated_from^{t2}	2289

^{t1} The evidence and source refer to paragraphs in articles from the CTK database.

^{t2} mutated_from refers to an original claim, which this claim was created from.

Table 3.4: Data point example of CTKFacts dataset
The evidence identifier is in a format *ctkId_paragraphNumber*, where *ctkId* is the id of the wanted article in CTK archive and *paragraphNumber* refers to the paragraph in the article (0 is a headline).

3.2 Choice of task

As we illustrated in fact-checking pipeline section 1.1.1, the fact-checking project consists of multiple components, which contain different NLP models, whose predictions we could explain. Namely, we were deciding between document retrieval (DR) and natural language inference (NLI) tasks.

In order to choose the most suitable task, we need to keep in mind the goal of the fact-checking project. And that is to help journalists verify statements (claims), to save their time, as well as mental capacity and guide them through the quanta of text and information to the sought answer.

We believe that of the two tasks discussed, an explanation of NLI will be a bigger help in the verification process because it is at the end of the fact-checking pipeline, and the user can therefore see the direct contribution to the final result.

Of course, as we stated in the Introduction chapter 1, the automated fact-checking application should not be used as an omniscient, statement-labelling black box, but rather as a tool guiding the user to the answer. And in this process, document retrieval is a crucial step. However, when the user gets a response that the fact-checked statement is or is not true, the first thing, they would probably want to see, is why. And since the prediction was made with respect to the chosen background text (context), pointing out, what the prediction was based upon in the particular text will most likely provide a satisfactory answer.

We could argue that if there was a mistake in the earlier step – document retrieval, then the explanation of the prediction of the NLI model is arbitrary because the background data is noise. And in that case, the user would possibly prefer to retrace the steps of the pipeline and choose a more suitable background text, where the explanation of the DR task would be more useful.

And while that is true, the user first needs to discover that the chosen text is irrelevant. And possibly seeing a nonsensical explanation would be an indicator that something is wrong. Because while the main benefit of clear explanations in this project is a fast look up of evidence, there is also the benefit that seeing a nonsensical explanation should alert the user not to automatically trust the machine.

Another reason why we chose the NLI task was its strong background, from the perspective of both theoretical ground and already implemented frameworks. Many interpretability methods specifically consider text classification. Papers which propose them often dedicate a section or a paragraph to the text classification problem and implemented libraries offer a pre-built solution.

Of course, there are implemented libraries for the interpretation of sequence-to-sequence models (e.g.. SHAP[3]) as well as the theory behind them, however, there are not as many options. And we hope that the more explored domain will yield better results.

Nevertheless, explaining DR models would also be very beneficial for the project and while we will not cover it in this work, we would recommend it for future work.

3.3 Natural Language Inference

Now, that we outlined the task, we are going to focus on, let us examine it closer in order to understand what exactly we are going to explain.

As described in section 1.1.1, in our fact-checking pipeline, when we want to verify a claim, we start by retrieving a set of evidence relevant to the claim, then we take the claim and the set of evidence and proceed to infer whether the claim is supported by the evidence.

The task in which evidence is retrieved is called document retrieval and beside a brief description in Fact-checking pipeline section 1.1.1 is not further described in this work, it can be found in (DUR$^+$22).

[3]https://shap.readthedocs.io

The task in which we classify the veracity of the claim based on the retrieved evidence is called *Natural Language Inference (NLI).*

3.3.1 Definition

Natural language inference is the problem of determining whether a natural language hypothesis h *can reasonably be inferred from a natural language premise* p. (Mac09)

In our case the hypothesis is the claim and our premise is the evidence. The output labels of NLI can be: *entailment* (in our case SUPPORTS label), *contradiction* (in our case REFUTES) or *neutral*, sometimes also undetermined (in our case NEI). Neutral means that the hypothesis is compatible with but not inferable from the premise.

3.3.2 Models

The NLI (or Recognizing Textual Entailment - RTE, how the task has been previously known), has been historically solved using a range of approaches (WJ15): shallow - relying on lexical similarities, deep - relying on full semantic interpretation or approaches relying on formal logic. In the last decade, the problem has been taken on by neural networks, namely with long short-term memory networks, recurrent neural networks and in the last five years, as nearly every task in NLP, by Transformer-based models (section 1.1.2).

The state-of-the-art NLI models used in our fact-checking pipeline (presented in (DUR+22)) are all based on BERT (section 1.1.2). And they all rely on transfer learning section 1.1.2.

The authors tested different models pretrained on Czech data - multilingual (SlavicBERT or Sentence M-Bert (RG19)), crosslingual (XLM-RoBERTa) and even multilingual Czech models (RobeCzech (SNSS21), FERNET-C5 (LŠ21)). The best results were achieved with XLM-RoBERTa models (CKG+19).

The highest accuracy on the CTKFacts dataset was achieved with the XLM-RoBERTa model finetuned on the NLI-related SQuAD2 (RZLL16) downstream task, therefore we are going to use this model.

The highest accuracy on CsFEVER data was achieved with the XLM-RoBERTa model finetuned on the crosslingual XNLI (CRL+18) task. However, we do not have the model available, therefore we are going to use the highest-scoring model available to us, which XLM-RoBERTa finetuned on the NLI-related SQuAD2 as for CTKFacts.

3.3.3 Input processing

We already stated that the output of our NLI model is a label - SUPPORTS, REFUTES, NEI. We further wrote that the input is a claim and an evidence (fig. 1.1). We also presented the structure and content of datasets (well visible in table 3.4, table 3.2) and each data point contains references to the evidence, not the evidence itself. But in what format is the data actually passed to the NLI model? In order to be able to design our explanation, we need to understand the structure of the input. Therefore we shall now elaborate on the data processing and clarify the true input to the model.

For simplicity reason, we are going to use one exemplary claim - an example from the CTKFacts dataset shown in table 3.4. The process is analogical to CsFEVER just with a different reference system.

We start with a dataset stored in JSONL[4] format with an id, label ($\in SUPPORTS, REFUTES, NEI$), claim and a list of evidence sets [5]. An evidence data point can have different forms, but overall, it is a reference to a paragraph of text. A set of evidence is then a set of paragraphs which are together necessary to infer the output label.

Text evidence format. We retrieve the text from the evidence paragraphs and replace the references with the real text. We call this data format a *text evidence format* and a data example in this format can be seen in table 3.5.

id	2292
label	SUPPORTS
claim	"Jozef Tiso byl odsouzen za válečné zločiny."
evidence	["Jozef Tiso byl po druhé světové válce obviněn z plné zodpovědnosti (...) "^{t1}
	[" Esterházy byl v roce 1947 za své předválečné a válečné aktivity odsouzen (...)"^{t1}]

^{t1} paragraph is truncated

Table 3.5: Data point example of CTKFacts in *text evidence format* with truncated paragraphs

NLI evidence format. Finally, we transform the data into *NLI evidence format*, in which we pass the data to the model. (this format was used for model training and we, therefore, use it also for testing, production and the explanation process.) In this format, we have one data point for each evidence set. Therefore we might need to split the original entry and end up with a different number of data points. Our example in NLI evidence format is split into two - table 3.6 and table 3.7.

[4] https://jsonlines.org/

[5] We are disregarding other properties, because we do not need them.

label	SUPPORTS
claim	"Jozef Tiso byl odsouzen za válečné zločiny."
context	["Jozef Tiso byl po druhé světové válce obviněn z plné zodpovědnosti (...) "[t1]

[t1] paragraph is truncated

Table 3.6: Data point example of CTKFacts in *NLI evidence format* with truncated paragraphs
The original data point was split into two - one fore each evidence set.

label	SUPPORTS
claim	"Jozef Tiso byl odsouzen za válečné zločiny."
context	["Esterházy byl v roce 1947 za své předválečné a válečné aktivity odsouzen (...)"[t1]

[t1] paragraph is truncated

Table 3.7: Data point example of CTKFacts in *NLI evidence format* with truncated paragraphs
Data point for the second evidence set.

Chapter 4

Solution

In this chapter, we will present our solution - the explainability methods used and their produced explanations. We will focus on subjectively pre-selecting methods and their parameter settings which yield the best results. We will later statistically evaluate the methods with their best parametrization in chapter 5.

We also implemented and tested a different, functional, approach of method and parametrization assessment, which does not rely on human evaluation and is, therefore, more scalable and less subjective. This measure, however, did not yield valid results, therefore we did not use it to select the best method parametrizations. Its implementation details, results, as well as problems, can be found in appendix B.

4.1 Design

In the first section, we will elaborate on the design decisions - e.i. the choice of methods and the form of the explanation. And we will include the reasoning behind our choices.

4.1.1 Choice of methods

For the choice of interpretability method, we considered three aspects - the fit for our use case, the recognition of the method and its availability.

The most important aspect was the fit for the project. The goal of the project is to explain an individual model output to a journalist, such that it is easier for them to understand and verify the prediction. Therefore, we limit ourselves to the local interpretability methods whose explanations come in a form understandable to a layman (a non-machine learning professional), preferably a list of weighted text units.

Second, we considered how well the method is established. With explainable AI gaining significance, there are plenty of interpretability methods. When selecting which methods we would use, we examined the theoretical foundation of the method as well as the quality and magnitude of its testing.

Finally, we took into account, how demanding it would be for us to apply the method i.e. if we would need to develop the implementation ourselves or if there were already applicable implementations available.

We found implementations for all the explainability methods we considered. However, some were several years dead projects on GitHub. Others were in process of heavy development with many yet unimplemented features and weekly changes. However, we also found extended and well-developed libraries. In these cases, we were further examining whether they provide an implementation for text input and if they include visualization options.

Our decision. We decided to use LIME (section 2.4.1) and SHAP (section 2.4.3). Both methods are local and both produce explanations in the desired form of text unit and assigned significance value.

Regarding their theoretical base and testing, SHAP, together with Shapley values (section 2.4.2), which it is built upon, are possibly the only interpretability methods with a solid theoretical groundwork (Mol22).

And while LIME does not stand on as strong background as SHAP does, it is a well-established interpretability method, with many published use cases. And in the paper, where the method was introduced (RSG16), the authors test it with a human-grounded test (section 2.5).

Finally, arguably the main advantage of both methods is that they are both provided with well-implemented libraries[1], which also include explanation of text classification and a range of visualization options.

4.1.2 Explanation content

Standard text classification is usually an assignment of a label to an input text - e.g. sentiment analysis of reviews. However, as described in section 3.3.3, the input into our models is two texts - a claim and a background text (context). We were therefore considering which of the input texts should be included in the explanation.

We eventually decided that the explanation should contain important phrases solely from the background text (and not from the claim). Because the end-user is interested in the evidence that the claim was supported/refuted and the evidence must be present in the background text (that is how the datasets were composed).

[1]https://www.lime-ml.readthedocs.io/ and https://www.shap.readthedocs.io/

In comparison, if we chose to use both texts for the explanation, the user would also see which parts of the claim led to the prediction. This could draw the user's attention, and they would then focus also on evaluating which parts of the claim are most significant instead of solely searching for the key information in the background text. The approach with both texts could be useful for complex claims, however, all claims in our datasets are simple - by definition in the dataset design (DUR$^+$22).

4.1.3 Tokenization

In NLP, before we pass a text to a prediction model, we need to split it into smaller text units. This process is called tokenization and the text units are then called tokens.

For the interpretability methods, we also need to define the form of a token and by extension the form of our explanation (granularity of the text units, which we will assign weight to).

A compact and to human reader naturally understandable unit is a word. It is so instinctive, that some interpretability libraries automatically assume word tokenization. And we are going mainly focus on it too. However, to gain a better, more complex overview, we will not limit ourselves only to words.

The NLI models whose prediction we are explaining use WordPiece tokenization (WSC$^+$16) which splits the text into sub-words. We were curious how different the results would be if we used the model's tokenization level which moreover strips the word from its suffixes and prefixes.

Finally, we try tokenizing the text into sentences, because sentences can hold context that single words cannot and we believe that a single sentence might be more readable than ten discontinuous words.

There is another aspect to consider, which favours tokenization to coarser units. For explanation purposes, each token represents a feature and the smaller token, the more of them there will be in the text. A high number of features is a problem because the computational complexity of SHAP increases exponentially with the number of features. Hence there is a price to pay for small tokens. Either in a form of high computation time or in the quality of the explanation, because we will need to approximate it more.

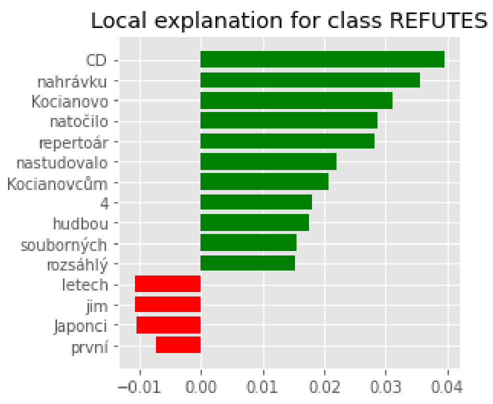
4.2 Implementation

For each dataset (section 3.1), we set to find the ideal method and its parametrization that would produce the best explanations. In order to do so, we used the following approach:

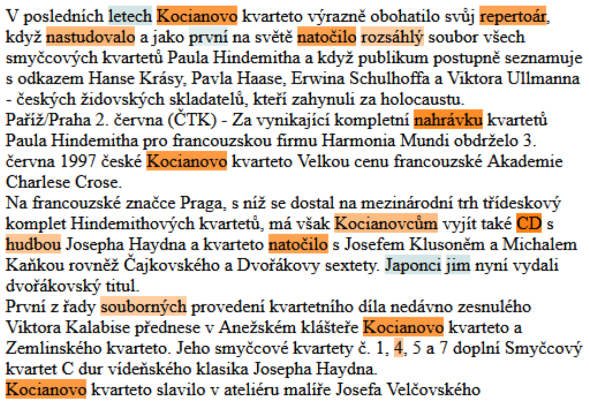
Initially, for each dataset, we randomly sampled 10 data points. We saved the ids and kept using the fixed samples across individual methods and their paremtrizations.

