FELIX: Automatic and Interpretable Feature Engineering Using LLMs

Simon Malberg[0009−0000−3781−4700] (✉), Edoardo Mosca[0000−0003−4045−5328], and Georg Groh

School of Computation, Information and Technology, Technical University of Munich, 80333 Munich, Germany {simon.malberg, edoardo.mosca}@tum.de, grohg@in.tum.de

Abstract. Pre-processing and feature engineering are essential yet labor-intensive components of NLP. Engineers must often balance the demand for high model accuracy against interpretability, all while having to deal with unstructured data. We address this issue by introducing ***F**eature **E**ngineering with **LL**Ms for **I**nterpretability and E**x**plainability* (FELIX), a novel approach harnessing the vast world knowledge embedded in pre-trained Large Language Models (LLMs) to automatically generate a set of features describing the data. These features are human-interpretable, bring structure to text samples, and can be easily leveraged to train downstream classifiers. We test FELIX across five different text classification tasks, showing that it performs better than feature extraction baselines such as TF-IDF and LLM's embeddings as well as s.o.t.a. LLM's zero-shot performance and a fine-tuned text classifier. Further experiments also showcase FELIX's strengths in terms of sample efficiency and generalization capabilities, making it a low-effort and reliable method for automatic and interpretable feature extraction. We release our code and supplementary material: https://github.com/simonmalberg/felix.

Keywords: Large Language Models · Natural Language Processing · Feature Engineering · Text Classification

1 Introduction

Feature engineering (also known as *feature extraction* or *feature discovery*) is an essential part of the data science workflow. Feature engineering refers to the process of extracting characteristics from raw data to support a downstream statistical model, such as a classifier [13]. Many popular classifier models such as Logistic Regression [8] and Random Forest [5] require structured tabular data and cannot directly be trained on raw text data. As performance of these downstream models depends on the quality of the extracted features, data scientists typically dedicate a large share of their time to engineering and evaluating suitable features. According to a survey by Forbes, data scientists spend about 70% of their time preparing data and mining it for patterns [35]. These activities often require in-depth domain knowledge and complex judgment, complicating automation and thus making feature engineering costly.

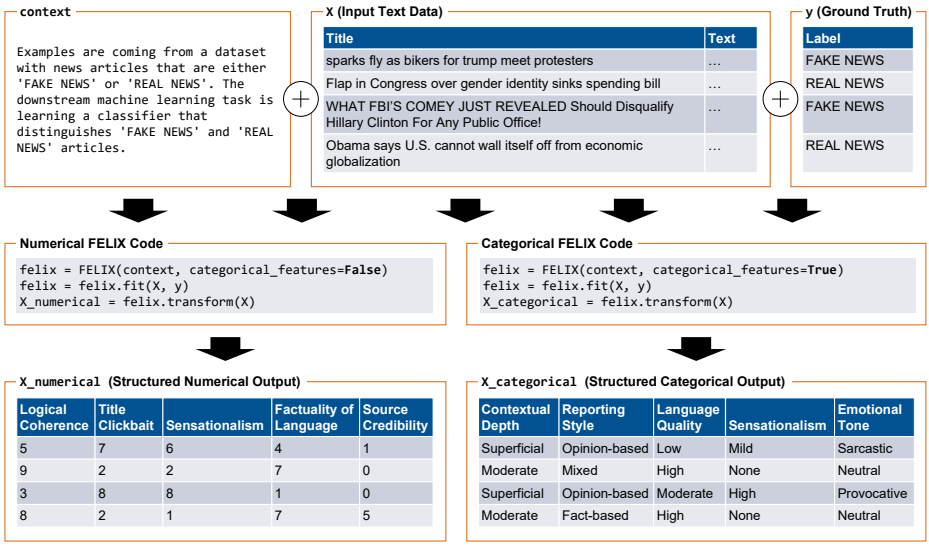


Fig. 1: In this example, FELIX learns features for detecting fake news articles: FELIX takes unstructured input data together with ground truth labels and a context/task description and converts it into structured numerical or categorical data. While doing this, FELIX learns both, appropriate features and their values. FELIX-learned features can serve as input for further data analysis.

The massive amount of knowledge embedded in pre-trained *Large Language Models* (LLMs) [27] potentially makes them a key ingredient in future *Automated Machine Learning* (AutoML) systems [29]. As identified by [38], three opportunities arise to use LLMs for AutoML: (1) simplifying human interaction with AutoML systems (especially setup and interpretation), (2) suggesting good initial configurations based on distilled knowledge, and (3) acting as components of an AutoML system. However, the required prompt engineering can once again become time-consuming [38, 7]. In line with these opportunities, we propose ***F**eature **E**ngineering with **LL**Ms for **I**nterpretability and E**x**plainability* (FELIX), a method to automatically transform unstructured text data into a structured tabular representation with human-interpretable features. Figure 1 shows an example of how to use FELIX to transform text data. Our contribution can be summarized as follows:

1. We introduce FELIX, a novel approach to automatically extract human-interpretable features from unstructured text data. FELIX leverages an LLM to generate candidate features from labeled example pairs, filter out redundant features, and accordingly assign feature values to each input instance.
2. We evaluate the relevance of FELIX-learned features across five downstream classification tasks, showing that FELIX outperforms traditional text feature

extraction methods such as TF-IDF and LLM embeddings as well as state-of-the-art LLM zero-shot capabilities and a fine-tuned RoBERTa model [21].
3. We thoroughly analyze FELIX with regards to sample efficiency and generalization capabilities to unseen data demonstrating that it is a highly sample-efficient feature engineering method and robust on out-of-domain data.

