TECOFES: Text Column Featurization using Semantic Analysis

Ananya Singha, Mukul Singh, Ashish Tiwari, Sumit Gulwani, Vu Le, Chris Parnin
Microsoft Corporation, Redmond, WA
{ ananyasingha, singhmukul, astiwar, sumitg, levu, chrisparnin }@microsoft.com

Abstract

Extracting insights from text columns can be challenging and time-intensive. Existing methods for topic modeling and feature extraction are based on syntactic features and often overlook the semantics. We introduce the semantic text column featurization problem, and present a scalable approach for automatically solving it. We extract a small sample smartly, use a large language model (LLM) to label only the sample, and then lift the labeling to the whole column using text embeddings. We evaluate our approach by turning existing text classification benchmarks into semantic categorization benchmarks. Our approach performs better than baselines and naive use of LLMs.

1 Introduction

Text columns appear commonly in tables; for example, a column of product reviews or user feedback appears as a text column. A common task is to generate insights from such textual data. For doing that, typically, the free-form text is mapped into some kind of structured text, or features, to facilitate downstream analysis. One particularly useful class of features are obtained by performing categorization or classification of the textual data and assigning a label or category to each data point. Here, the categories or labels are taken from some finite set. For example, product reviews can be assigned labels indicating their sentiment, or they could be assigned labels indicating the product category (toys, furniture, etc.) they mention.

Assigning meaningful semantic labels to a large text column is tedious work. There is little automated support in popular tabular data processing tools for helping with this job. In this paper, we introduce the *semantic text column featurization* problem, or semantic categorization in short. We then present several variants of the problem that arise in pragmatic settings. We also present a scalable approach for extracting semantic features from text columns. Our approach samples a diverse set, uses an LLM to label the sample, and extends that labeling to the whole column using embeddings. The cost of using LLM is acceptable since the work would have to be done manually otherwise. Moreover, our approach uses the LLM sparingly compared to a naive application of LLM to every cell of the table. The semantic categorization problem is reminiscent of topic modeling (Blei et al., 2003b,a); however, here we are interested in semantic labels that need not be the topics mentioned directly in the text.

2 Semantic Categorization

We start with the problem definition. We define a general class of problems, parameterized by information $\mathcal{I}$ that is available at the time of performing semantic categorization. Thereafter, we will present specific choices for the parameter $\mathcal{I}$.

Definition 1 (Semantic Categorization Problem) *Given a column of textual data* $\mathtt{D}$*, which is an array of strings, and some additional information* $\mathcal{I}$ *in the form of a constraint* $\phi_{\mathcal{I}}$*, the* semantic categorization problem *seeks to generate a finite set of labels* $\mathtt{L}$*, and a mapping* $\mathtt{M} : \mathtt{D} \mapsto \mathtt{L}$ *such that (a)* $\mathtt{L}$ *and* $\mathtt{M}$ *are consistent with* $\mathcal{I}$*; that is,* $\phi_{\mathcal{I}}(\mathtt{L}, \mathtt{M})$ *evaluates to true, and*
(b) there exists some fixed semantic relation $\mathtt{R}$ *such that for every* i*, the relation* $\mathtt{R}$ *holds between the text* $\mathtt{D}[i]$ *and the label* $\mathtt{M}(\mathtt{D}[i])$ *assigned to it; that is,* $\mathtt{R}(\mathtt{D}[i], \mathtt{M}(\mathtt{D}[i]))$ *is true.*

A key feature of the semantic categorization problem is that the relation $\mathtt{R}$ is not known a-priori, and moreover, we may not have access to (training) examples for $\mathtt{R}$. The relation $\mathtt{R}$ becomes evident from the natural language semantics of the text in $\mathtt{D}$ and the labels $\mathtt{L}$ assigned to that text.

Example 1 *Consider the text column in Table 1. The semantic categorization problem seeks to generate the labels shown in Column "Label" for the text data given in Column "Text".*

Row	Text	Label
1.	The food was good.	Restaurant
2.	The dishes were delicious.	Restaurant
3.	Appetizers were awesome.	Restaurant
4.	The teddy bear was poor quality.	Toys
5.	The toy didn't work.	Toys
6.	Had to fix the broken doll.	Toys
7.	. . . [other 20 reviews about toys]	. . .

Table 1: Given the Column "Text", the goal is to generate the labels in Column "Label".

The above problem definition is parameterized by $\mathcal{I}$ and the constraint $\phi_{\mathcal{I}}$. We now instantiate these parameters by specific values to obtain different variants of the semantic categorization problem. These variants arise naturally when building an advanced "suggestions" tool for a data analyst working on text columns.