Then, for each method with specific settings, we generated explanations for our samples and individually considered their output.

From our perspective, the evaluation of the explanations is the fundamental problem of the whole interpretability field as we already argued in section 2.5. Due to the absence of a functionally-ground evaluation and the high requirements (on time and resources) of human evaluation, we assessed the explanations by reviewing and comparing them in various forms - in the raw form of a token-weight dictionary but mainly in form of visualizations. Most frequently we used highlighted text fig. 4.1b or occasionally a bar plot fig. 4.1a. We think that the highlighted text is the most informative presentation of the explanations and hence we would also use it as a visualization format presented during the experiments (and to the user).



(a) : Bar plot visualization



V posledních letech Kocianovo kvarteto výrazně obohatilo svůj repertoár, když nastudovalo a jako první na světě natočilo rozsáhlý soubor všech smyčcových kvartetů Paula Hindemitha a když publikum postupně seznamuje s odkazem Hanse Krásy, Pavla Haase, Erwina Schulhoffa a Viktora Ullmanna - českých židovských skladatelů, kteří zahynuli za holocaustu.
Paříž/Praha 2. června (ČTK) - Za vynikající kompletní nahrávku kvartetů Paula Hindemitha pro francouzskou firmu Harmonia Mundi obdrželo 3. června 1997 české Kocianovo kvarteto Velkou cenu francouzské Akademie Charlese Crose.
Na francouzské značce Praga, s níž se dostal na mezinárodní trh třídeskový komplet Hindemithových kvartetů, má však Kocianovcům vyjít také CD s hudbou Josepha Haydna a kvarteto natočilo s Josefem Klusoněm a Michalem Kaňkou rovněž Čajkovského a Dvořákovy sextety. Japonci jim nyní vydali dvořákovský titul.
První z řady souborných provedení kvartetního díla nedávno zesnulého Viktora Kalabise přednese v Anežském klášteře Kocianovo kvarteto a Zemlinského kvarteto. Jeho smyčcové kvartety č. 1, 4, 5 a 7 doplní Smyčcový kvartet C dur vídeňského klasika Josepha Haydna.
Kocianovo kvarteto slavilo v ateliéru malíře Josefa Velčovského

(b) : Highlighted text visualization

Figure 4.1: Example of visualizations - produced by LIME
Claim : *Kocianovo kvarteto odmítlo nahrát skladby Paula Hindemitha.* = *The Kocian Quartet refused to record Paul Hindemith's compositions.*

Based on the above described individual assessment of 10 samples, we filtered out scenarios which produced incoherent and nonsensical explanations. For method settings with reasonable explanations, we sampled more data points (this time new random ones) and repeated the process.

We excluded a parameter setting when out of the 10 samples it produced 2 nonsensical explanations (mainly explains against the decision or contains only random meaningless words) or 4 poor explanations (some words are relevant but most are not).

At this point, we had a good idea about the general quality of the explanations and we moved to the next phase - selecting one, best, parameter setting for both methods (per dataset), such that we could compare them between each other in the experiments (chapter 5).

Best parametrization. We took the remaining method settings (since they produced steadily best results), selected those corresponding to the tokenization level specified for the experiment and conducted a series of short pairwise blinded tests, for which we used the settings and annotation platform designed for the experiments (section 5.1.1).

The flow of each of the tests was following:

- Randomly sample 15 data points.
- Produce explanations with both compared parameter settings.
- In our annotation platform (described in section 5.1.1), one sample at a time, display the explanations side by side and let the annotator (which was us) choose which of the explanations is more helpful.
- Determine the winner of each test using majority voting.

We repeated the test until for each dataset we were left with one parametrization for LIME and one for SHAP.

4.3 Methods and parametrization results

In the following overview, we are going to present a subset of tested parameter settings for each method, chosen to demonstrate method properties and as well as its best results.

We are going to use the same data point[2] for all showcases to make the differences more distinctive. The sample comes from the CTKFacts dataset but the following properties of the methods apply to explanations of the CsFEVER dataset as well.

We did not expect to find such unity in the explanation qualities across the datasets. We suspect that it could be caused by the fact, that both the task as well as the explained NLI models are very similar.[3].

4.3.1 LIME

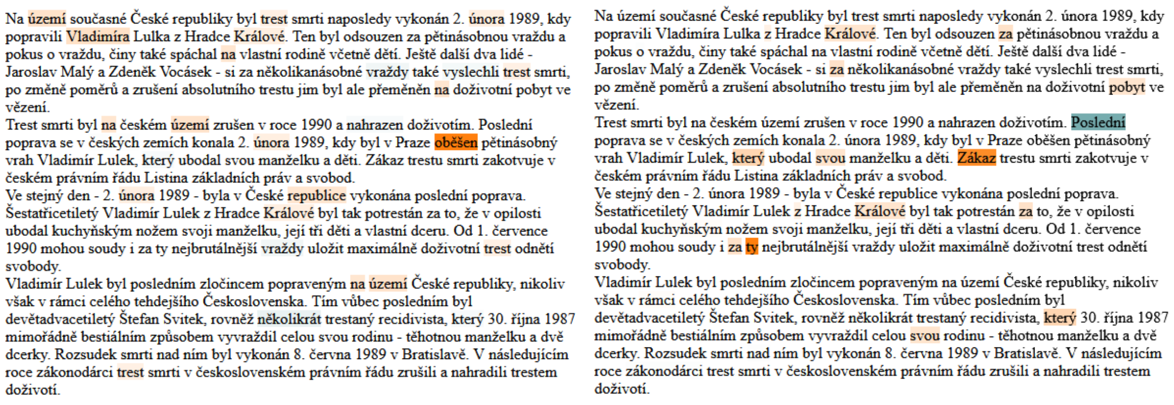
As described in section 2.4.1, LIME trains a local surrogate model in the neighbourhood of the explained prediction. It does so by generating new data points in the proximity of the example in question and then training an interpretable model section 2.3 on them. In the current implementation and therefore in our project the interpretable model is a sparse linear regression model.

[2]claim: *V České republice osvobodili před trestem smrti Vladimíra Lulka.* = *In the Czech Republic, Vladimir Lulek was freed from the death penalty.*; label : REFUTES

[3]Both XLM-RoBERTa model finetuned on the NLI-related SQuAD2 downstream task section 3.3.2

We used its already mentioned library[4], which provides implementation for explaining text classification as well as visualization options. We only needed to make minor changes to the library to fit our input and allow the changes of some parameters. We tried various parameter settings, which we shall now go through and present our findings. All following examples will use word tokenization level[5], which we primarily focused on[6].

Volatility of explanations. As we wrote in section 2.4.1, one of the main disadvantages of LIME is its instability. This deficiency is well illustrated in fig. 4.2, which shows two explanations created with the exact same settings, but a different random seed.[7] The difference does not disappear even with a substantial increase in generated samples. However, as is illustrated in fig. 4.3, it can be diminished.


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

Figure 4.2: Explanations with the same parameters but a different seed with 1000 generated samples
LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weighst, tg = True

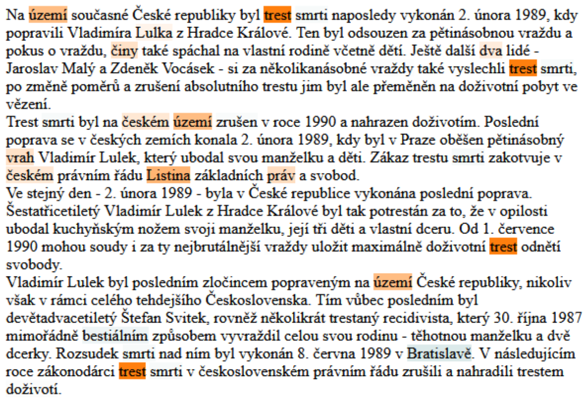
Kernel width. We used an exponential kernel function in which the kernel width parameter (σ) had a significant effect on the explanation. Generally, $\sigma \in [0.2, 0.35]$ produced comprehensible and reasonably stable explanations. Out of this bound, even usually even small changes had a non-trivial impact. Figure section 4.3.1 demonstrates a difference between kernel width 0.35 (fig. 4.4a) and 0.4 (fig. 4.4b).

[4]`https://www.lime-ml.readthedocs.io/`

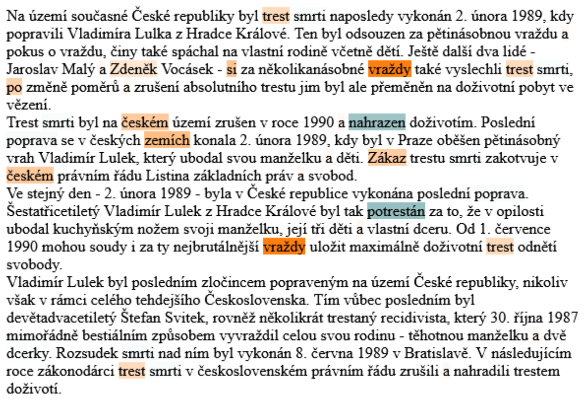
[5]LIME implementation by default tokenizes the text into words. The method automatically removes non-word characters from the feature space and returns them to the modified version of the text before it passes it to the model. However, for the Czech text, it considers all diacritics to be non-word characters and the resulting tokenization was an unreasonable mixture of words and sub-words. Therefore we used a wrapper around nltk word tokenizer - `https://www.nltk.org/` (BKL09), which works well also for Czech texts.

[6]Unlike in SHAP for reasons outlined in section 4.3.3

[7]We generated 5 random numbers from 0 to 2000. Then we created an explanation by LIME with the same settings just a different random seed. And we present two, most distinct explanations.



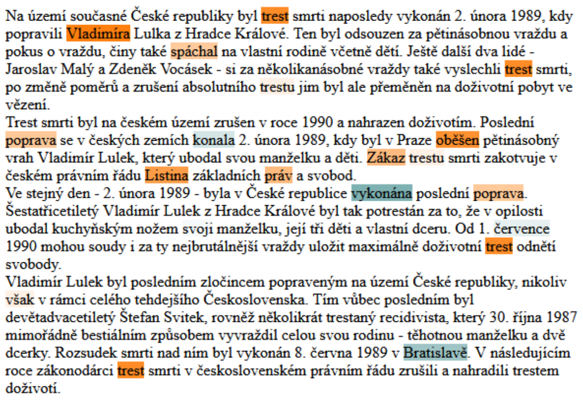
Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

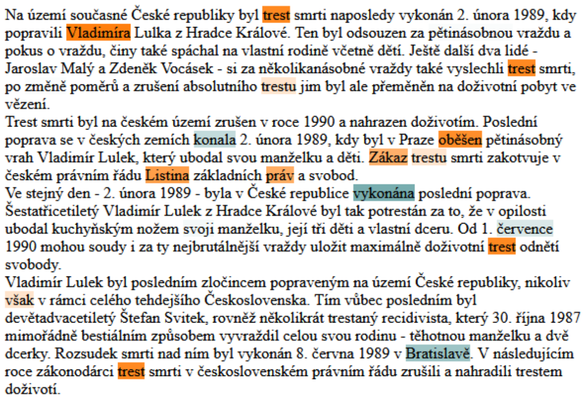
Figure 4.3: Explanations with the same parameters but a different seed with 5000 generated samples
LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, tg = True
We used the same seeds as in fig. 4.2. Compared to experiment with 1000 samples, now the methods at least both refer to words "trest" = punishment and "Českém" = czech, with reference to area. From our point of view, the right explanation might be less helpful than when it was created with 1000 samples because the word "oběšen" = hanged is not highlighted anymore. This further demonstrates the volatility of LIME explanations, especially with a lower number of samples.

We were testing each of the following scenarios with a range of different kernel widths and we shall always present the best one.



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : kernel width 0.35



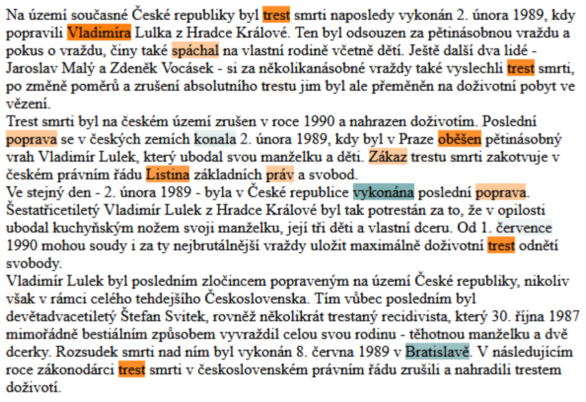
Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(b) : kernel width 0.4

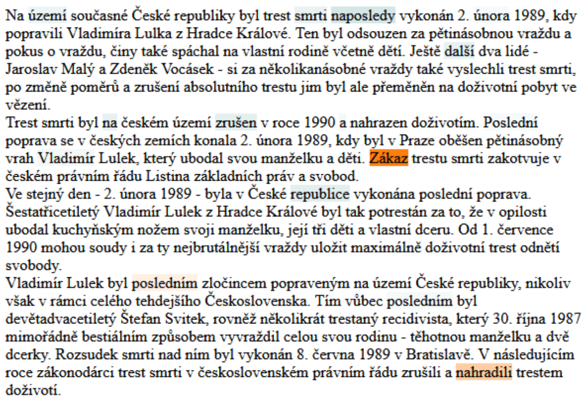
Figure 4.4: Difference in kernel width
LIME explanations with word tokenization, exponential kernel function with cosine distance, explaining 15 significant features selected by highest-weights, tg = True, 5000 samples
This particular instance does not show significant differences, however explanation with $\sigma = 0.4$ is completely missing the word "poprava" = execution. Explanations for the $\sigma \in [0.2, 0.35]$ usually contain the same words and only vary in weight.

Token grouping. In the text, there can be multiple occurrences of a token. We can either treat each token as a unique entity - based on its position, or we can group them the same tokens together. In the later case, the generated text perturbations will either contain all instances of the word or none. And the result will contain one weight for the word in all of its occurrences. In the method description, we refer to this property as *token grouping* (tg = True if all instances of a token should be treated as one).