2 Related Work

We point to related works in three different areas: (1) automated feature engineering without LLMs, (2) automated feature engineering with LLMs, and (3) LLMs in data science and AutoML.

Automated feature engineering without LLMs. Even before generative LLMs became popular, some works have proposed systems for automatic feature engineering. The most noticeable of these systems arguably include *Cognito* [19] (later on further generalized in [18]), *Neural Feature Search* (NFS) [6], and *autofeat* [16]. *Cognito* explores feature construction choices hierarchically and non-exhaustively through a greedy exploration strategy with the goal of progressively maximizing the accuracy of the model. Users can specify a domain or data-specific preferences to prioritize the exploration. *NFS* is a neural architecture for automated feature engineering that trains a recurrent neural network with reinforcement learning to transform each raw feature with the goal of maximizing the performance of the machine learning algorithm. *Autofeat* is a Python library for the automatic generation and selection of a set of non-linear features to improve the prediction accuracy of a linear model while retaining its interpretability.

Automated feature engineering with LLMs. To the best of our knowledge, five methods for LLM-based feature engineering have been proposed by other researchers so far. *Context-Aware Automated Feature Engineering* (CAAFE) [15] uses an LLM to generate Python code for creating new features on tabular datasets. CAAFE further provides a textual explanation for each generated feature helping users with the interpretation. *SMARTFEAT* [20] crafts new features and writes the Python code required for creating these features through interacting with a *Foundation Model. Crafting High-Level Latents* (CHiLL) [25] takes expert-crafted queries with natural-language specifications of features and then extracts these features zero-shot from health records using LLMs. This way, CHiLL can handle high-level and more interpretable features that can be used by simple linear predictive models. *Large Language Model Feature Extractor* (LLMFE) [1] uses LLMs to extract characteristical differences between human-written and machine-generated scientific papers [28]. Since LLMFE does not directly generalize to other domains, is not fully automated, and is limited to numerical features only, it served as the main inspiration for us to evolve the idea into a general automated feature engineering approach. *Clue And Reasoning*

Prompting (CARP) [37] does not primarily do feature engineering but performs text classification by first prompting an LLM to find superficial clues (e.g., keywords, tones, semantic relations) and then use these clues to reason about the final classification decision. The identified clues help users to interpret the data and understand the classification rationale.

LLMs in data science and AutoML. Beyond using LLMs for feature extraction, several papers have explored opportunities to utilize LLMs for other steps of the data science workflow and for AutoML. Tornede et al. [38] explore the opportunities, challenges, and risks of using LLMs to improve AutoML and using AutoML to improve LLMs. Chopra et al. [7] identify challenges of data scientists conversing with LLM-powered chatbots. They find that data scientists spend a significant amount of time (64%) constructing prompts, including gathering and expressing the overarching domain context. Hassani et al. [12] present another overview of the opportunities and challenges associated with using *ChatGPT* in data science. They conclude that ChatGPT has the potential to greatly enhance the productivity and accuracy of data science workflows. Hassan et al. [11] developed a ChatGPT-based end-to-end system for conversational data science. *AutoML-GPT* Zhang et al. [41] automates several process steps of the machine learning pipeline using *GPT*. Ma et al. [22] introduce *InsightPilot*, an automated data exploration system that combines an LLM with an insight engine integrating different insight discovery tools. Other works apply LLMs for data wrangling [17, 30] and data pre-processing—e.g. error detection, data imputation, schema matching, and entity matching [40].

FELIX stands out from previous work with its end-to-end, generalized approach to automatic feature extraction from unstructured text data. Unlike existing methods, FELIX leverages LLM reasoning capabilities to directly analyze unstructured data, generating meaningful and interpretable classification features without intermediate code artifacts. Additionally, hierarchically grouping and selecting candidate features introduces a filtering step that ensures stronger independence between the features and the information they convey.

3 FELIX

We introduce FELIX as a new approach to learn interpretable data representations for textual data using LLMs. FELIX comes in two variants, one learns **numerical features** (with integer values ranging from 0 to 10) and the other one learns **categorical features** (taking exactly one value out of a set of 2-5 possible values). FELIX aims to discover highly-contextualized, sometimes complex or abstract features in the data (e.g., concepts such as logical cohesiveness) that are simple to grasp and interpret for human users. These features are found through a three-phase process (Figure 2 shows a visual overview):

1. **Feature generation:** Generate m_{cand} feature candidates from multiple pairs of training examples.

2. **Feature selection:** Select a subset of $m_{sel} \leq m_{cand}$ features from the generated feature candidates.
3. **Value assignment:** Convert each example into its feature representation vector of size m_{sel} by assigning a concrete value to each feature.

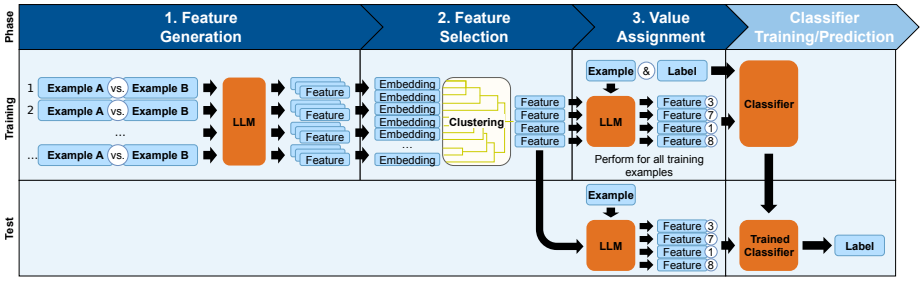


Fig. 2: This Figure shows FELIX's model architecture. First, features are generated from pairs of examples from different classes. Second, the generated features are consolidated into a final feature set by clustering them based on their semantic embeddings. Third, each example is converted into a feature representation by using an LLM to assign a value for each feature. Finally, classical machine learning models can be trained on the obtained data representation.