By-Label semantic categorization. In this variant, the extra information, $\mathcal{I}$, is a set $\mathtt{L}'$ of labels, and the constraint, $\phi_{\mathcal{I}}$, simply checks that the set of labels $\mathtt{L}$ returned in a solution is exactly equal to the set $\mathtt{L}'$ given in $\mathcal{I}$. Specifically,

$$\mathcal{I} := \mathtt{L}' \qquad \phi_{\mathcal{I}}(\mathtt{L}, \mathtt{M}) := (\mathtt{L}' == \mathtt{L})$$

By-Example semantic categorization. In this variant, the extra information, $\mathcal{I}$, contains a set of pairs in the relation $\mathtt{R}$ such that all desired labels appear in this set. The constraint, $\phi_{\mathcal{I}}$, checks that the labels assigned to these texts in the solution exactly match the labels assigned to them in $\mathcal{I}$, and that the solution uses no more labels than what are provided in $\mathcal{I}$. Specifically,

$$\begin{aligned} \mathcal{I} &:= \{(t, l) \mid t \in \mathtt{D}, l \in \mathtt{L}\} \subset \mathtt{R} \\ \phi_{\mathcal{I}}(\mathtt{L}, \mathtt{M}) &:= \bigwedge_{(t,l)\in\mathcal{I}} (\mathtt{M}(t) == l) \wedge \mathtt{L} = \mathcal{I}|_{\mathtt{L}}, \end{aligned}$$

where $\mathcal{I}|_{\mathtt{L}}$ denotes the set of all labels that appear in $\mathcal{I}$.

By-ColumnName semantic categorization. In this variant, the extra information, $\mathcal{I}$, contains the *column name* for the new column; that is, $\mathcal{I}$ is a string describing the attribute or feature of the textual data that is expected to be captured using the labels. The constraint, $\phi_{\mathcal{I}}$, checks that the labels $\mathtt{L}$ generated in the solution are reasonable values for the attribute given in $\mathcal{I}$.

Unsupervised semantic categorization. In this variant, the extra information, $\mathcal{I}$, is empty, and the constraint, $\phi_{\mathcal{I}}$, is $True$. Note that this variant does not put any additional restrictions on the labels or the mapping, apart from the correctness requirement (b) in Definition 1.

Algorithm 1: TECOFES: High-Level Approach

Input: Column data $\mathtt{D}$, information $\mathcal{I}$
Output: Mapping $\mathtt{M}$ from $\mathtt{D}$ to $\mathtt{L}$

1: Compute embeddings $\mathtt{D}^e$ for text strings in $\mathtt{D}$
2: $\mathtt{D}' \leftarrow \mathtt{SmartSample}(\mathtt{D}, \mathtt{D}^e)$
3: Use an LLM to construct $\mathtt{M}' : \mathtt{D}' \mapsto \mathtt{L}'$ using $\mathcal{I}$ and $\mathtt{D}'$
4: $\mathtt{M} \leftarrow \mathtt{ExtendMapping}(\mathtt{M}', \mathtt{D}, \mathtt{D}')$
5: **return** $\mathtt{M}$

3 The TECOFES Approach

The information $\mathcal{I}$ in the different variants eventually comes from the user. Since we want to minimize the work of the user, we focus on the *unsupervised* and the *By-columnName* semantic categorization problems.

Algorithm 1 presents our high-level approach TECOFES. Using the text column $\mathtt{D}$ and any optional information $\mathcal{I}$, we first calculate embeddings (Salton et al., 1975; Bengio et al., 2000; Neelakantan et al., 2022) for every text in $\mathtt{D}$, and then perform 3 steps: (1) use these embeddings to sample $\mathtt{D}'$ from $\mathtt{D}$ using a method $\mathtt{SmartSample}$, (2) use a large language model (LLM) to label $\mathtt{D}'$ using additional information $\mathcal{I}$, and (3) extend the labeling of $\mathtt{D}'$ to a labeling of the full column $\mathtt{D}$ using the method $\mathtt{ExtendMapping}$. Figure 1 illustrates our overall approach. We next provide details for Steps (1)-(3).