We expected the LIME to yield better explanations with a unique assessment of tokens (tg = false) because the role of a token can vary depending on the context. More importantly, it could be a distraction and lead the human eye to the word's insignificant occurrences. However, as is presented in fig. 4.5, the method performed much better with the property on (tg = true) - practically with no exceptions.


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : token grouping = True


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

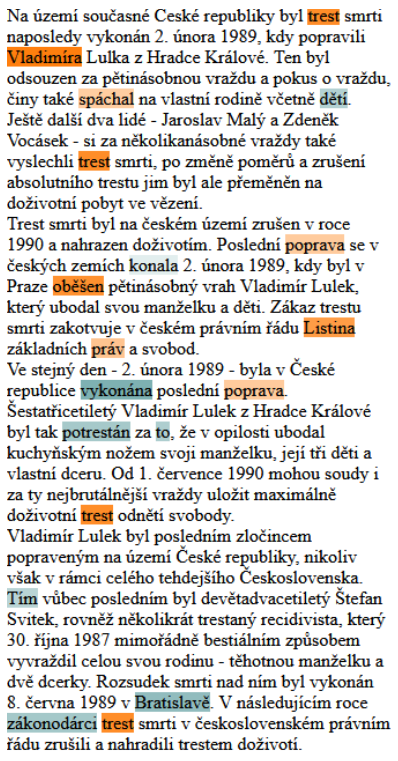
(b) : token grouping = False

Figure 4.5: Difference in token grouping
LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features selected by highest-weights, 5000 samples
Without token grouping LIME highlights only several words (caused by the limit in explanation size), which are often marked as opposed to the decision and the majority is not very informative. In this case, the only words find to support the decision were "Zákaz" = ban, "nahradili" = replaced, and "posledním" = last.

Feature selection. LIME locally replaces our complex neural model with a surrogate - linear regression model section 2.3, which is intuitively interpretable because for each feature we see a weight. What is however crucial for its intuitive interpretability is sparsity. When tokenizing the input text to words, our feature space is extended and it would be easy to get lost in the number of weights. In order to introduce sparsity, we use a feature selection algorithm.

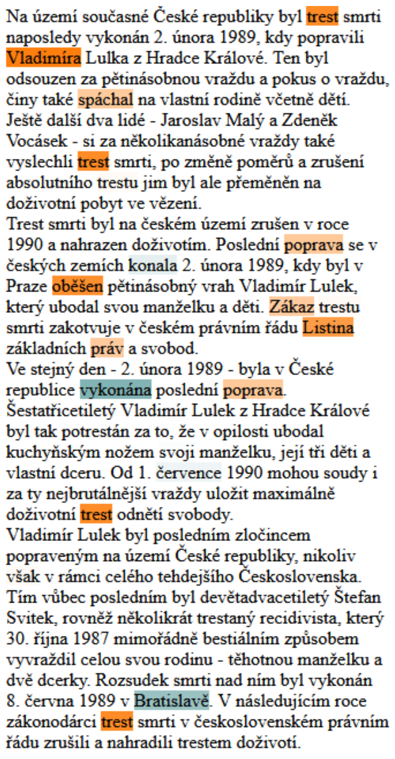
In figure fig. 4.6, we present the results of different feature selection algorithms we tested - Lasso, forward feature selection and selection by the highest weights. The best results were from our point of view generally, as well as for this sample achieved when using features with the highest weights. We expected better results from Lasso because it is a frequently used and bench-marked feature selection algorithm (TLM10). However, for our textual inputs, it produced explanations with too many tokens and not enough sparsity, which made them less readable.

Forward feature selection algorithm produced typically similar (sometimes the same) explanations as highest weights. However, as is shown in table 4.1, when using forward feature selection it took on average 13.23% longer to compute the explanation, therefore we overall preferred the highest weight algorithm.



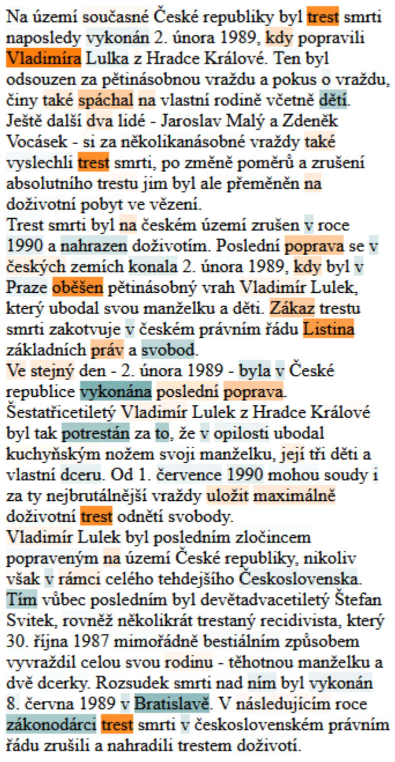
Na území současné Ceské republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : Forward selection



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(b) : Highest weights



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(c) : Lasso

Figure 4.6: Feature selection algorithms
LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features (when applicable) , tg = True, 5000 samples
All explanations are comparable with the same most significant words, however, Lasso produces the most noise. Explanation created with the highest weights feature selection is most expressive and from our point of view also most readable.

	Forward selection	Highest weights	Lasso
Time per instance [s]	25.013	22.091	22.14

Table 4.1: Mean execution time for LIME with different feature selection algorithms
LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features, tg = True, 5000 samples. Calculated on 50 random instances from CTKFacts dataset.

Distance function. So far all examples used Cosine distance.[8] Compared to other tested distance functions, it consistently created the most sensible explanations. Figure fig. 4.7 shows an example with Cosine, Euclidean and Manhattan distances. It is however worth mentioning that Lasso feature selection algorithm did not work at all with Euclidean distance (all features had the same values).

Selected parametrization

The best parametrizations were compared together with Text augmented LIME (section 4.3.2) in series of short pairwise blinded tests (described in section 4.2). For LIME we nominated the following parametrizations:

CsFEVER. (results of the tests can be found in appendix C.1)

- *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
- *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, features selected by Lasso, 5000 samples, tg = True*
- *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

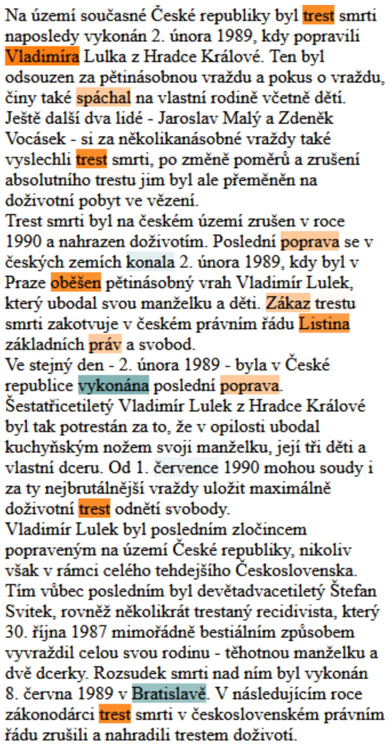
The best selected parametrization (used in the experiment) is however none of the above listed because it was produced by Text augmented LIME which is described in the next section.

CTKFacts. (results of the tests can be found in appendix C.2)

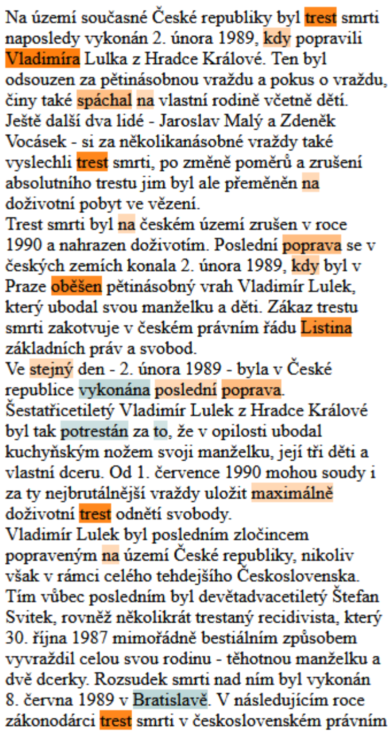
- ***LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True***
- *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

The best selected parametrization (used in the experiment) is : *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

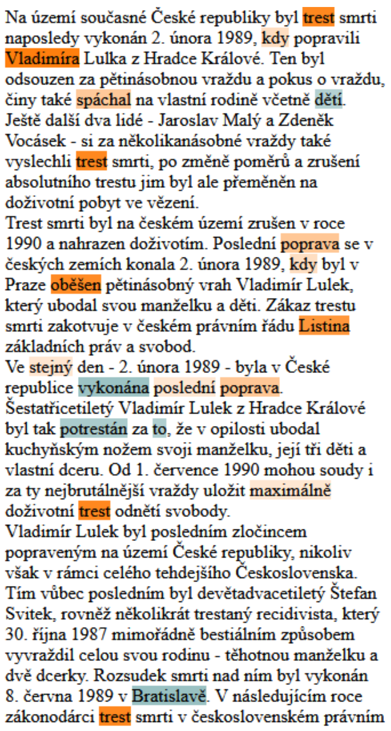
[8]Distance function is used to measure distances between original and perturbed instances in eq. (2.4).


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí.
Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa.
Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : Cosine


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí.
Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa.
Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním

(b) : Euclidean


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí.
Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava.
Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa.
Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním

(c) : Manhattan distance

Figure 4.7: Distance function
LIME explanations with word tokenization, exponential kernel function with σ = 0.3, explaining 15 significant features selected by highest-weights, tg = True, 5000 samples
In this case, all explanations are comparable, but Euclidean and Manhattan distances produce noisier explanations. (In this example e.g. words "na", "kdy", "stejný").

4.3.2 Text augmented LIME

The text implementation in the current version works as described in the theoretical section 2.4.1. LIME splits the text into tokens and the neighbourhood data points are generated by erasing random tokens from the original sequence.

We thought that there is a space for improvement in this approach. Especially in the representation of tokens - a token (word) is expressed as a binary value. And new variations of the data are generated by setting the word as present (value 1) or missing (value 0).

Rather than as binary data, we prefer to approach tokens as numerical values, which, can be changed to a different (plausible) value, not only to 0.

For comparison, for tabular data, LIME creates the perturbed input samples by replacing some feature values with a new value - randomly drawn from a normal distribution with the mean and standard deviation calculated from the feature values in the training dataset.

We cannot apply the same technique as LIME does for tabular text because we do not have a fixed feature space. However, we suggest that instead of removing a word, we could replace it with a similar word.

The key is to represent a token as a numerical value. In contrast to tabular data, the numerical value of a word is not a scalar but a vector. Such vector representation of a word is called word embedding.

We propose the following approach based on word similarity:

1. We tokenize the original text sequence s to tokens T, using a word level tokenization.
2. We randomly draw $p * |T|$ words W, where p is a proportion of tokens which should be replaced (in original text LIME deleted).
3. For each word $w \in W$ (to be mutated words) :
    a. We generate m words similar to w using and external language model.
    b. From the generated words, we draw a word *w'*. We draw *w'* in random with probabilities proportional to the similarity of each generated word the original word w
    c. We replace w by *w'* in the original text sequence s and get mutated text sequence *s'*.
4. To further use LIME, we also calculate the distance d, between the original text s and the mutated text *s'* as $d = D(s, s')$ where

   D is a defined distance function; vector $s = [1]^{|T|}$; vector $s' = [w \in T : SF(w, w')]$; SF is a defined similarity function
5. continue in LIME for text execution

One advantage is that we shall only change well-defined segments in the process. We can therefore reuse the implementation of LIME for text.

The main implementation decision is in 3.a, in generating similar words. For this purpose we use a fastText model (BGJM16) - a language model for text representation which operates on a character level and represents a word as a sum of the character n-gram vectors.[9]

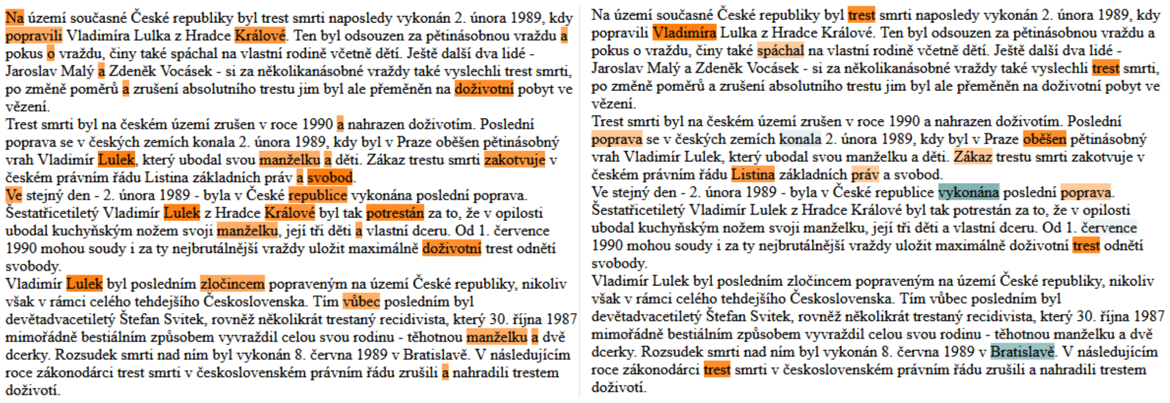
In particular, we are using a Czech fastText model presented in (GBG+18), trained on Wikipedia and Common Crawl[10] corpora. We use cosine similarity

[9]We also experimented with the Word2vec (MCCD13) model which operates on a word level and embeds words based on the context of neighbouring words. However, Word2vec cannot produce embeddings for out-of-vocabulary words (words not present in the training data) and therefore it was not suitable for our use case. (fastText is usable even with new words as long as they had at least one character ngram present in the training data.)

[10]`https://commoncrawl.org/`

as SF and (as for basic text LIME) test different distance functions D.