3.1 Feature Generation

To generate suitable feature candidates, an LLM is presented with a dataset-specific context/task description, similar to [14], as well as two examples from different classes and prompted to suggest a set of features that would best distinguish examples of the two classes. This step is repeated multiple times on different pairs of examples to learn a diverse set of features.

FELIX combines training examples into pairs through an oversampling strategy where it sequentially draws one example from each of two classes until both classes are exhausted. If one class is exhausted before the other, FELIX starts drawing the first example of the exhausted class again, so that each training example is seen by the LLM at least once. We found this pairing strategy to strike a reasonable balance between learning from a rich set of training examples while not requiring too many prompts to the LLM.

3.2 Feature Selection

The feature generation phase may have resulted in a large number m_{cand} of feature candidates (in our experience, typically 5-10 features per example pair). A large feature set would be computationally costly in the following steps and

might yield undesirable effects of high dimensionality [3]. FELIX builds on the fact that the feature set was generated through multiple independent prompts and likely contains several semantically redundant features. FELIX performs a three-step consolidation of the feature set to select $m_{sel} \leq m_{cand}$ unique features:

1. **Calculating embeddings:** A semantic embedding vector is calculated for each feature by serializing the feature schema (i.e., feature name, possible values or value range, and feature description) into a *JSON* format and converting the JSON string into a text embedding vector.
2. **Clustering features:** Feature embeddings are clustered using the *HDBSCAN* algorithm [26]. HDBSCAN uses an internal heuristic to determine the optimal number of clusters.
3. **Selecting representatives:** To filter duplicates and semantically similar features, only one feature is kept from each cluster. This is the feature with the shortest distance to the cluster centroid in the semantic embedding space.

This procedure substantially reduces data dimensionality and thus prompt sizes and run costs. We found that selecting features through clustering their embeddings provides more control, reliability, and traceability than prompting an LLM to consolidate the feature set. We provide a detailed validity analysis of using HDBSCAN clustering for feature selection in the supplementary material, including an in-depth exploration of other clustering configurations and alternative clustering algorithms.

3.3 Value Assignment

In the value assignment phase, FELIX transforms examples from a textual form into a structured feature representation by assigning concrete values to features. As assigning appropriate values to such highly-contextualized features typically requires reasoning, some form of general world understanding, and possibly domain knowledge, FELIX uses a pre-trained LLM to assign values to each feature. The LLM is provided with one example as well as the m_{sel} schemas of the selected features and prompted to return exactly one value per feature, i.e., an integer for numerical features or one of the allowed categorical values.

In rare cases, the LLM may return non-compliant values, i.e., values not in the set of allowed values. FELIX treats these occasions as missing values. For the purpose of this paper, we handle missing values as follows:

- **For numerical features**, we drop all feature columns that have more than 10% missing values. We fill the remaining missing values using *k-Nearest-Neighbors* (KNN) imputations [39] with scikit-learn's *KNNImputer* implementation based on $k = 5$ nearest neighbors and a distance weighting.
- **For categorical features**, we simply convert the data into a one-hot encoded data representation where missing values result in 0 values on all corresponding dummy columns.

In both cases, examples are converted into a purely numeric tabular format that can be handled by Logistic Regression and Random Forest while still ensuring traceability back to the LLM-generated feature schemas for explainability. As with the feature selection phase, we provide another validity analysis of using an LLM for assigning feature values in the supplementary material.

4 Experiments

To assess the quality of the features and data transformations generated by FELIX, we perform several experiments where FELIX transformations serve as input to a classifier. We measure end-to-end classification performance using F1 scores. As classifiers, we use the scikit-learn [33] implementations of Logistic Regression (**LR**) [8] and Random Forest (**RF**) [5] with default hyperparameters. Both classifiers are commonly used in research and industry and complement FELIX with suitable explanation insights through their learned coefficients and feature importance values, respectively. As LLMs, we use OpenAI's `gpt-3.5-turbo-0613` [31], falling back to `gpt-3.5-turbo-16k-0613` for longer-context prompts, and `gpt-4-1106-preview` [32]. We use OpenAI's default temperature of 0.7 during the feature generation phase to obtain sufficiently creative features and a temperature of 0.0 during the value assignment phase for consistent behavior. Feature embeddings are created using `chunk_size=1000` with OpenAI's `text-embedding-ada-002-v2`, which work well with both cosine and euclidean distance metrics[1]. We choose the latter for HDBSCAN clustering and selecting the feature closest to the cluster centroid from each cluster.

In the following, Section 4.1 describes the datasets, Section 4.2 describes the baselines, and Sections 4.3, 4.4, and 4.5 describe the experiments evaluating the relevance of FELIX features, sample efficiency, and generalization capabilities. Lastly, Section 4.6 discusses the interpretability of FELIX-generated features.