Example 2 *In the first step of our approach for solving the problem instance in Example 1, we sample from Table 1 to get a diverse set. Our smart sampling approaches are more likely to include one of Rows 1,2,3 and one of Rows 4,5,6,. . .,* even if the data is skewed heavily to the toys. *Say we picked Rows 1,4 as a sample set. Next, we ask the LLM to label these two rows, and say the LLM gives the labels shown in Table 1 for these 2 rows. In the last step, we extend the labeling of Rows 1 and 4 to label all remaining rows.*

3.1 Smart Sampling

The goal of the smart sampling step is to pick a small set $\mathtt{D}'$ that exhibits all the "diversity" present in $\mathtt{D}$. We want $\mathtt{D}'$ to be small because we use expensive LLM calls to label $\mathtt{D}'$. We propose three

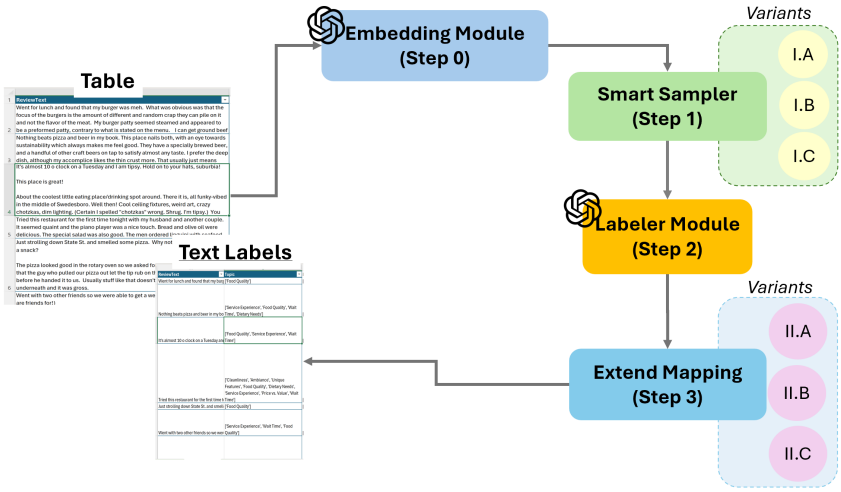

Figure 1: A workflow diagram showing different steps in the TECOFES approach.

implementations for `SmartSample`:

I.A We use a standard clustering algorithm, namely Agglomerative Clustering (Gordon, 1999), on the embeddings $\mathtt{D}^e$ to get $n/2$ clusters of $\mathtt{D}$, and then we randomly sample 2 elements from each cluster to get $\mathtt{D}'$.

I.B We get $\mathtt{D}'$ by sampling n elements uniformly over the distance of an embedding from the centroid of $\mathtt{D}^e$. Here the intuition is that we want points far away from the centroid (outliers) to be equally likely to be sampled as points close to the centroid.

I.C We compute $n/2$ principal components of $\mathtt{D}^e$ (using principal component analysis (Abdi and Williams, 2010)), transform $\mathtt{D}^e$ into the resulting $n/2$-dimensional space, and then pick points achieving extreme (largest and smallest) values in each component (to get n elements in $\mathtt{D}'$).

3.2 Using LLMs to label Samples

The prompt used to label the sampled data consists of three parts. The first part contains the following task description.

Your task is to label the given sentences. These sentences are sampled from a dataset. It is possible that there are sentences that share a label. Treat it like a multi-class classification task and come up with labels for the sentences. I will provide the sentences starting with a "-" and a line break in between them.

The second part of the prompt contains a description of the optional additional information $\mathcal{I}$ available. This is absent for the unsupervised case and takes the following form for other cases:

By-Example: *You are also provided with a few examples of labeled sentences, which you can take inspiration from, for further labeling other sentences.*

By-Label: *You are also provided with a set of possible labels that you can use to label each sentence.*

By-ColumnName: *You are also provided with a user intent/column name which briefly describes underlying intent of the classification and can help you pick suitable labels.*

The last part of the prompt includes the additional information promised above and the unlabeled sentences that the model needs to label.

3.3 Extending the Mapping

The `ExtendMapping` method solves the by-example semantic categorization problem; that is, the goal is to extend a mapping $\mathtt{M} : \mathtt{D}' \mapsto \mathtt{L}$ from $\mathtt{D}'$ to a mapping from $\mathtt{D}$ (to $\mathtt{L}$). We propose three implementations for `ExtendMapping`:

II.A We use an LLM to *expand* each label $l \in \mathtt{L}$ into text, $\mathtt{expanded}(l)$, containing words, phrases and sentences that are semantically similar to the label l (Carpineto and Romano, 2012). We then compute an embedding of the expanded label and assign each unmapped text in $\mathtt{D}$ to the label whose expanded form is closest to it in the embedding space; that is, $\mathtt{M}(t) = l$ where $l = \mathtt{argmax}_{l \in \mathtt{L}} \cos(t^e, \mathtt{expanded}(l)^e)$.