Results



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : Text augmented LIME, $p = 0.3$

Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení.
Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod.
Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody.
Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(b) : Traditional LIME

Figure 4.8: Text augmented LIME traditional text LIME

Both LIME methods use: exponential kernel function with cosine distance and $\sigma = 0.3$*, explaining 15 significant features selected by highest-weights, tg = True, 5000 samples*
In comparison with basic LIME, text augmented LIME often assigns high value to conjunctions or prepositions, expecially those with only one letter (e.i. a, i, u, v, k). In this example it is presented by words "a", "o", "na". The explanation is not bad, it contains the name of the person of interest ("Lulek"), refers to the activity in question "popravili" = executed, "potrestán" = punished and there are references to the area searched "Králové" (a part of a name of a Czech city), "republice" = republic. It however also contains lots of noise.

We tested the implementation with different parameter settings (as with text LIME section 4.3.1) and the produced explanations are quite unstable. Even more volatile compared to traditional text LIME. Sometimes, the generated explanation works as intended and indeed marks words related to the context. However, often the method produces a seemingly incoherent explanation based on prepositions and conjunctions instead (fig. 4.8).

We suspect that the reason could be the fastText model used because as can be seen in figure fig. 4.9, the similarities between words learnt by the model can be unreasonable or even discriminatory.

We have noticed, that greater kernel width ($\sigma = 0.5$) in combination with low p (e.g. 0.2) occasionally produced very helpful explanations. These results were however one of the least stable explanations, we would therefore not recommend this combination for production. Instead would prefer $p[0.2, 0.5]$ and kernel width $\sigma \in [0.2, 0.35]$ which produced steadily good results. It is also worth mentioning, that text augmentation LIME performed well only with Cosine distance and a highest-weights or forward feature selection algorithm.

Word	Cosine similarity
Trest	0.741931
tresty	0.739696
potrest	0.710362
trestem	0.636579
podmíečný	0.630155
trestu	0.625051
trest.	0.609836
postih	0.603276
doživotí	0.585734
odpykat	0.572743

(a) : "trest" = punishment

Word	Cosine similarity
nebyl	0.815657
Byl	0.763696
býval	0.735082
měl	0.718258
mohl	0.684727
Nebyl	0.67833
stal	0.657446
musel	0.650827
zůstal	0.639246
býval	0.639237

(b) : "byl" = was

Word	Cosine similarity
,	0.635318
i	0.565356
či	0.470163
ฯ勗	0.461585
já	0.461199
4Ra	0.449055
ケ	0.446649
ᓚ広	0.446437
ajá	0.44569
瞿	0.445167

(c) : "a" = and

Word	Cosine similarity
islám	0.744493
Islámu	0.654754
Islamismus	0.632659
islamismus	0.62812
al-Islám	0.594323
islámu	0.593809
al-islám	0.58701
Terorismus	0.586763
Muslimové	0.561179
Islámský	0.560412

(d) : "Islám" = Islam

Figure 4.9: Most similar words and Cosine similarities to their original word
For several examples, we generated the 10 most similar words with the fastText model used in the project to demonstrate some of its weaknesses. There are many ordinary reasonable embedding (such as in example fig. 4.9a). However, e.g. for the word "byl" = was fig. 4.9b the closest term found is "nebyl" = "was not" and while this replacement would completely invert the meaning of the sentence, the similarity between these two words is higher than the similarity between the same word with its first letter in capital.
For many one-lettered words, such as "a" = and (fig. 4.9c), many foreign characters are found, which do not make sense in the Czech context.
Finally, we added fig. 4.9d, which demonstrates how the model is biased and discriminatory since the 8^{th} closest word found for "Islám" = Islam is "Terorismus" = Terrorism.

Compared to traditional text LIME, text augmentation LIME performed better on CsFEVER than on CTKFacts. We believe that it could be caused by the fastText model used, which was trained on Wikipedia corpus and therefore it was a better fit for the lingo in CsFEVER.

Execution time. The execution time of text augmented LIME was higher than of traditional text LIME. This is not a surprise, since retrieving closest embeddings is an expensive operation, which we repeat many times ($p * |T|$) during the course of the method execution. Figure 4.10 demonstrates how execution time grew with increasing proportion of words mutated.

From our exploration as well as conducted experiments chapter 5 we deduce that text augmented LIME is a promising variant of traditional text LIME. While the method currently yields explanations of fluctuating quality, for future work we would recommend testing it with a different language model because the unstable performance could be caused by the unreasonable embedding model.

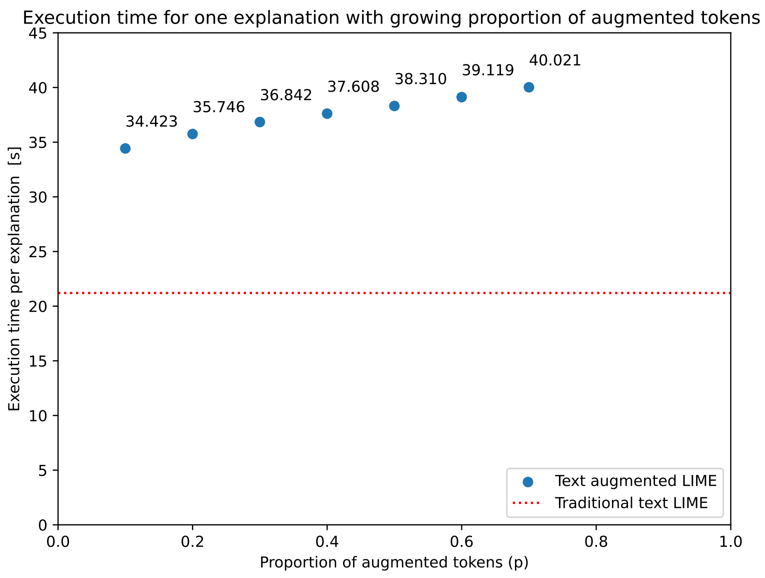


Figure 4.10: Execution time of Text augmented LIME with varying proportion of words mutated in comparison to traditional LIME
Text augmented LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features selected by highest-weights, tg = True, 5000 samples, p = 0.3 in comparison with *LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features selected by highest-weights, tg = True, 5000 samples* Calculated on 50 random instances from CTKFacts dataset. Time per instance.

Selected parametrization

The best parametrizations were compared together with best traditional text LIME parametrizations in series of short pairwise blinded tests (described in section 4.2). For Text augmented LIME we nominated the following parametrizations:

CsFEVER. (results of the tests can be found in appendix C.1)

- ***Text augment LIME explanations with p = 0.3, word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features selected by forward-pass, 5000 samples, tg = True***
- *Text augment LIME explanations with p = 0.4, word tokenization, exponential kernel function with cosine distance and σ = 0.25, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

The best selected parametrization (used in the experiment) is *Text augment*

LIME explanations with p = 0.3, word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True.

CTKFacts. (results of the tests can be found in appendix C.2)

- *Text augment LIME explanations with p = 0.3, word tokenization, exponential kernel function with cosine distance and σ = 0.25, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
- *Text augment LIME explanations with p = 0.4, word tokenization, exponential kernel function with cosine distance and σ = 0.2, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

The best selected parametrization (used in the experiment) is however none of the above listed because it was produced by traditional text LIME.

4.3.3 SHAP

For SHAP (section 2.4.3), we used its official library [11] but we could not use it directly because it did not work for our input format.[12]

Instead, we forked the project and used our, edited, version. The development process was challenging since the library is extensive and various (although usually small) changes were required in different files and methods. We, therefore, spent a considerable amount of time on making SHAP work with our use case.

The library itself does not offer such a variety of parameters as LIME does. This arrives from the difference between the methods themselves. As is described in section 2.4.1 and visible in eq. (2.3) or eq. (2.4), LIME leaves a lot of space for further specification (distance function, kernel function, even interpretable model could be changed). SHAP, on the other hand, is strictly defined and does not leave much for parametrization. Therefore, we did not perform such a thorough search of the best parametrization settings as we did with LIME.

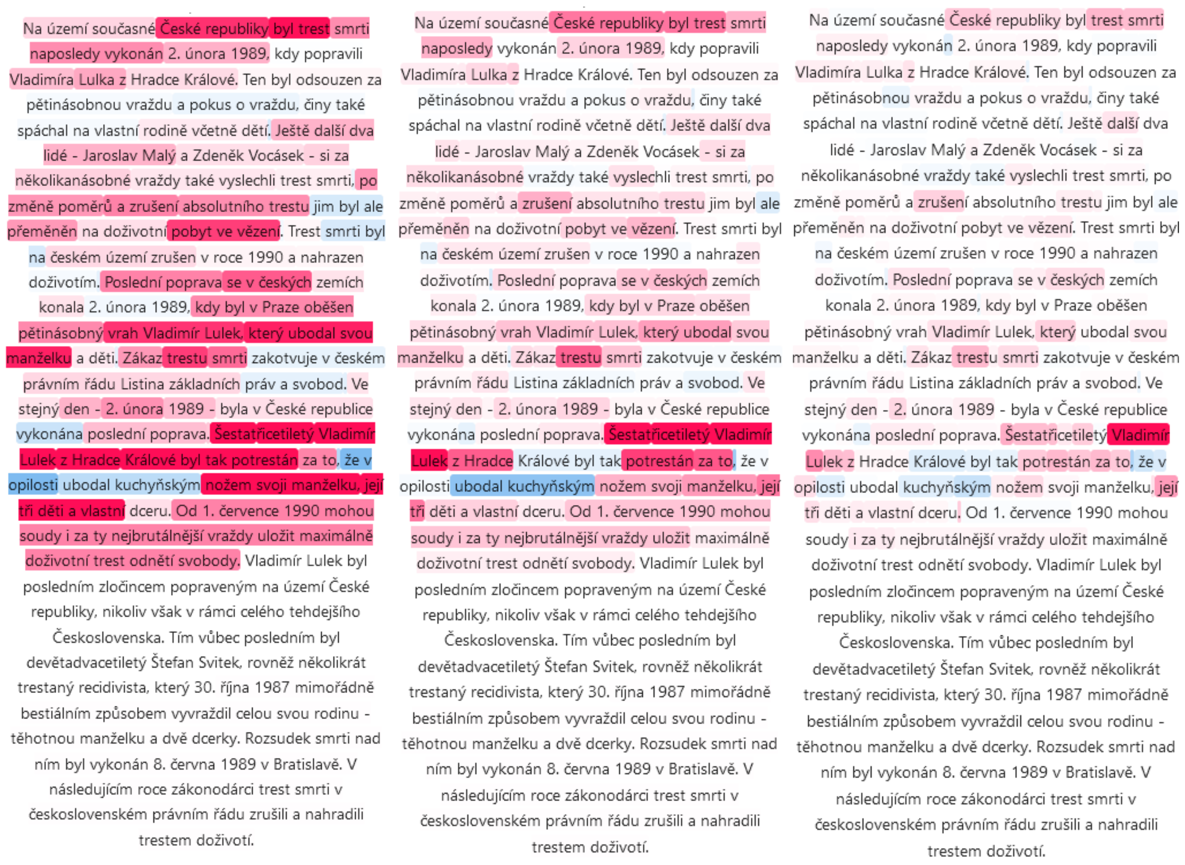
There were only three parameters to consider - the tokenization level, masking and the approximation algorithm. Here is where we expected to compare different SHAP options because the library offers several estimation techniques including Kernel SHAP and Deep SHAP. Nevertheless, none of these advanced approaches is available for text input.

Both Deep SHAP and Kernel SHAP require, for the estimation of SHAP values, reference feature values - mean feature values computed from the

[11] `https://www.shap.readthedocs.io/`

[12] Our input consists of two texts, they are both necessary for prediction but only one of them should be used in the explanation (section 4.1.2).

training data (or its subset). This is not possible for text input which does not contain a fixed feature set.



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení. Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod. Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody. Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : 500 evaluations

Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení. Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod. Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody. Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(b) : 1000 evaluations

Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení. Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod. Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody. Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(c) : 5000 evaluations

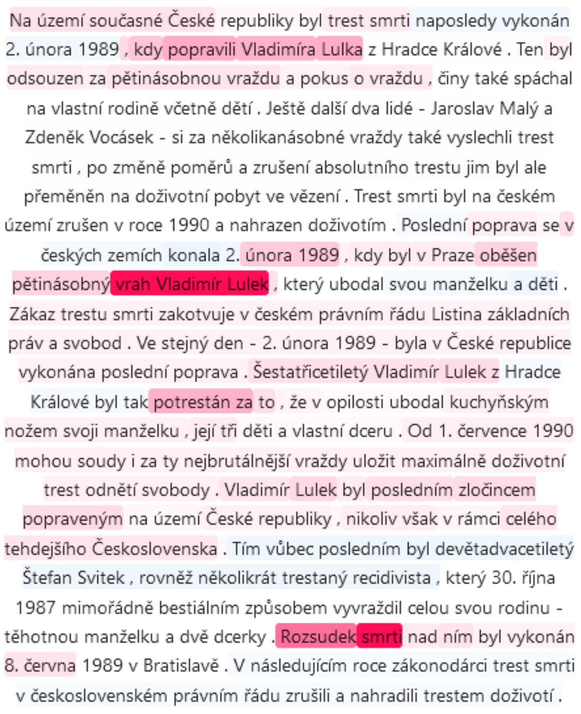
Figure 4.11: Increasing number of evaluations WordPiece tokenization
SHAP with WordPiece tokenization and model mask
The text is tokenized into sub-words. The partitioning algorithm uses hierarchical clustering to assign tokens into groups. In case of a low number of evaluations (fig. 4.11a), there are fewer bigger text chunks which are assigned the same Shapley value. With the growing number of evaluations, the algorithm gets to the granularity of specified tokens (fig. 4.11c). From our point of view, subword granularity is too fine and we get better results when we use fewer evaluations, in this case, 1000 (visible in fig. 4.11b).