4.1 Datasets

We select five binary text classification datasets from different domains and sample a balanced subset from each to evaluate FELIX under various circumstances:

1. **Sentiment:** The *Tweet Sentiment Multilingual* dataset[2] [2] is a sentiment classification dataset with tweets in 8 different languages: Arabic, English, French, German, Hindi, Italian, Portuguese, and Spanish. We select only tweets with a *positive* or *negative* sentiment to construct a binary classification task. Our sample is balanced across the 8 languages.
2. **Hate speech:** The *Hate Speech Dataset from a White Supremacy Forum*[3] [10] contains posts sampled from a white supremacist forum, which are labeled as either *hate speech* (i.e., disparaging a target group of people based

[1] https://help.openai.com/en/articles/6824809-embeddings-frequently-asked-questions
[2] https://huggingface.co/datasets/cardiffnlp/tweet_sentiment_multilingual
[3] https://huggingface.co/datasets/hate_speech18

on some characteristic such as race or gender) or *no hate speech*. We only use the `text` column as classification input.

3. **Amazon/Yelp:** The *Amazon Review Polarity* dataset[4] [24] contains product reviews from Amazon. Each review is labeled as *negative* (1-2-star rating) or *positive* (4-5-star rating). The *Yelp Polarity* dataset[5] [42] contains *negative* (1-2-star rating) and *positive* (3-4-star rating) reviews from Yelp. We use it together with the Amazon dataset for our generalization experiment.
4. **Fake news:** The *Fake News TFG* dataset[6] contains news articles that are labeled as *false* (i.e., fake news) or *true* (i.e., real news).
5. **Papers:** The *IDMGSP* dataset[7] [1] contains the titles, abstracts, introductions, and conclusions of scientific papers, which are either *human-written* or *machine-generated* using *SCIgen*, *GPT-2*, *GPT-3*, *ChatGPT*, and *Galactica* models. Our sample includes an equal number of examples from each machine generator.

4.2 Baselines

To put results into perspective, we compare FELIX against four baselines:

1. **TF-IDF:** We transform text data into a *TF-IDF* representation [36]. During text preprocessing, we remove English stop words, convert all characters to lower case, remove non-alphabetic tokens, and apply a Porter stemmer [34] using the *NLTK* package [4].
2. **Text embeddings:** We transform text data into embeddings using OpenAI's `text-embedding-ada-002` model and `chunk_size=1000`.
3. **Zero-shot GPT-3.5/4 on raw text:** We classify the raw text data directly using `gpt-3.5-turbo-0613` or `gpt-4-1106-preview` in a zero-shot manner.
4. **RoBERTa on raw text:** We fine-tune a RoBERTa Base [21] model for the classification task. For a fair comparison to FELIX, we train RoBERTa for only one epoch (**R-1**). For reference, we also include a variant trained for 100 epochs (**R-100**). We use a batch size of 16 and a learning rate of $5e-5$ as suggested by the original BERT paper [9].

4.3 Experiment A: Feature Relevance

To assess if FELIX learns relevant features with predictive qualities, we evaluate end-to-end classification performance across all five datasets. While FELIX is not primarily meant to be a classifier but rather an instrument for interpretable feature exploration, classification performance serves as a proxy for the relevance and information content of the discovered features.

[4] https://huggingface.co/datasets/amazon_polarity
[5] https://huggingface.co/datasets/yelp_polarity
[6] https://huggingface.co/datasets/GonzaloA/fake_news
[7] https://huggingface.co/datasets/tum-nlp/IDMGSP

Setup. From each of the five datasets, we sample a training set of size $n_{train} = 50$ and a test set of size $n_{test} = 100$. We fit/fine-tune FELIX, TF-IDF, and RoBERTa to the training dataset. We then transform all examples in the training and test datasets into feature representations using the fitted FELIX, TF-IDF, and text embedding models, respectively. We fit both, LR and RF classifiers to the transformed training datasets and measure classification performance (F1 scores) on the corresponding transformed test datasets. For the zero-shot LLM classification baselines, we use the training set only to extract the set of unique classes and then classify the test dataset zero-shot into these classes. We repeat all FELIX, zero-shot LLM, and RoBERTa runs three times and report the average performance.

Data representation	**Classifier**	**Sentiment**	**Hate Speech**	**Amazon**	**Fake News**	**Papers**	**Avg.**
FELIX GPT-4 Cat.	LR	80.0%	86.3%	98.0%	95.3%	91.0%	90.1%
FELIX GPT-4 Cat.	RF	79.3%	86.3%	98.3%	95.0%	91.7%	90.1%
FELIX GPT-4 Num.	LR	79.9%	83.7%	98.0%	95.7%	**93.3%**	90.1%
FELIX GPT-4 Num.	RF	83.6%	**87.0%**	96.3%	96.3%	92.7%	**91.2%**
FELIX GPT-3.5 Cat.	LR	77.6%	86.2%	94.2%	91.7%	72.2%	84.4%
FELIX GPT-3.5 Cat.	RF	76.9%	85.3%	94.9%	91.3%	66.2%	82.9%
FELIX GPT-3.5 Num.	LR	76.0%	78.0%	97.7%	76.2%	71.3%	79.8%
FELIX GPT-3.5 Num.	RF	79.5%	76.5%	**99.0%**	77.4%	73.5%	81.2%
Raw Text	GPT-4	**93.0%**	84.9%	**99.0%**	86.1%	81.2%	88.8%
Raw Text	GPT-3.5	86.5%	82.6%	97.7%	93.7%	43.2%	80.7%
Raw Text	R-100	47.8%	70.5%	90.6%	**97.3**%	89.3%	79.1%
Raw Text	R-1	43.7%	46.3%	42.4%	55.4%	44.3%	46.4%
Text Embeddings	LR	57.1%	83.0%	97.0%	91.0%	59.9%	77.6%
Text Embeddings	RF	58.8%	80.0%	86.0%	90.0%	63.9%	75.7%
TF-IDF	LR	52.5%	71.0%	76.0%	80.8%	64.0%	68.8%
TF-IDF	RF	51.5%	38.6%	80.0%	92.0%	76.1%	67.6%

Table 1: Results of experiment A (feature relevance). All values are F1 scores.