II.B For every unmapped text t in $\mathtt{D}$, we find the closest mapped text (closest in the embedding space) and assign t to the label of the closest mapped text; that is, $\mathtt{M}(t) = \mathtt{M}(t')$ where $t' =$

$\texttt{argmax}_{t' \in \texttt{D}'} \cos(t^e, t'^e)$.

II.C In this version, we do not expand the labels, but simply use the raw labels and their embeddings to find the closest label to an unmapped text(in the embedding space); that is, $\texttt{M}(t) = l$ where $l = \texttt{argmax}_{l \in \texttt{L}} \cos(t^e, l^e)$.

Note that the implementations **II.A** and **II.C** for `ExtendMapping` only assume access to the labels `L` and do not require a partial mapping from $\texttt{D}'$ to `L`. Thus, they can be viewed as approaches for solving the by-label semantic categorization problem. The procedure **II.B** solves the by-example semantic categorization problem.

The idea behind expanding the label in **II.A** is similar to that in query rewriting and expansion used in information retrieval (Carpineto and Romano, 2012). However, here we are expanding the labels, which are more like the *corpus* in the information retrieval analogy.

4 Evaluation

We apply TECOFES approach on publicly available classification datasets by turning them to different variants of the semantic categorization problem by hiding information.

Datasets. For detailed performance analysis, we picked multi-class and multi-label classification datasets from kaggle; details in Appendix A. We do a stratified sampling across each class which leaves us with 200 to 2000 datapoints based on the number of classes present. We target having about 50 representative samples from each class.

Experimental Setup. In all experiments, we used OpenAI `Ada-text-002` embedding model to compute embeddings for text (Neelakantan et al., 2022). The LLM steps were performed using GPT4o model (Ying et al., 2024) with temperature 0 and n=1 to get deterministic behavior. The only exception was when we generated synonyms in Approach II.A, where we used temperature 0.7 and n=50 to ensure diversity in the generated synonyms.

Baseline. To the best of our knowledge, there is no fully automated system for semantic feature extraction. The only natural baseline was to use an LLM to generate semantic feature labels by making one call for each row of the given dataset. We also use Latent Dirichlet Allocation (LDA) from topic modeling as the second baseline.

Metrics. We measure the performance of TECOFES using Partition Match and Semantic Match metrics. Given a textual Column `D` with n values, let P be the predicted labels $\{p_1, \ldots, p_n\}$ and let G be the ground truth labels $\{g_1, \ldots, g_n\}$.

Partition Match: Define an indicator function $\delta(x, y)$ which returns 1 if $x = y$ and 0 otherwise. The Partition Match metric can be rewritten as:

$$\text{Partition Match} = \frac{1}{\binom{n}{2}} \sum_{i<j} \delta(\delta(p_i, p_j), \delta(g_i, g_j))$$

where the summation is over all pairs (i, j) with $1 \leq i < j \leq n$. Partition Match is higher when the labelings P and G both agree on which pairs of elements they deem equal (and which unequal).

Semantic Match: Labels are unknown in the *Unsupervised* and *By-ColumnName* variants of semantic categorization problem. Consequently, the labels generated by TECOFES may not *syntactically* match the ground-truth labels. Hence, we define Semantic Match which matches labels semantically.

$$\text{Semantic Match} = \frac{1}{n} \sum_{i} \text{LLM\_Call}(\hat{p}_i, \hat{g}_i)$$

where $\text{LLM\_call}(\hat{p}_i, \hat{g}_i)$ is a function that checks if the predicted label is semantically similar to the ground truth label using a prompt. See Appendix B for the prompt and additional metrics based on *approximate syntactic* label matches.

5 Results

5.1 Comparing TECOFES variants

By picking one of the 3 variants for sampling and one of the 3 variants for extending the partial mapping, we get a total of 9 different TECOFES procedures. Figure 2 shows the performance (y-axis) on different metrics (colors) of the 9 different procedures (x-axis) aggregated across all benchmarks and across all variants of the *semantic categorization* problem. The version I.C-II.C, which uses PCA-based sampling (I.C) and label-similarity-based (II.C) labeling, performs the best on the metric that combines all metrics (rightmost bar in each set). A mildly surprising finding is that comparing the text embedding with embeddings of the labels (II.C) is better than comparing it with embedding of text (II.B). A possible reason is that matching with text is error prone because text often contains much more irrelevant information, which can cause spurious matches.