Partition explainer. We used a partition explainer, which computes Shapley values recursively via a hierarchy of features that defines the feature coalitions. The method uses hierarchical clustering to group features which are then treated as one unit and the same contributions are assigned to all features in the group. As a result, the explanations created with a low number of executions of the model are continuous - tokens close to each other have a similar value. And with the increasing number of evaluations, the partition tree is deeper and the explanations become more granular. (Meaning that if we chose a word tokenization, then there will be significant differences on the level of words.) This is well illustrated in figure fig. 4.11 where we increase the number of evaluations performed.

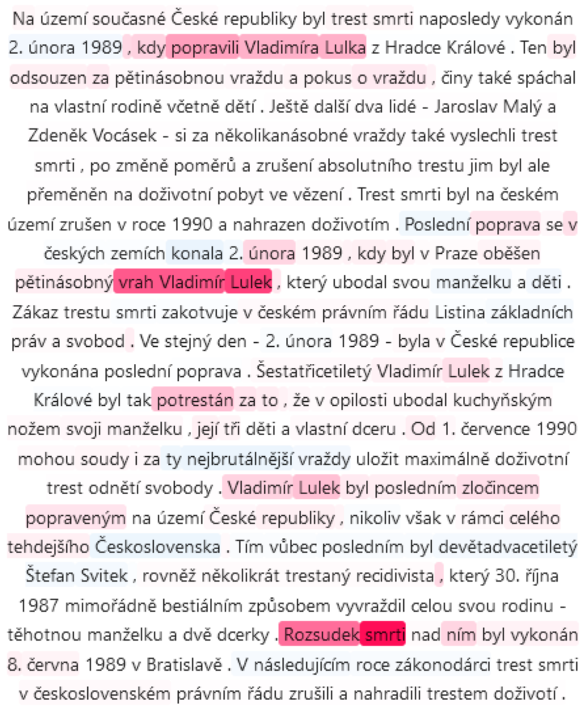
The runtime of the exact algorithm is quadratic to the number of features which is a great advantage compared to the exponential complexity of the Kernel SHAP (section 2.4.3) or the Shepley sampling values section 2.4.2. However, unless tokenizing on a very coarse level, such as sentences, we not going to compute true Shapley values but only their approximation.[13]

It is worth mentioning that the effect of the increasing number of evaluations was lower for coarser tokenization levels (comparison of fig. 4.11 and fig. 4.12). It is logical because for fewer features we need fewer evaluations to compute their exact value. Therefore, considering how the partitioning explainer works, the changes in the explanation will be diminishing with the increasing number of evaluations.

[13]The parameter which we describe as a number of evaluations is actually the maximum number of evaluations (max evals). If the algorithm needs fewer executions to compute the exact values, then it will only perform the evaluations necessary.


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989 , kdy popravili Vladimíra Lulka z Hradce Králové . Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu , činy také spáchal na vlastní rodině včetně dětí . Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti , po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení . Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím . Poslední poprava se v českých zemích konala 2. února 1989 , kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek , který ubodal svou manželku a děti . Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod . Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava . Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to , že v opilosti ubodal kuchyňským nožem svoji manželku , její tři děti a vlastní dceru . Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody . Vladimír Lulek byl posledním zločincem popraveným na území České republiky , nikoliv však v rámci celého tehdejšího Československa . Tím vůbec posledním byl devětadvacetiletý Štefan Svitek , rovněž několikrát trestaný recidivista , který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky . Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě . V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí .

(a) : 500 evaluations


Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989 , kdy popravili Vladimíra Lulka z Hradce Králové . Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu , činy také spáchal na vlastní rodině včetně dětí . Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti , po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení . Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím . Poslední poprava se v českých zemích konala 2. února 1989 , kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek , který ubodal svou manželku a děti . Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod . Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava . Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to , že v opilosti ubodal kuchyňským nožem svoji manželku , její tři děti a vlastní dceru . Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody . Vladimír Lulek byl posledním zločincem popraveným na území České republiky , nikoliv však v rámci celého tehdejšího Československa . Tím vůbec posledním byl devětadvacetiletý Štefan Svitek , rovněž několikrát trestaný recidivista , který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky . Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě . V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí .

(b) : 1000 evaluations

Figure 4.12: Increasing number of evaluations word tokenization

SHAP with word tokenization and model mask
With coarser granularity of tokens, the differences between explanations diminish. For some explanations, these two settings produced completely the same results. This explanation well demonstrates the bias (most likely in the model), which could also confuse the user. The most significant phrases found are "vrah Vladimír Lulek" = murderer Vladimír Lulek and "Rozsudek smrti" = death sentence. The second term is however in the article used in reference to a different person and while provides context what the article is about, it does not contain any information about the truthfulness of the claim. Nevertheless, the next significant term is "popravili Vladimíra Lulka" = "they executed Vladimir Lulek", which leads us back to the right track.

Tokenization level. Figure 4.11c demonstrates well that sub-words are not very suitable for interpretability purposes. It benefits from the grouping SHAP makes for a fewer number of evaluations because then the text units are more coherent. In comparison, we reached steadily good results with word tokenization (fig. 4.12).

In fig. 4.13 we can see the results of the coarsest tokenization tested - sentence and sub-sentence. We started with MorphoDiTa sentence tokenizer for the Czech language from (SSH14), whose output we wrapped, such that it worked with SHAP. Its results were pleasing. In many cases, the explanation contained only one sentence which[14] held all the crucial information. However, in other cases, the explanation contained most of the original text. That is mainly because the articles use primarily compound sentences. Therefore, next, we implemented a sub-sentence tokenizer, which further splits the outcome of the sentence tokenizer on conjunctions, which are in the Czech language used between sentences.[15] The results are presented in fig. B.1b and we consider it to be the best explainer we found.

Nevertheless, we did not include this parametrization in the experiments. The experiments contain only parametrizations with word tokenization because we wanted to ensure blinding and for text augmented LIME (which proved to be the best variant for LIME for CsFEVER dataset) we could not use any other tokenization level then word tokenization[16].

Selected parametrization

The following parametrizations were compared in series of short pairwise blinded tests (described in section 4.2) to find best parametrization for the experiments.

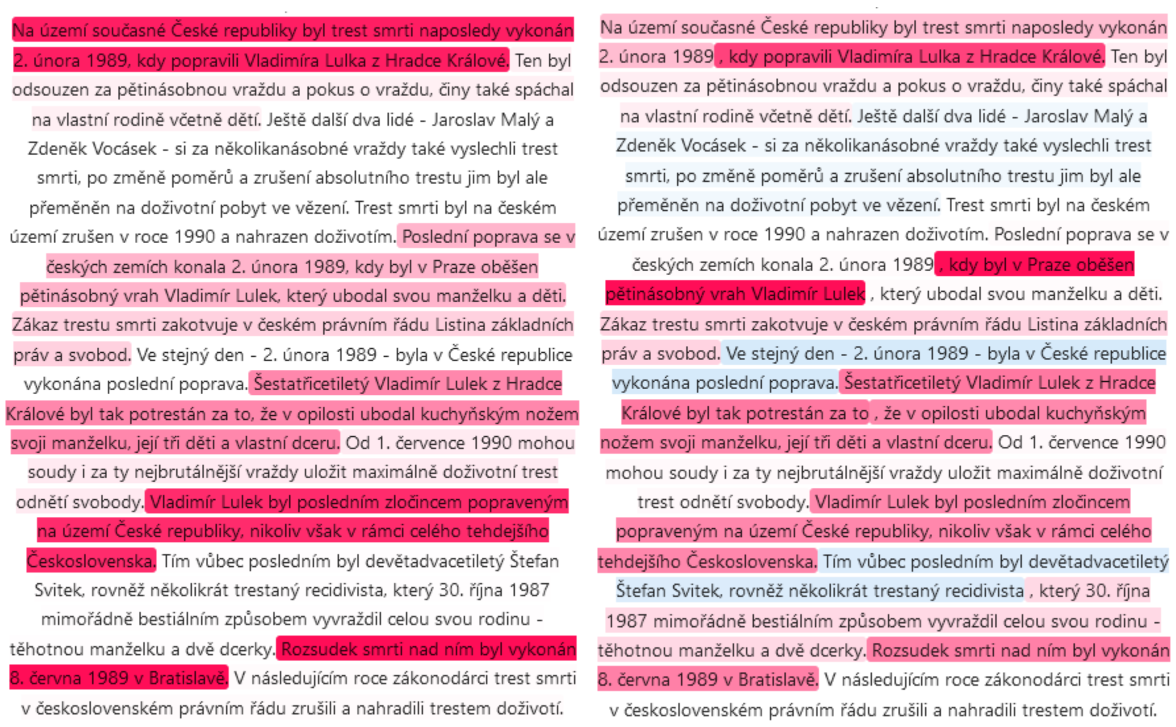
CsFEVER. (results of the tests can be found in appendix C.1)

- *SHAP with word tokenization, model mask, 1000 maximal evaluations*
- ***SHAP with word tokenization, model mask, 5000 maximal evaluations***

[14]SHAP calculates values for all tokens, therefore the explanation contained all sentences but only to one of them assigned a significant value.

[15]Our pragmatic approach turned out to work well but it does not separate clauses which are connected by a comma or conjunction often used to separate words (e.g. "a" = and). We would prefer to tokenize compound sentences into individual clauses. However, that is a complex linguistic task not within the scope of this thesis.

[16]For text augmented LIME, WordPiece tokenization did not make much sense because the majority of the "words" created from the mutated sub-words did not exist in the Czech language. We would like to test text augmented LIME with the sentence and sub-sentence tokenization, however, it is a complex task, out of the scope of this work. It would require a sentence embedding model and a base of similar sentences.



Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989, kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení. Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989, kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek, který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod. Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to, že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody. Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista, který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(a) : Sentence SHAP

Na území současné České republiky byl trest smrti naposledy vykonán 2. února 1989 , kdy popravili Vladimíra Lulka z Hradce Králové. Ten byl odsouzen za pětinásobnou vraždu a pokus o vraždu, činy také spáchal na vlastní rodině včetně dětí. Ještě další dva lidé - Jaroslav Malý a Zdeněk Vocásek - si za několikanásobné vraždy také vyslechli trest smrti, po změně poměrů a zrušení absolutního trestu jim byl ale přeměněn na doživotní pobyt ve vězení. Trest smrti byl na českém území zrušen v roce 1990 a nahrazen doživotím. Poslední poprava se v českých zemích konala 2. února 1989 , kdy byl v Praze oběšen pětinásobný vrah Vladimír Lulek , který ubodal svou manželku a děti. Zákaz trestu smrti zakotvuje v českém právním řádu Listina základních práv a svobod. Ve stejný den - 2. února 1989 - byla v České republice vykonána poslední poprava. Šestatřicetiletý Vladimír Lulek z Hradce Králové byl tak potrestán za to , že v opilosti ubodal kuchyňským nožem svoji manželku, její tři děti a vlastní dceru. Od 1. července 1990 mohou soudy i za ty nejbrutálnější vraždy uložit maximálně doživotní trest odnětí svobody. Vladimír Lulek byl posledním zločincem popraveným na území České republiky, nikoliv však v rámci celého tehdejšího Československa. Tím vůbec posledním byl devětadvacetiletý Štefan Svitek, rovněž několikrát trestaný recidivista , který 30. října 1987 mimořádně bestiálním způsobem vyvraždil celou svou rodinu - těhotnou manželku a dvě dcerky. Rozsudek smrti nad ním byl vykonán 8. června 1989 v Bratislavě. V následujícím roce zákonodárci trest smrti v československém právním řádu zrušili a nahradili trestem doživotí.

(b) : Sub-sentence SHAP

Figure 4.13: Increasing number of evaluations word tokenization

SHAP with model mask and 1000 maximal evaluation

The best selected parametrization (used in the experiment) is *SHAP with word tokenization, model mask, 5000 maximal evaluations.*

CTKFacts. (results of the tests can be found in appendix C.2)

- *SHAP with word tokenization, model mask, 500 maximal evaluations*
- ***SHAP with word tokenization, model mask, 1000 maximal evaluations***

The best selected parametrization (used in the experiment) is *SHAP with word tokenization, model mask, 1000 maximal evaluations.*

Chapter 5

Experiments

In this chapter, we are going to compare explainability methods (which we presented in section 4.3). We will now describe the design and methodology used for our experiments and later present their results.

5.1 Design

The one objective we essentially want to achieve with this project is to enable people to find the key information in text as effortlessly and quickly as possible. Therefore, we decided to design our experiments with respect to this objective and measure the contribution of the explanation for the end-user.

Since the task of helping human fact-checkers verify claims relies on human-computer interaction, we use human-based evaluation, which is well suited for such tasks, as we explained in section 2.5. The strongest evidence would be provided with an application-grounded evaluation. However, conducting such an experiment would require engaging the target audience - journalists, and we do not have the resources necessary for that.

Therefore, we are going to administrate human-ground evaluation and conduct experiments with non-professional human subjects.

Binary forced-choice experiment. On both, CsFEVER and CTKFacts datasets, we will conduct an experiment in which a human subject is presented with two explanations for a given claim - one generated by best parametrized LIME and one by best parametrized SHAP, its correct label and is asked to choose which explanation is more helpful.

5.1.1 Binary forced choice experiment

In this experiment we are going to compare how helpful explanations produced by SHAP and LIME methods are to the human audience.

Design decisions

Choice of method settings. Since the experiment requires human annotators (and thus is demanding to administer), we decided to conduct it only once for each dataset - with a single, best applicable method setting from both LIME and SHAP. We also wanted the test to be blinded, therefore we had to decide on a common token granularity for both these methods.[1] with us as annotators.

We decided to use word granularity because it provided reasonable but varying results and it could be used by all tested methods.

Next, we had to choose one, best, parameter setting for both methods (and word tokenization), such that we could compare them with each other in the experiment.

We realise, that a robust solution would be to perform this experiment repeatedly for each explanation method, such that we could compare different parameter settings until we found the best one. However, that was not a feasible solution due to the capacity of annotators. Therefore, instead, we use procedure described in section 4.2, which is basically a small and less formal version of this experiment[2].