Results. Table 1 shows the results of the feature relevance experiment. According to the cross-dataset averages, all FELIX-based classifiers achieve a better classification performance than TF-IDF- and embedding-based classifiers as well as the fine-tuned RoBERTa classifier. With the exception of FELIX GPT-3.5 Numerical, all FELIX-based classifiers also beat their zero-shot baseline on average, suggesting that FELIX can further improve classification performance independent of the underlying LLM. The fine-tuned RoBERTa model (R-1) is not able to learn accurate classifications from the limited training data. We saw it requiring up to 100 epochs of training (R-100) to catch up with some of the other models. For FELIX, we observe some variance across the datasets. Cate-

gorical variants of FELIX seem to achieve slightly more consistent performance results while numerical FELIX variants show higher performance variability.

Looking at individual datasets, the tweet sentiment dataset is the only dataset where all FELIX-based classifiers fall noticeably behind their zero-shot LLM counterparts but still clearly beat the TF-IDF, text embeddings, and fine-tuned RoBERTa baselines. We hypothesize that this is because sentiment classification is a relatively narrow-facetted NLP task where FELIX does not extract any other highly predictive features besides multiple variations of sentiment.

On the hate speech and fake news datasets, FELIX variants using GPT-3.5 or numerical features do not consistently outperform their zero-shot baselines. However, FELIX GPT-4 Categorical demonstrates a consistent performance advantage over zero-shot GPT-4 on both datasets.

The Amazon dataset seems to be the least challenging among the datasets, with most classifiers achieving 90+% F1 scores. The zero-shot LLM classifiers already achieve near-perfect performance (99.0% and 97.7%), leaving some FELIX variants to perform slightly better or identically and other FELIX variants to perform slightly worse, however all reaching F1 scores greater than 94%.

The scientific papers dataset is the most difficult classification task, according to the measured zero-shot LLM classification performance. Zero-shot GPT-3.5 even achieves worse classification performance than the TF-IDF and text embeddings baselines. For both, GPT-3.5 and GPT-4, FELIX adds a significant performance improvement over a zero-shot LLM classification. This indicates that classifiers greatly profit from the features extracted by FELIX on this dataset. FELIX GPT-4 variants achieve the best classification performance by far.

We conclude that FELIX learns relevant features with predictive value on all tested datasets. FELIX GPT-4 variants seem to show a consistently high ability to learn such features. When using FELIX with the less powerful GPT-3.5, performance results seem more consistent when learning categorical features, whereas results are more variable when learning numerical features.

4.4 Experiment B: Sample Efficiency

The results from experiment A suggest that FELIX can more efficiently extract features from limited training data than simpler methods such as TF-IDF or fine-tuned text classifiers such as RoBERTa. To further investigate this hypothesis, we run a sample efficiency analysis to evaluate classifier performance for different training set sizes.

Setup. The experimental setup is the same as for experiment A. The only difference is the size of the training set which we vary to be $n_{train} \in \{10, 20, 50, 100\}$. Classification performance is still evaluated against a test set of size $n_{test} = 100$ in all cases. To limit our API spending, we perform each run only once (except the $n_{train} = 50$ runs already performed threefold for experiment A) and average performance across all five datasets. We do not evaluate FELIX GPT-4 variants with $n_{train} = 100$ on all five datasets as they tend to become fairly expensive at this sample size.

Results. Figure 3 shows the classification performance for different training set sizes. As expected, classification performance generally improves the more training examples are available. FELIX outperforms TF-IDF and text embeddings for basically all tested sample sizes while the advantage is largest for lower sample sizes of 10 to 50 training examples. For 100 training examples, the performance of embedding-based classifiers is almost on a par with the performance of FELIX GPT-3.5 whereas TF-IDF is still significantly behind. FELIX seems to require at least 20-50 training examples to perform competitively with its zero-shot LLM counterpart, suggesting that the most efficient range for learning features with FELIX lies between 20 and 100 training examples. Below 20 examples, zero-shot LLM classification would provide better results, and above 100 examples, the other baselines catch up with FELIX GPT-3.5's performance. The fine-tuned RoBERTa classifier (R-1) is not able to efficiently leverage the available training examples unless trained for a large number of epochs (R-100). Even then, it requires more than 50 labelled training examples to reach FELIX GPT-3.5's level of performance while still falling far behind FELIX GPT-4 and zero-shot GPT-4.

Fig. 3: Results of experiment B (sample efficiency). Scores show the average across all five datasets. FELIX GPT-4 has not been evaluated on 100 training examples due to limited budget. Zero-shot LLM performance (Raw Text GPT-3.5/4) is constant with regards to the training sample size as no model is trained.