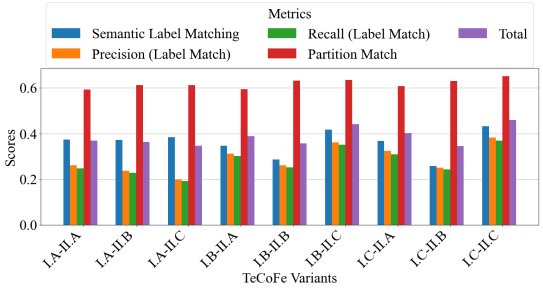

Figure 2: Performance of the 9 variants aggregated over benchmarks and problem variants. The rightmost bar in each group is the aggregation over all metrics.

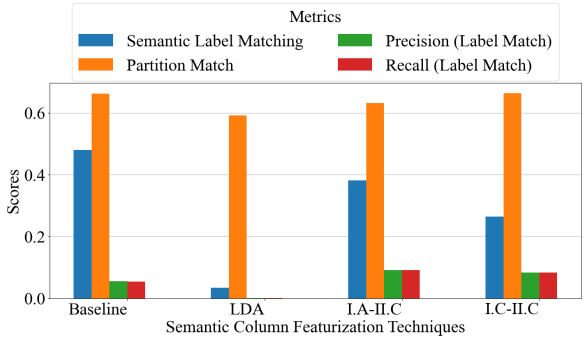

Figure 4: Performance of LLM *Baseline*, *LDA* compared against two *best performing* variants of TECOFES (I.C-II.C and I.A-II.C) using different metrics across all benchmark dataset on the *unsupervised* variant. Missing bars indicate a 0 score.

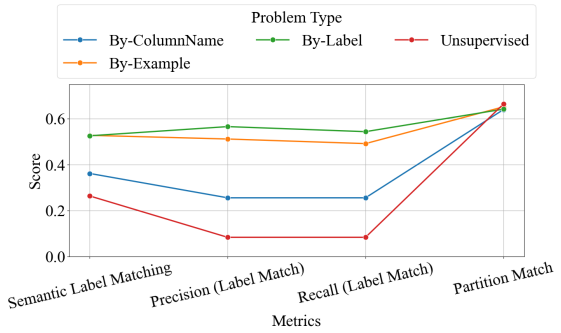

Figure 3: Performance of best TECOFES (I.C-II.C) variant across the different semantic categorization problem types.

5.2 Comparing problem variants

Let us fix the approach to I.C-II.C. Let us now consider the four problem variants of semantic categorization, and how I.C-II.C performs on these variants. Figure 3 presents the scores (y-axis) for different metrics (x-axis) achieved by I.C-II.C on different problem variants (colors). We observe that the *unsupervised* variant is the hardest, as expected, followed by *by-ColumnName*, and finally *by-example* and *by-label*. This is consistent across all metrics. The mildly surprising observation here was that *by-example* was not easier than *by-label* even though it has more information. This is possibly because the extra information in by-example is already available to an LLM.

5.3 Comparing TECOFES and Baselines

We compared our approach with two baselines for unsupervised learning methods. The first is an LLM baseline, where we use the LLM to label each row. The second baseline uses Latent Dirichlet Allocation (LDA) from topic modeling. Figure 4 shows the comparision between TECOFES variants I.A-II.C and I.C-II.C and the two baselines. Here we only compare on the *unsupervised* problem variant. Since LDA is a syntactic approach, it does not generate good semantic labels, but it is able to map text to (some representation of) its underlying "topic". Hence, the labeling generated by LDA has a reasonable partition match, but performs poorly on other metrics. The LLM baseline approach has a better Semantic Match score, but I.C-II.C does better on other metrics. Note that the LLM performs Semantic Match by saying "yes" or "no" when presented with a prediction text and the ground truth. This metric is higher for the LLM baseline, as the LLM generates verbose labels, making the evaluation LLM more likely to deem it a success.

The LLM baseline is not cost and time effective as each row of the corpus is passed through an LLM call which increases the latency along with cost. The cost for such a baseline approach will scale linearly as the number of data-points in the corpus increases. Table 2 points out the cost and the latency info.