Sampling. Once we established which methods and their settings we would use, we were able to use them to compute explanations. We decided to focus only on verifiable correctly predicted examples, i.e. we filtered out data points for which the model to be explained predicted an incorrect label or the correct label was NEI. For each dataset, we randomly shuffled such explanations, divided them into multiple groups, and passed each to a different annotator. Each example would therefore be evaluated only once. This way we can cover more examples (and get more conclusive results). However, it comes at the cost of the ground truth being based on one opinion. Furthermore, it made us unable to investigate whether annotators generally agreed on the same data points and as such better judge their objectivity.

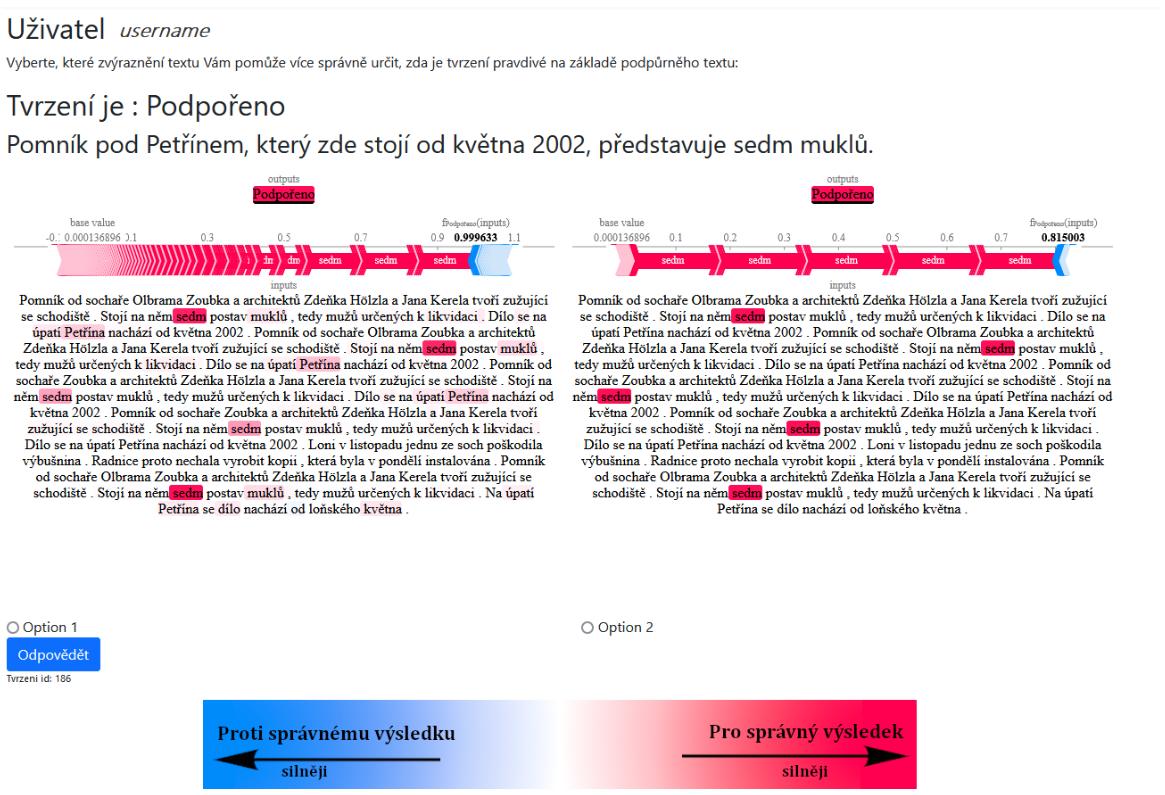
Realization

To conduct the experiment, we created a simple annotation platform which is shown in fig. 5.1.

The annotator is presented with a claim, its predicted label and two explanations side by side. And they are asked to choose which explanation they find

[1]An annotator would probably notice a difference in explanations and after a few examples, they could decide based on experience, or their preference for a granularity level could decide instead of the suitability of the whole explanation.

[2]We use the same settings and annotation platform as for the experiment itself (section 5.1.1)



Uživatel *username*

Vyberte, které zvýraznění textu Vám pomůže více správně určit, zda je tvrzení pravdivé na základě podpůrného textu:

Tvrzení je : Podpořeno

Pomník pod Petřínem, který zde stojí od května 2002, představuje sedm muklů.

Pomník od sochaře Olbrama Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Pomník od sochaře Olbrama Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Pomník od sochaře Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Pomník od sochaře Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Loni v listopadu jednu ze soch poškodila výbušnina . Radnice proto nechala vyrobit kopii , která byla v pondělí instalována . Pomník od sochaře Olbrama Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Na úpatí Petřína se dílo nachází od loňského května .

Pomník od sochaře Olbrama Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Pomník od sochaře Olbrama Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Pomník od sochaře Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Pomník od sochaře Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Dílo se na úpatí Petřína nachází od května 2002 . Loni v listopadu jednu ze soch poškodila výbušnina . Radnice proto nechala vyrobit kopii , která byla v pondělí instalována . Pomník od sochaře Olbrama Zoubka a architektů Zdeňka Hölzla a Jana Kerela tvoří zužující se schodiště . Stojí na něm sedm postav muklů , tedy mužů určených k likvidaci . Na úpatí Petřína se dílo nachází od loňského května .

Figure 5.1: Annotation platform. An annotator is presented with claim, its label and two explanations side by side and is asked, which they find more helpful to guide them to the correct label.

more helpful to correctly decide whether the claim is supported or refuted by the background text. Choosing neither is not an option, user must always pick one or the other.[3]

The explanation itself has a form of highlighted background text. Not to influence the annotator with different visualization forms, we first unified the visualization to text plot from SHAP. An example of such a transformation for LIME explanation can be found in the appendix. (appendix D.1)

For each example, we randomly assign right and left position to SHAP and LIME, such that the annotator cannot rely on the explanation position.

The answer is recorded with the name of preferred method.

[3]If the explanations were exactly the same or the data point was corrupted and the background text did not contain the information necessary, the annotator marked the instance id and we removed the instance from the evaluated data.

Formal assignment

We denote a discrete random variable $x =$ number of times SHAP produced more helpful explanation than LIME.

The random variable x then has a binomial distribution $x \backsim \mathcal{B}(n, \pi)$ (Joh16).

n is number of independent experiments. In our case number of evaluated claim-context pairs.

π is the population parameter and corresponds to the probability of SHAP producing more helpful explanation in any trial.[4]

To determine, whether we can conclude that either SHAP or LIME produce better explanations more often, we are going to perform two exact one-tailed binomial hypothesis tests (Joh16) with significance level $\alpha = 0.05$ and the following settings:

1. We test, whether SHAP produces more helpful explanation than LIME in more then 50% of cases.

   H_0^1: $\pi \leq 0.5$

   H_A^1: $\pi > 0.5$

2. We test, whether LIME produces more helpful explanation than SHAP in more then 50% of cases.

   H_0^2: $\pi \geq 0.5$

   H_A^2: $\pi < 0.5$

5.2 Results

5.2.1 Binary forced choice experiment for CsFever data

The experiment as described in section 5.1.1 was conducted on 51[5] samples randomly pulled from the dataset of verifiable correctly predicted explanations.

The compared methods are *SHAP with sampling approximation and maximum of 5000 evaluations* (further just SHAP) and *Text augment LIME explanations with p = 0.3, word tokenization, exponential kernel function with cosine*

[4]Percentage of all claim-context pairs (in the entire population) for which SHAP produces better result.

[5]In total, the annotators assessed 115 samples, as can be seen in appendix D.2. However, the sampled subset contained many samples, for whom the background text did not contain the information needed, e.i. the label of the data point was incorrect, it was marked as verifiable instead of NEI. This is a known problem in CsFEVER dataset as described in section 3.1.1.

distance and σ = 0.3, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True (further just LIME). They were chosen according to the plan described in the experiment design section (section 5.1.1) - e.i. is with a series of short tests described in section 4.2 whose results are available in the appendix C.1.

The collected and anonymized data can be found in appendix D.2.

LIME	SHAP	Total
18	33	51

Table 5.1: Binary forced-choice experiment to compare LIME and SHAP methods for CsFEVER dataset

As specified in section 5.1.1, we denote a discrete random variable x = number of times SHAP produced a more helpful explanation than LIME.

We are now going to perform two one-tailed exact binomial tests $x \backsim \mathcal{B}(51, \pi)$ with significance level $\alpha = 0.05$:

1. H_0^1: $\pi \leq 0.5$[6]

   H_A^1: $\pi > 0.5$ [7]

   p-value $= 0.0244 < 0.05$

   Since the p-value is less than α, we reject the null hypothesis and therefore conclude that SHAP produces a more helpful explanation more frequently than LIME.

2. Considering the significant result from the previous test, refusing that LIME produces better explanations as often, or more often than SHAP, the following test is redundant.

   H_0^2: $\pi \geq 0.5$

   H_A^2: $\pi < 0.5$ [8]

   p-value $= 0.98795 > 0.05$

   We do not reject the null hypothesis.

[6]SHAP produces a more helpful explanation than LIME in 50%, or less than 50% of cases. Therefore either LIME is a more suitable method or both methods are equal and their choice is arbitrary.

[7]SHAP produces better results more frequently then LIME.

[8]SHAP produces a more helpful explanation than LIME in less than 50% of cases. Therefore by extension LIME produces a more helpful explanation more frequently than SHAP.

5.2.2 Binary forced choice experiment for CTKFacts data

The experiment as described in section 5.1.1 was conducted on 75 samples randomly pulled from the dataset of verifiable correctly predicted explanations.

The compared methods are *SHAP with sampling approximation and maximum of 1000 evaluations*(further just SHAP) and *LIME with exponential kernel function with kernel width 0.25 and cosine distance* (further just LIME). They were chosen according to the plan described in the experiment design section (section 5.1.1) - e.i. is with a series of short tests described in section 4.2 whose results are available in the appendix C.2.

The collected and anonymized data can be found in appendix D.3.

LIME	SHAP	Total
13	62	75

Table 5.2: Binary forced-choice experiment to compare LIME and SHAP methods for CTKFacts dataset

As specified in section 5.1.1, we denote a discrete random variable $x =$ number of times SHAP produced a more helpful explanation than LIME.

We are now going to perform two one-tailed exact binomial tests $x \backsim \mathcal{B}(75, \pi)$ with significance level $\alpha = 0.05$:

1. H_0^1: $\pi \leq 0.5$[6]

   H_A^1: $\pi > 0.5$ [7]

   p-value $= 4.20\text{e-}9 < 0.05$

   Since the p-value is less than α, we reject the null hypothesis and therefore conclude that SHAP produces a more helpful explanation more frequently than LIME.

2. Considering the significant result from the previous test, refusing that LIME produces better explanations as often, or more often than SHAP, the following test is redundant.

   H_0^2: $\pi \geq 0.5$

   H_A^2: $\pi < 0.5$ [8]

   p-value $= 0.99999999915308 > 0.05$

   We do not reject the null hypothesis.

Chapter 6

Conclusion

In this work, we have explored interpretability methods and the possibilities of their application to the state-of-the-art natural language processing (NLP) models used within fact-checking pipeline.

We assessed which of the pipeline tasks is most beneficial and suitable for interpretation and concluded that it is the natural language inference (NLI) task, which is at the end of the pipeline and infers whether a claim is supported, refuted or unverifiable from the textual context.

Further, we focused on the application of interpretability methods on NLI models fine-tuned for claim verification on two Czech datasets - CsFEVER and CTKFacts.

More specifically, we compared two local model-agnostic interpretability methods - LIME and SHAP. For this task, we used and edited partition explainer from SHAP library[1] and text explainer from LIME library[2]. For LIME we also designed and implemented a text-augmented version, that mutates the textual input based on word similarities in a fastText model.

We assessed different characteristics of the methods and compared many different parameter settings to find the best parametrization for each method and dataset, which we then compared in binary forced-choice experiments using a human-grounded evaluation.

The results of the tests were statistically evaluated and show that for both datasets SHAP is the better-suited method because the explanations it produces were found more helpful for the majority of instances.

All methods that we tested in the experiments used word tokenization. However, from our, subjective, perspective sub-sentence tokenization for SHAP yields better results. We did not include this setting in the experiment to ensure blinding but we propose it for further testing.

[1]`https://shap.readthedocs.io/`, available under *MIT license*

[2]`https://lime-ml.readthedocs.io`, available under *BSD 2-Clause "Simplified" License*

Future work

- Extend testing to application-grounded test, which provides stronger evidence for human-computer interaction and allows testing of methods with very different explanations, e.g. with varying tokenization. This would require real domain professionals (e.g. students of journalism) and a real task. We propose the following design:
  - For a series of claims, the annotator is instructed to verify, whether the claim is supported or refuted based on a provided context.
  - For each instance, they are presented with a claim and an explanation (generated by one of the randomly chosen compared methods).
  - We measure the correctness of their choices (using standard metrics - accuracy, recall...) and the time required for the task.
  - We statistically evaluate which method produced explanations which helped annotators be more efficient.
- The text-augmented LIME rely on embedding model used. We propose training our own fastText model on our available training data.
- Interpret explanations of the document retrieval task.

Bibliography

[AB18] Amina Adadi and Mohammed Berrada, *Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)*, IEEE Access **6** (2018), 52138–52160.

[AMJ18] David Alvarez-Melis and Tommi S Jaakkola, *On the robustness of interpretability methods*.

[BB19] Tom Buchanan and Vladlena Benson, *Spreading disinformation on facebook: Do trust in message source, risk propensity, or personality affect the organic reach of "fake news"?*, Social media + society **5** (2019), no. 4, 205630511988865 (English).

[BGJM16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, *Enriching word vectors with subword information*.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper, *Natural language processing with python*, O'Reilly Media, Sebastopol, CA, July 2009 (en).

[Bur17] Joanna M. Burkhardt, *History of fake news*, Library technology reports **53** (2017), no. 8, 5–2 (English).