4.5 Experiment C: Generalization Capabilities

In this experiment, we assess FELIX's generalization capabilities and how well it adapts to out-of-domain data. In particular, we evaluate classification performance when models are trained on data from one domain and then tasked to predict classes of data from a slightly different domain. We hypothesize that

using a pre-trained LLM for feature generation and value assignment, FELIX should adapt to out-of-domain situations better than the more direct and mechanical feature engineering methods such as TF-IDF or text embeddings.

Setup. The experimental setup is identical to the setup used in experiment A except for the following adjustment: Train and test sets are sampled from two different datasets. We sample training sets of size $n_{train} = 50$ and test sets of size $n_{test} = 100$ from the Amazon and Yelp datasets and evaluate classifier performance with all four combinations of training and test sets. As with experiment A, we repeat all FELIX and zero-shot LLM runs three times and report the average performance.

Training Dataset =		**Amazon**	**Yelp**		**Yelp**	**Amazon**	
Test Dataset =		**Amazon**	**Amazon**		**Yelp**	**Yelp**	
Data representation	**Classifier**	**F1**	**F1**	**Δ F1**	**F1**	**F1**	**Δ F1**
FELIX GPT-4 Cat.	LR	98.0%	97.3%	-0.7%	96.0%	95.7%	-0.3%
FELIX GPT-4 Cat.	RF	98.3%	97.3%	-1.0%	**96.3%**	96.3%	0.0%
FELIX GPT-4 Num.	LR	98.0%	**99.3%**	1.4%	95.0%	96.0%	1.1%
FELIX GPT-4 Num.	RF	96.3%	**99.3%**	3.1%	95.7%	**97.0%**	1.4%
FELIX GPT-3.5 Cat.	LR	94.2%	96.3%	2.2%	92.0%	94.0%	2.2%
FELIX GPT-3.5 Cat.	RF	94.9%	96.3%	1.5%	92.3%	94.3%	2.2%
FELIX GPT-3.5 Num.	LR	97.7%	93.9%	-3.8%	93.7%	90.6%	-3.2%
FELIX GPT-3.5 Num.	RF	**99.0%**	95.0%	-4.1%	92.6%	93.0%	0.3%
Raw Text	R-100	90.6%	94.0%	3.8%	83.2%	78.1%	-6.1%
Raw Text	R-1	42.4%	49.8%	**17.3%**	50.7%	44.9%	-11.4%
Text Embeddings	LR	97.0%	82.5%	-15.0%	85.0%	88.9%	**4.6%**
Text Embeddings	RF	86.0%	80.3%	-6.6%	89.0%	84.8%	-4.7%
TF-IDF	LR	76.0%	61.0%	-19.7%	66.9%	59.9%	-10.5%
TF-IDF	RF	80.0%	66.8%	-16.4%	57.4%	45.0%	-21.7%

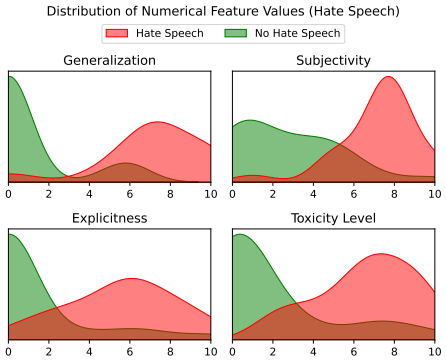
Table 2: Results of experiment C (generalization capabilities). Δ F1 is the percentage increase/decrease in F1 scores when models are evaluated on out-of-domain data (i.e., test dataset is different from training dataset). Zero-shot GPT variants are not included as they are not fine-tuned to the training dataset.

Results. Table 2 contains the results of the generalization experiments. We see that FELIX-based classifiers outperform embeddings- and TF-IDF-based classifiers as well as the fine-tuned RoBERTa classifiers in all evaluated in-domain (training and test sets from the same dataset) and out-of-domain (training and test sets from different datasets) settings. For text embeddings and TF-IDF, we see significant performance drops of up to 21.7% when switching from an in-domain to an out-of-domain setting. In contrast, more than half of the FELIX

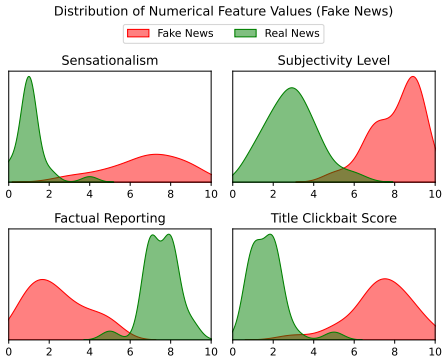
variants even improve their classification performance in out-of-domain settings and never lose more than 4.1% of their in-domain classification performance when evaluated on out-of-domain test sets. This suggests that FELIX features tend to be fairly generalizable and robust, even in out-of-domain settings.

4.6 Interpretability

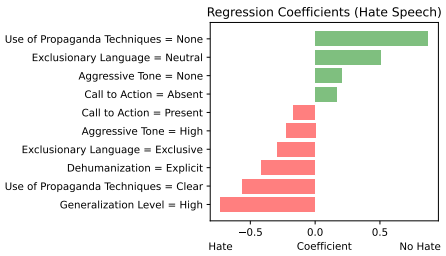
Combining FELIX-generated features with explainable classifiers such as Logistic Regression and Random Forest allows users to post-hoc evaluate feature coefficients and importance scores to judge the features' effectiveness. Figure 4 shows some exemplary analyses of features learned by FELIX.