Techniques	Cost($)	Time(sec)
Baseline	7.41	68.04
I.A-II.C	0.17	1.38
I.C-II.C	0.20	1.28

Table 2: Cost and Time effectiveness between the baseline and the TECOFES variants for *Unsupervised* semantic column featurization problem

6 Limitations

The main limitation of the work is that there is no guarantee that the assigned labels correctly match their corresponding text. LLMs are prone to hallucinations and wrong answers, and we use LLMs to label a small diverse sample (before it is extended to a labeling of the full dataset). A possible mitigation is to have the user check the labels generated by the LLM for the small sample. Another mitigation would be to design techniques to automatically detect labels that may be incorrect and in need of inspection by the user.

A second limitation is that the unsupervised semantic categorization problem suffers from severe underspecification. Several different kinds of semantic labels could be used to cluster a text column, and the unsupervised version offers no hint on which one is desired. Again, a mitigation here would be to engage with the user and ask the user to confirm the target labels before proceeding.

A third limitation is that our technique is based on using LLMs, which are costly to run and not deterministic. Our approach is designed to minimize the use of LLMs, but it still depends on it critically. One of our metrics, namely Semantic Match, used for evaluation is LLM-based and not completely reliable. We mitigated that risk by using other syntactic metrics in conjunction with the Semantic Match metric.

References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

David M. Blei, Michael I. Jordan, Thomas L. Griffiths, and Joshua B Tenenbaum. 2003a. Hierarchical topic models and the nested chinese restaurant process. In *Proc. Advances in Neural Information Processing Systems 16*. MIT Press.

David M. Blei, Andrew Y. Ng, and Michael I Jordan. 2003b. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1).

Allan David Gordon. 1999. *Classification*. CRC Press.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Zonghao Ying, Aishan Liu, Xianglong Liu, and Dacheng Tao. 2024. Unveiling the safety of gpt-4o: An empirical study using jailbreak attacks. *arXiv preprint arXiv:2406.06302*.

A Evaluation DataSets

For detailed performance analysis we picked 5 datasets from kaggle, out of which two are multi-label and the other three are multi-class. Following are the datasets and their brief description:

- **IMDB Genre dataset**: It is a multi-class classification dataset comprising of a textual column with movie description and a label column. There are four classes in the dataset; *Action*, *Comedy*, *Romance*, and *Horror*.
- **Twitter dataset**: It is a multi-class classification dataset comprising of a textual column comprising of tweets and Comments from twitter and Reddit. There are three classes in the dataset; *Positive*, *Negative* and *Neural*.
- **Academic Classification Dataset**: It is a multi-label classification dataset comprising of a text heavy abstract column along with six different labels (Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, Quantitative Finance).
- **Wikipedia promotional Dataset**: It is a multi-label classification dataset with 6 labels classifying the article text picked up from wikipedia. The classification labels are advert, coi, favpov, pr, resume.
- **News Categories Dataset**: Comprises of news article spread across following classes; Entertainment,Food, Economy, Sports, International relations etc.

B Metrics

Apart from Semantic Match and Partition Match, we also computed metrics based on syntactic or approximate syntactic matches. Semantic Match requires LLM calls and can be noisy. We, therefore, complemented it with syntactic *Label Match* metrics. Since labels may not be exactly syntactically equal, we used approximate match. In particular, we used Label Match Precision as defined by

$$\text{Precision} = \frac{1}{n}\sum_{i} \text{sub\_match}(\hat{p}_i, \hat{g}_i)$$

where$\hat{p}_i, \hat{g}_i$ are normalized versions of labels (p_i, g_i) obtained by removing any articles and stemming. The sub_match$(\hat{p}_i, \hat{g}_i)$ is a function that checks if the normalized predicted label $\hat{p}_i$ is a substring of the normalized ground truth label $\hat{g}_i$. If prediction is a *set* of labels, then we can extend the above in the standard way to define **Label Match Precision** and **Label Match Recall**. These standard notions were used in our work.

Semantic Matching of Labels Following is the prompt template that we use to compute semantic match between the predicted label and the ground truth label.

```
#Role
Given two strings, check if they are semantically equivalent.
Two strings are considered semantically equivalent if they
convey the same meaning or share a meta-label even if they
are not exactly the same in terms of characters or words
used. You should return `True` if the strings are semantically
equivalent, and `False` otherwise.

# Examples:
("Artificial Intelligence", "AI") == True
('Health and Lifestyle', 'Health') == True
('Sustainability and Environmental News', 'International
relations') == False

# User Input
Directly answer True or false as in the examples and end the
response using # End of Answer
("{Prediction}", "{Ground_Truth}") ==
```

Figure 5: Prompt Template used for Semantic Matching of Labels Metric.