[CKG$^+$19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov, *Unsupervised cross-lingual representation learning at scale*.

[CPC19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso, *Machine learning interpretability: A survey on methods and metrics*, Electronics (Basel) **8** (2019), no. 8, 832 (en).

[CRL$^+$18] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov, *XNLI: Evaluating cross-lingual sentence representations*, Proceedings of the 2018 Conference on Empirical Methods in Natural Language

Processing (Brussels, Belgium), Association for Computational Linguistics, October-November 2018, pp. 2475–2485.

[DB79] David L Davies and Donald W Bouldin, *A cluster separation measure*, IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-1** (1979), no. 2, 224–227.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding.*

[DUR+22] Jan Drchal, Herbert Ullrich, Martin R'ypar, Hana Vincourov'a, and Václav Moravec, *Csfever and ctkfacts: Czech datasets for fact verification*, ArXiv **abs/2201.11115** (2022).

[DVK17] Finale Doshi-Velez and Been Kim, *Towards a rigorous science of interpretable machine learning.*

[GBG+18] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov, *Learning word vectors for 157 languages*, Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[GG21] Alex Gramegna and Paolo Giudici, *SHAP and LIME: An evaluation of discriminative power in credit risk*, Front Artif Intell **4** (2021), 752558 (en).

[Joh16] David L Johnson, *Statistical tools for the comprehensive practice of industrial hygiene and environmental health sciences*, Standards Information Network, December 2016 (en), Available from: ProQuest Ebook Central, `http://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=4773832` [10 May 2022].

[LCGH13] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker, *Accurate intelligible models with pairwise interactions*, Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM, August 2013.

[LEL18] Scott M Lundberg, Gabriel G Erion, and Su-In Lee, *Consistent individualized feature attribution for tree ensembles.*

[Lip16] Zachary C Lipton, *The mythos of model interpretability.*

[LL17] Scott Lundberg and Su-In Lee, *A unified approach to interpreting model predictions.*

[LŠ21] Jan Lehečka and Jan Švec, *Comparison of czech transformers on text classification tasks*, Statistical Language and Speech Processing, Lecture notes in computer science, Springer International Publishing, Cham, 2021, pp. 27–37.

[Mac09] Bill MacCartney, *Natural language inference*, Ph.D. thesis, STANFORD UNIVERSITY, 2009.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient estimation of word representations in vector space.*

[MDP$^+$11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, *Learning word vectors for sentiment analysis*, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Portland, Oregon, USA), Association for Computational Linguistics, June 2011, pp. 142–150.

[Mol22] Christoph Molnar, *Interpretable machine learning: A guide for making black box models explainable*, Independently published, March 2022 (en).

[MV20] Ričards Marcinkevičs and Julia E Vogt, *Interpretability and explainability: A machine learning zoo mini-tour.*

[MZR18] Sina Mohseni, Niloofar Zarei, and Eric D Ragan, *A multidisciplinary survey and framework for design and evaluation of explainable AI systems.*

[OW19] Cailin O'Connor and James Owen Weatherall, *The misinformation age : How false beliefs spread*, Yale University Press, 2019 (en), Available from: ProQuest Ebook Central, `http://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=5607600` [12 May 2022].

[RG19] Nils Reimers and Iryna Gurevych, *Sentence-BERT: Sentence embeddings using Siamese BERT-networks*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (Hong Kong, China), Association for Computational Linguistics, November 2019, pp. 3982–3992.

[Rou87] Peter J Rousseeuw, *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*, J. Comput. Appl. Math. **20** (1987), 53–65 (en).

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, *Why should I trust you?*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), ACM, August 2016.

[Rud19] Cynthia Rudin, *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*, Nat Mach Intell **1** (2019), no. 5, 206–215 (en).

[RZLL16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, *SQuAD: 100,000+ questions for machine comprehension of text*, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (Austin, Texas), Association for Computational Linguistics, November 2016, pp. 2383–2392.

[Sch95] Sarah Schafer, *Vonnegut and clancy on technology, managing technology article*, Dec 1995.

[Sha16] L. S. Shapley, *17. a value for n-person games*, pp. 307–318, Princeton University Press, 2016.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch, *Neural machine translation of rare words with subword units*, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Berlin, Germany), Association for Computational Linguistics, August 2016, pp. 1715–1725.

[ŠK14] Erik Štrumbelj and Igor Kononenko, *Explaining prediction models and individual predictions with feature contributions*, Knowl. Inf. Syst. **41** (2014), no. 3, 647–665 (en).

[SNSS21] Milan Straka, Jakub Náplava, Jana Straková, and David Samuel, *RobeCzech: Czech RoBERTa, a monolingual contextualized language representation model*, Text, Speech, and Dialogue, Lecture notes in computer science, Springer International Publishing, Cham, 2021, pp. 197–209.

[SSH14] Jana Straková, Milan Straka, and Jan Hajič, *Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition*, Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (Baltimore, Maryland), Association for Computational Linguistics, June 2014, pp. 13–18.

[TLM10] Athanasios Tsanas, Max A. Little, and Patrick E. McSharry, *A simple filter benchmark for feature selection*, 2010.

[TVC$^+$18] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal, *The fact extraction and VERification (FEVER) shared task*.

[TVCM18] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal, *FEVER: a large-scale dataset for fact extraction and VERification*.

[VR14] Andreas Vlachos and Sebastian Riedel, *Fact checking: Task definition and dataset construction*, Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science (Stroudsburg, PA, USA), Association for Computational Linguistics, 2014.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is all you need.*

[Wan18] Cen Wan, *Hierarchical feature selection for knowledge discovery: Application of data mining to the biology of ageing*, Springer International Publishing AG, Cham, 2018 (English).

[WDS+20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush, *Transformers: State-of-the-art natural language processing*, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (Online), Association for Computational Linguistics, October 2020, pp. 38–45.

[WJ15] Shuohang Wang and Jing Jiang, *Learning natural language inference with LSTM*, CoRR **abs/1512.08849** (2015).

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean, *Google's neural machine translation system: Bridging the gap between human and machine translation.*

[ZAZ21] Xia Zeng, Amani S Abumansour, and Arkaitz Zubiaga, *Automated fact-checking: A survey*, Lang. Linguist. Compass **15** (2021), no. 10 (en).

[ZC18] Jianlong Zhou and Fang Chen (eds.), *Human and machine learning*, 1 ed., Human-Computer Interaction Series, Springer International Publishing, Cham, Switzerland, June 2018 (en).

Appendix A

Abbreviations used

BERT Bidirectional Encoder Representations from Transformers

ČTK Czech Press Agency

DR Fact Extraction and Verification

FEVER Fact Extraction and Verification

Lasso Least Absolute Shrinkage and Selection Operator

LIME Local Interpretable Model-agnostic Explanations

NEI Not Enough Info (label)

NLI Natural Language Inference

NLP Natural Language Processing

RNN Recurrent Neural Networks

RTE Recognizing Textual Entailment

SHAP SHapley Additive exPlanations

TF-IDF Term Frequency - Inverse Document Frequency

Appendix B

Functional evaluation metric

When we started searching through literature for information about interpretability assessment measures, we hoped to find a metric, which we could apply to the gained explanation results and rigorously compare the results of individual methods and their parameters among each other without the need for human evaluation.

However, as we wrote in section 2.5, automatic evaluation of explanations is problematic, presumably due to the subjective nature of explainability. Therefore, throughout the work, we have used human-based evaluations, even though, we think that it could be biased, due to the discussed imperceptible line between the model's correctness and the interpretability fidelity.

To gain another perspective, we developed a functional evaluation metric, which we could compare with our human-based experiment results. It is a modification of an unsupervised approach from (GG21) and it measures the consistency of explanations produced by the interpretability method - how well the methods are able to define distinct groups of observations.

The authors compare LIME and SHAP explanations for tabular data. They use both methods to produce explanations for a constant set of input instances. They then cluster the explanations, which have a format of feature-weight pairs[1] into k groups and measure Silhouette coefficient (Rou87) and the Davies–Bouldin index (DB79) which both reflect the within-cluster cohesion and between-clusters separation. The intuition behind the method is that there should be a single or a few patterns within the explanations within the explanations created by the same method and parameter setting (some features are constantly more important than others).

In the paper, the method was used on tabular data, i.e. with a constant feature set. Therefore, to use the method with our textual input, we had to transform our text tokens[2] into a fixed feature space.

[1]For tabular data, we can imagine a table with a row for each explanation, a column for each feature and a weight assigned to the feature in the explanation as the value.

[2]Since we apply the metric only to methods using word tokenization, we use *token* and

Transformation into fixed feature space. We decided to cluster the tokens into n groups.[3] In order to do so, we needed to represent the words as numerical values - word embeddings[4]. For this purpose, we used the same fastText model as we did for Text augmented LIME (section 4.3.2). We clustered the normalized embeddings using clustering algorithms (Optics, DBSCAN, Spectral clustering) but the best results were reached with K-means. Most other algorithms usually contained one giant cluster with the majority of all tokens.

Then we mapped the explanations into the new feature space - each explanation had n features, whose values were aggregations of the original values assigned to features which were now grouped together. We used maximum as an aggregation function.[5]

Finally, we applied the clustering as described in the paper - using K-means and spectral clustering and Euclidean distance.

Results. While results, which can be found in table B.1 and table B.2 look overly well for LIME, from fig. B.1 we see that the high Silhouette coefficient and low Davies–Bouldin index is caused by the fact that for LIME nearly all explanations are grouped in the same cluster.

The original paper does not consider such eventuality. We assume one or more of the following reasons:

- The explanations produced by LIME are all very consistent assigning similar values to similar words. Our experience and used literature however do not support such findings.
- LIME explanations are so chaotic that no patterns among them could be found. We also find this strange because while the results of LIME were inconsistent the method did not perform so much worse than SHAP for whom the metric worked.
- The approximations, made to unite the feature space and transform the explanations, result in losing too much information and the whole metric is futile. This is possible. However, for SHAP we obtain reasonable, consistent results (compared to the paper). Therefore, we do not understand why the metric does not work properly for LIME only. When

word interchangeably in this section.

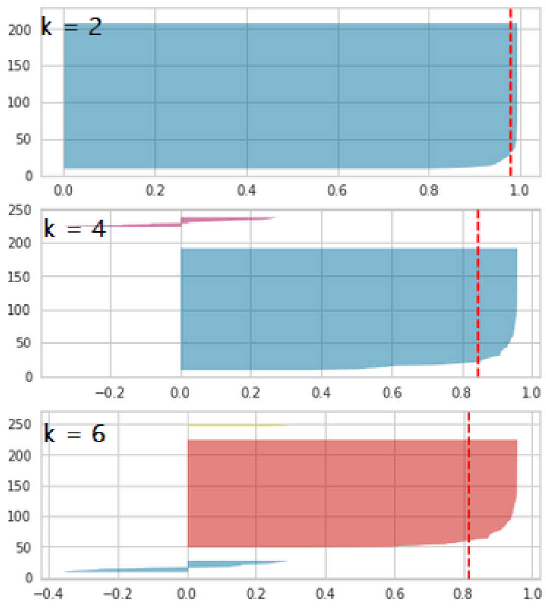
[3]We initially tested using a large feature space with all tokens with a LIME weight or SHAP value. This approach however did not yield good results because the number of features significantly exceeded the number of data points and the resulting dataset contained mainly missing values.

[4]Given the nature of embeddings, the idea was that semantically similar words would be clustered together, which matches them - on text meaning level - representing a similar feature.

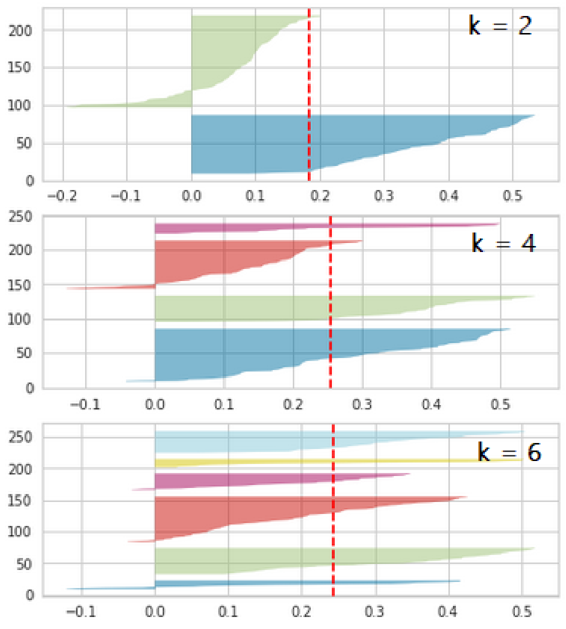
[5]We tested summing the values, or their absolute values, however maximum resulted in the most stable results.

often the words highly evaluated in the explanations were the same for both methods.

Since we do not understand what exactly the results represent and further research is needed, we do not draw any conclusions from this experiment.



(a) : LIME



(b) : SHAP

Figure B.1: CsFEVER : Size of clusters
Silhouette visualization of clusters with $k = 2$, $k = 4$ and $k = 6$. We see that while SHAP's explantations are reasonably distributed in all k clusters, LIME's explanations are nearly all grouped in one.