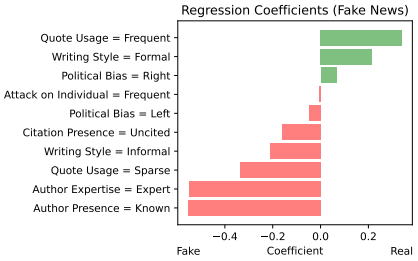
(a) Feature distributions for hate speech and non-hate speech



(b) Feature distributions for fake news and real news



(c) Feature coefficients for hate speech and non-hate speech



(d) Feature coefficients for fake news and real news

Fig. 4: Exemplary analyses of features learned by FELIX GPT-4 on the hate speech and fake news datasets. Figures 4a and 4b show the distribution of numerical feature values for the top 4 features according to Random Forest feature importance. Figures 4c and 4d show the Logistic Regression coefficients learned for some hand-selected dummies (i.e., feature-value pairs).

For instance, according to Figure 4c, hate speech seems to be characterized by strongly generalizing, exclusionary, and dehumanizing language with clear use of propaganda techniques. Using a call to action and an aggressive tone are additional, yet somewhat weaker indicators of hate speech. Figures 4b and 4d provide some explanations of the characteristics of fake news. Fake news tend to have an informal writing style with few citations and quotes, whereas real news tend to be more formal and quote more frequently. Fake news are significantly more sensationalist and use clickbait titles more often than real news. Interestingly, the coefficients suggest that fake news often have a known author that is presented as an expert. The political bias (left or right) seems to have only a weak correlation with the news being fake or real. Likewise, verbal attacks targeting a specific individual seem to be no strong indicator, neither for fake news nor for real news.

These analyses allow users to obtain tangible first insights about the data without investigating it manually. The features often capture abstract, highly-contextualized concepts (e.g., *sensationalism*) that are easy to grasp for humans but may be harder to extract from simpler text analysis methods such as TF-IDF. Ultimately, these insights may help to formulate initial hypotheses about the underlying domain quickly.

5 Conclusion

We have introduced ***F**eature **E**ngineering with **LLM**s for **I**nterpretability and E**x**plainability* (FELIX), a method for data exploration and insights generation using the vast general world knowledge embedded in pre-trained LLMs. Given a relatively small set of as few as 20 to 50 labeled training examples, FELIX is able to learn both, insightful features and their values from unstructured text data. Across five popular datasets from different domains, we demonstrated that features learned by FELIX are relevant and informative, in many cases with a predictive value greater than that of features based on TF-IDF or text embeddings. On four out of the five evaluated datasets, data representations generated by FELIX enabled even classical classifiers such as Logistic Regression and Random Forest to surpass the F1 scores achieved by state-of-the-art LLMs predicting the same examples in a zero-shot manner. We further demonstrated that FELIX is highly sample-efficient and more robust in out-of-domain settings than classical methods. Finally, we discussed how FELIX features can be analyzed and interpreted to obtain new insights about the underlying domain without investigating the data manually. We release our code and supplementary material[8].

Limitations and Future Work. Future work with a greater API budget may evaluate FELIX on more datasets, larger samples, and with other LLMs to uncover additional strengths and weaknesses of the methodology and hint at potential aspects of FELIX to be further improved. This could include combining

[8] https://github.com/simonmalberg/felix

FELIX-learned features with features from word-based methods such as TF-IDF to learn effective representations in any setting, even narrow-facetted tasks such as sentiment classification. Lastly, we expect that optimizing the feature selection procedure—e.g., retaining multiple features per cluster—would be the strongest lever for further improvements (see the supplementary material for a more detailed analysis).

Ethical Considerations. While FELIX may support users throughout the feature discovery process, it is important to note that FELIX relies on a pre-trained LLM. This may lead to biases, gaps, and other inaccuracies in model outputs that are not immediately obvious. Users should always sanity-check results and take specific care when using these results in high-stakes applications. FELIX should never be used for making automated decisions about humans.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Bibliography

[1] Abdalla, M.H.I., Malberg, S., Dementieva, D., Mosca, E., Groh, G.: A benchmark dataset to distinguish human-written and machine-generated scientific papers. Information **14**(10), 522 (2023)

[2] Barbieri, F., Espinosa Anke, L., Camacho-Collados, J.: XLM-T: Multilingual language models in Twitter for sentiment analysis and beyond. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference. pp. 258–266. European Language Resources Association, Marseille, France (Jun 2022), https://aclanthology.org/2022.lrec-1.27

[3] Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)

[4] Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." (2009)

[5] Breiman, L.: Random forests. Machine learning **45**, 5–32 (2001)

[6] Chen, X., Lin, Q., Luo, C., Li, X., Zhang, H., Xu, Y., Dang, Y., Sui, K., Zhang, X., Qiao, B., et al.: Neural feature search: A neural architecture for automated feature engineering. In: 2019 IEEE International Conference on Data Mining (ICDM). pp. 71–80. IEEE (2019)

[7] Chopra, B., Singha, A., Fariha, A., Gulwani, S., Parnin, C., Tiwari, A., Henley, A.Z.: Conversational challenges in ai-powered data science: Obstacles, needs, and design opportunities (2023)

[8] Cox, D.R.: The regression analysis of binary sequences. Journal of the Royal Statistical Society: Series B (Methodological) **20**(2), 215–232 (1958)

[9] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)

[10] de Gibert, O., Perez, N., García-Pablos, A., Cuadros, M.: Hate speech dataset from a white supremacy forum. In: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2). pp. 11–20. Association for Computational Linguistics, Brussels, Belgium (Oct 2018). https://doi.org/10.18653/v1/W18-5102, https://aclanthology.org/W18-5102

[11] Hassan, M.M., Knipper, A., Santu, S.K.K.: Chatgpt as your personal data scientist. arXiv preprint arXiv:2305.13657 (2023)

[12] Hassani, H., Silva, E.S.: The role of chatgpt in data science: how ai-assisted conversational interfaces are revolutionizing the field. Big data and cognitive computing **7**(2), 62 (2023)

[13] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer series in statistics, Springer (2009), https://books.google.de/books?id=eBSgoAEACAAJ

[14] Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., Sontag, D.: Tabllm: Few-shot classification of tabular data with large language models. In: International Conference on Artificial Intelligence and Statistics. pp. 5549–5581. PMLR (2023)

[15] Hollmann, N., Müller, S., Hutter, F.: Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. Advances in Neural Information Processing Systems **36** (2024)
[16] Horn, F., Pack, R., Rieger, M.: The autofeat python library for automated feature engineering and selection. In: Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I. pp. 111–120. Springer (2020)
[17] Jaimovitch-López, G., Ferri, C., Hernández-Orallo, J., Martínez-Plumed, F., Ramírez-Quintana, M.J.: Can language models automate data wrangling? Machine Learning **112**(6), 2053–2082 (2023)
[18] Khurana, U., Samulowitz, H., Turaga, D.: Feature engineering for predictive modeling using reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
[19] Khurana, U., Turaga, D., Samulowitz, H., Parthasrathy, S.: Cognito: Automated feature engineering for supervised learning. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). pp. 1304–1307. IEEE (2016)
[20] Lin, Y., Ding, B., Jagadish, H., Zhou, J.: Smartfeat: Efficient feature construction through feature-level foundation model interactions. arXiv preprint arXiv:2309.07856 (2023)
[21] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach (2019)
[22] Ma, P., Ding, R., Wang, S., Han, S., Zhang, D.: Insightpilot: An llm-empowered automated data exploration system. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 346–352 (2023)
[23] MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
[24] McAuley, J., Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. In: Proceedings of the 7th ACM conference on Recommender systems. pp. 165–172 (2013)
[25] McInerney, D.J., Young, G., van de Meent, J.W., Wallace, B.C.: Chill: Zero-shot custom interpretable feature extraction from clinical notes with large language models. arXiv preprint arXiv:2302.12343 (2023)
[26] McInnes, L., Healy, J., Astels, S.: hdbscan: Hierarchical density based clustering. Journal of Open Source Software **2**(11), 205 (2017). https://doi.org/10.21105/joss.00205, https://doi.org/10.21105/joss.00205
[27] Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heintz, I., Roth, D.: Recent advances in natural language processing via large pre-trained language models: A survey. ACM Comput. Surv. **56**(2) (sep 2023). https://doi.org/10.1145/3605943, https://doi.org/10.1145/3605943

[28] Mosca, E., Abdalla, M.H.I., Basso, P., Musumeci, M., Groh, G.: Distinguishing fact from fiction: A benchmark dataset for identifying machine-generated scientific papers in the LLM era. In: Ovalle, A., Chang, K.W., Mehrabi, N., Pruksachatkun, Y., Galystan, A., Dhamala, J., Verma, A., Cao, T., Kumar, A., Gupta, R. (eds.) Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023). pp. 190–207. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.trustnlp-1.17, https://aclanthology.org/2023.trustnlp-1.17
[29] Mumuni, A., Mumuni, F.: Automated data processing and feature engineering for deep learning and big data applications: a survey. Journal of Information and Intelligence (2024)
[30] Narayan, A., Chami, I., Orr, L., Arora, S., Ré, C.: Can foundation models wrangle your data? (2022)
[31] OpenAI: Chatgpt. https://openai.com/blog/chat-ai/ (2022), accessed on February 26, 2023
[32] OpenAI: Gpt-4 technical report (2023)
[33] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
[34] Porter, M.F.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
[35] Press, G.: Cleaning big data: Most time-consuming, least enjoyable data science task, survey says (Mar 2016), https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/
[36] Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management **24**(5), 513–523 (1988)
[37] Sun, X., Li, X., Li, J., Wu, F., Guo, S., Zhang, T., Wang, G.: Text classification via large language models (2023)
[38] Tornede, A., Deng, D., Eimer, T., Giovanelli, J., Mohan, A., Ruhkopf, T., Segel, S., Theodorakopoulos, D., Tornede, T., Wachsmuth, H., Lindauer, M.: Automl in the age of large language models: Current challenges, future opportunities and risks (2023)
[39] Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for DNA microarrays . Bioinformatics **17**(6), 520–525 (06 2001). https://doi.org/10.1093/bioinformatics/17.6.520, https://doi.org/10.1093/bioinformatics/17.6.520
[40] Zhang, H., Dong, Y., Xiao, C., Oyamada, M.: Large language models as data preprocessors. arXiv preprint arXiv:2308.16361 (2023)
[41] Zhang, S., Gong, C., Wu, L., Liu, X., Zhou, M.: Automl-gpt: Automatic machine learning with gpt. arXiv preprint arXiv:2305.02499 (2023)
[42] Zhang, X., Zhao, J., LeCun, Y.: Character-level Convolutional Networks for Text Classification. arXiv:1509.01626 [cs] (Sep 2015)