Method	Sil. K-means	DBI K-means	Sil. Spectral	DBI Spectral
TA LIME[1]	0.818	0.485	0.759	1.172
TA LIME[2]	0.68	0.86	0.679	0.943
LIME[3]	0.909	0.763	0.902	0.722
LIME[4]	0.862	1.003	0.86	1.018
LIME[5]	0.856	0.877	0.863	0.929
SHAP[6]	0.276	1.238	0.286	1.172
SHAP[7]	0.244	1.235	0.28	1.149

[1] *Text augment LIME explanations with $p = 0.4$, word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
[2] *Text augment LIME explanations with $p = 0.3$, word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
[3] *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
[4] *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.23$, features selected by Lasso, 5000 samples, tg = True*
[5] *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
[6] *SHAP with word tokenization, model mask, 5000 maximal evaluations*
[7] *SHAP with word tokenization, model mask, 1000 maximal evaluations*

Table B.1: CsFEVER: Functional evaluation of interpretability consistency

Method	Sil. K-means	DBI K-means	Sil. Spectral	DBI Spectral
LIME[1]	0.826	1.001	0.837	1.003
LIME[2]	0.802	1.039	0.803	1.144
TA LIME[3]	0.89	0.45	0.722	1.35
TA LIME[4]	0.968	0	0.847	3.012
SHAP[5]	0.338	1.163	0.386	1.267
SHAP[6]	0.297	1.373	0.341	1.457

[1] *LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.3, explaining 15 significant features selected by forward-pass, 5000 samples, tg = True*

[2] *LIME explanations with word tokenization, exponential kernel function with cosine distance and σ = 0.25, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

[3] *Text augment LIME explanations with p = 0.3, word tokenization, exponential kernel function with cosine distance and σ = 0.25, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

[4] *Text augment LIME explanations with p = 0.4, word tokenization, exponential kernel function with cosine distance and σ = 0.2, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

[5] *SHAP with word tokenization, model mask, 1000 maximal evaluations*

[6] *SHAP with word tokenization, model mask, 500 maximal evaluations*

Table B.2: CTKFacts: Functional evaluation of interpretability consistency

Appendix C

Choice of the best parametrization

C.1 CsFEVER

For CsFEVER we often had to exclude multiple examples because the data was corrupted and the context did not contain the information needed to verify the claim. In other words, the label was incorrect and it was supposed to be *NEI*. We did not sample new examples and only evaluated valid responses.

LIME

1. *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
2. *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.23$, features selected by Lasso, 5000 samples, tg = True*
3. *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
4. *Text augment LIME explanations with $p = 0.3$, word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
5. *Text augment LIME explanations with $p = 0.4$, word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

Setting 1	Setting 2	Total
8	3	12

Table C.1: CsFEVER: Compare LIME parametrizations

Setting 3	Setting 5	Total
4	9	12

Table C.2: CsFEVER: Compare LIME parametrizations

Setting 4	Setting 5	Total
8	4	12

Table C.3: CsFEVER: Compare LIME parametrizations

Setting 1	Setting 4	Total
2	13	15

Table C.4: CsFEVER: Compare LIME parametrizations

SHAP

1. *SHAP with word tokenization, model mask, 1000 maximal evaluations*
2. *SHAP with word tokenization, model mask, 5000 maximal evaluations*

SHAP max_evals 5000	SHAP max_evals 1000	Total
10	4	14

Table C.5: CsFEVER: Compare SHAP parametrizations

C.2 CTKFacts

LIME

1. *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.3$, explaining 15 significant features selected by forward-pass, 5000 samples, tg = True*
2. *LIME explanations with word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
3. *Text augment LIME explanations with $p = 0.3$, word tokenization, exponential kernel function with cosine distance and $\sigma = 0.25$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*
4. *Text augment LIME explanations with $p = 0.4$, word tokenization, exponential kernel function with cosine distance and $\sigma = 0.2$, explaining 15 significant features selected by highest-weights, 5000 samples, tg = True*

Setting 1	Setting 2	Total
6	9	15

Table C.6: CTKFacts: Compare LIME parametrizations

Setting 3	Setting 4	Total
4	11	15

Table C.7: CTKFacts: Compare LIME parametrizations

Setting 1	Setting 4	Total
8	7	15

Table C.8: CTKFacts: Compare LIME parametrizations
We selected *Setting 1* according to the methodology specified prior to the experiment, but with such close voting, we would appreciate more samples.

SHAP

- *SHAP with word tokenization, model mask, 500 maximal evaluations*
- *SHAP with word tokenization, model mask, 1000 maximal evaluations*

We excluded two samples because both methods produced the exactly same results. Overall, the explanations were often very similar and we decided based on nuances in the explanations. However, the results in table C.9 shows, that SHAP with max_evals 1000 produced more helpful explanations.

SHAP max_evals 500	SHAP max_evals 1000	Total
6	9	15

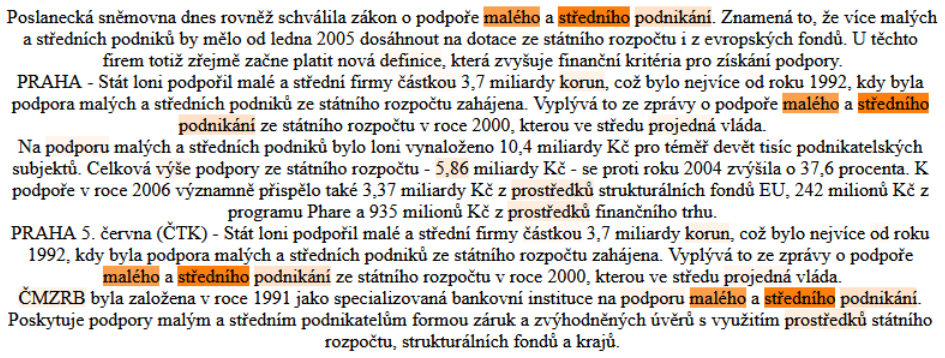
Table C.9: CTKFacts: Compare SHAP parametrizations

Appendix D

Experiments

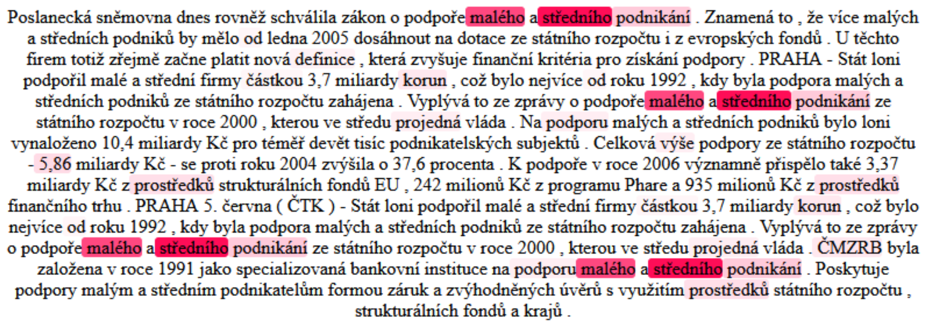
D.1 Transformation of LIME to SHAP

An example of an explanation in LIME, visualized with LIME visualization of highlighted text (fig. D.1), transformed into SHAP explanation visualized with SHAP visualization of highlighted text (fig. D.2).



Poslanecká sněmovna dnes rovněž schválila zákon o podpoře malého a středního podnikání. Znamená to, že více malých a středních podniků by mělo od ledna 2005 dosáhnout na dotace ze státního rozpočtu i z evropských fondů. U těchto firem totiž zřejmě začne platit nová definice, která zvyšuje finanční kritéria pro získání podpory.
PRAHA - Stát loni podpořil malé a střední firmy částkou 3,7 miliardy korun, což bylo nejvíce od roku 1992, kdy byla podpora malých a středních podniků ze státního rozpočtu zahájena. Vyplývá to ze zprávy o podpoře malého a středního podnikání ze státního rozpočtu v roce 2000, kterou ve středu projedná vláda.
Na podporu malých a středních podniků bylo loni vynaloženo 10,4 miliardy Kč pro téměř devět tisíc podnikatelských subjektů. Celková výše podpory ze státního rozpočtu - 5,86 miliardy Kč - se proti roku 2004 zvýšila o 37,6 procenta. K podpoře v roce 2006 významně přispělo také 3,37 miliardy Kč z prostředků strukturálních fondů EU, 242 milionů Kč z programu Phare a 935 milionů Kč z prostředků finančního trhu.
PRAHA 5. června (ČTK) - Stát loni podpořil malé a střední firmy částkou 3,7 miliardy korun, což bylo nejvíce od roku 1992, kdy byla podpora malých a středních podniků ze státního rozpočtu zahájena. Vyplývá to ze zprávy o podpoře malého a středního podnikání ze státního rozpočtu v roce 2000, kterou ve středu projedná vláda.
ČMZRB byla založena v roce 1991 jako specializovaná bankovní instituce na podporu malého a středního podnikání. Poskytuje podpory malým a středním podnikatelům formou záruk a zvýhodněných úvěrů s využitím prostředků státního rozpočtu, strukturálních fondů a krajů.

Figure D.1: LIME visualization



Poslanecká sněmovna dnes rovněž schválila zákon o podpoře malého a středního podnikání . Znamená to , že více malých a středních podniků by mělo od ledna 2005 dosáhnout na dotace ze státního rozpočtu i z evropských fondů . U těchto firem totiž zřejmě začne platit nová definice , která zvyšuje finanční kritéria pro získání podpory . PRAHA - Stát loni podpořil malé a střední firmy částkou 3,7 miliardy korun , což bylo nejvíce od roku 1992 , kdy byla podpora malých a středních podniků ze státního rozpočtu zahájena . Vyplývá to ze zprávy o podpoře malého a středního podnikání ze státního rozpočtu v roce 2000 , kterou ve středu projedná vláda . Na podporu malých a středních podniků bylo loni vynaloženo 10,4 miliardy Kč pro téměř devět tisíc podnikatelských subjektů . Celková výše podpory ze státního rozpočtu - 5,86 miliardy Kč - se proti roku 2004 zvýšila o 37,6 procenta . K podpoře v roce 2006 významně přispělo také 3,37 miliardy Kč z prostředků strukturálních fondů EU , 242 milionů Kč z programu Phare a 935 milionů Kč z prostředků finančního trhu . PRAHA 5. června (ČTK) - Stát loni podpořil malé a střední firmy částkou 3,7 miliardy korun , což bylo nejvíce od roku 1992 , kdy byla podpora malých a středních podniků ze státního rozpočtu zahájena . Vyplývá to ze zprávy o podpoře malého a středního podnikání ze státního rozpočtu v roce 2000 , kterou ve středu projedná vláda . ČMZRB byla založena v roce 1991 jako specializovaná bankovní instituce na podporu malého a středního podnikání . Poskytuje podpory malým a středním podnikatelům formou záruk a zvýhodněných úvěrů s využitím prostředků státního rozpočtu , strukturálních fondů a krajů .

Figure D.2: SHAP visualization

D.2 Binary forced choice experiment for CsFEVER data

claim	answer	user	claim	answer	user	claim	answer	user
4828	SHAP	user00	1134	SHAP	user01	4452	SHAP	user02
5013	SHAP	user01	4033	SHAP	user01	9762	SHAP	user02
19863	LIME	user01	16167	SHAP	user01	3749	SHAP	user02
19197	SHAP	user01	4187	LIME	user01	14077	LIME	user02
4871	LIME	user01	6182	SHAP	user01	1892	LIME	user02
11533	LIME	user01	19228	SHAP	user01	14935	LIME	user02
18517	LIME	user01	6120	LIME	user01	3473	SHAP	user02
15363	LIME	user01	18888	SHAP	user01	5261	SHAP	user02
1288	SHAP	user01	10920	SHAP	user01	8946	SHAP	user02
18402	LIME	user01	9133	SHAP	user01	17449	SHAP	user02
6217	LIME	user01	8894	SHAP	user01	13756	SHAP	user02
20882	LIME	user01	8996	LIME	user01	2310	LIME	user02
9293	SHAP	user01	18330	SHAP	user01	18907	SHAP	user02
6038	LIME	user01	6150	LIME	user01	3696	SHAP	user02
1602	SHAP	user01	12958	SHAP	user01	14055	SHAP	user02
17846	SHAP	user01	11366	SHAP	user02	11015	SHAP	user02
9158	SHAP	user01	8637	SHAP	user02	127	LIME	user02

Table D.1: Collected anonymized data for CsFEVER experiment

D.3 Binary forced choice experiment for CTKFacts

claim_id	answer	user_name
207	SHAP	user01
44	SHAP	user01
409	SHAP	user01
347	SHAP	user01
201	SHAP	user01
326	SHAP	user01
350	LIME	user01
294	SHAP	user01
60	SHAP	user01
266	SHAP	user01
23	SHAP	user01
379	SHAP	user01
331	SHAP	user01
339	SHAP	user01
204	SHAP	user01
151	SHAP	user01
437	SHAP	user01
68	LIME	user01
297	SHAP	user01
268	SHAP	user01
407	LIME	user01
48	SHAP	user01
387	SHAP	user01
433	SHAP	user01
406	SHAP	user01
451	SHAP	user01
82	LIME	user01
59	SHAP	user01
257	SHAP	user01
27	SHAP	user01
470	SHAP	user01
97	SHAP	user01
271	SHAP	user01
249	SHAP	user01
107	SHAP	user01
262	SHAP	user01
334	SHAP	user01
100	SHAP	user01
284	LIME	user01
182	SHAP	user01
338	SHAP	user01
408	SHAP	user01
135	SHAP	user01
9	SHAP	user01
340	SHAP	user01
126	SHAP	user01
244	SHAP	user01
360	SHAP	user01
56	SHAP	user01
385	SHAP	user01

Table D.2: Collected anonymized data for CTKFacts experiment - part I

claim_id	answer	user_name
369	SHAP	user01
265	SHAP	user01
282	SHAP	user01
21	LIME	user01
186	SHAP	user00
259	SHAP	user00
187	SHAP	user00
475	LIME	user00
217	LIME	user00
33	SHAP	user00
341	SHAP	user00
41	SHAP	user00
289	LIME	user00
211	SHAP	user00
325	SHAP	user00
80	SHAP	user00
458	LIME	user00
205	LIME	user00
128	LIME	user00
66	SHAP	user00
54	SHAP	user00
24	SHAP	user00
415	LIME	user00
288	SHAP	user00
469	SHAP	user00

Table D.3: Collected anonymized data for CTKFacts experiment - part II

Appendix E

List of Attachements

- Source code for the thesis.

  The enclosed source code does not contain models due to their size and the fact, that they were not trained as a part of this work.
  The explained models can be found on `https://huggingface.co/ctu-aic`
  and the used fastText model on `https://fasttext.cc/docs/en/crawl-vectors.html`