Representing and Clustering Errors in Offensive Language Detection

Jood Otey
Oakland University
joodotey@oakland.edu

Laura Biester
Middlebury College
lbiester@middlebury.edu

Steven R. Wilson
University of Michigan-Flint
steverw@umich.edu

Abstract

WARNING: paper contains offensive content.
Content moderation is essential in preventing the spread of harmful content on the Internet. However, there are instances where moderation fails and it is important to understand when and why that happens. Workflows that aim to uncover a system's weakness typically use clustering of the data points' embeddings to group errors together. In this paper, we evaluate the K-Means clustering of four text representations for the task of offensive language detection in English and Levantine Arabic. We find Sentence-BERT (SBERT) embeddings give the most human-interpretable clustering for English errors and the grouping is mainly based on the targeted group in the text. Meanwhile, SBERT embeddings of Large Language Model (LLM)-generated linguistic features give the most interpretable clustering for Arabic errors.[1]

1 Introduction

Content moderation systems are used to mitigate the spread of offensive content online. These systems are usually successful at flagging offensive language, but may also incorrectly remove non-offensive content, and this incorrectly flagged non-offensive content is disproportionately shared by people who identify with a marginalized group. Previous works have shown bias in hate speech detection systems when it comes to text written in African American English (Xia et al., 2020; Sap et al., 2019; Harris et al., 2022). Moreover, moderation systems struggle to classify implicit offensive language. Mendelsohn et al. (2023) tested dog whistle detection on the Perspective API[2] and found that it assigned lower ratings to examples that used dog whistles (subtle, potentially harmful messages intended to only be understood by certain groups) instead of slurs.

In order to work toward correcting these types of issues, offensive language detection models must be examined more closely to understand how and why they are making mistakes. Evaluation metrics like F1-score and accuracy provide a compact and high-level means of scoring models, but are not enough to fully understand a model's behavior. To uncover where a model underperforms, researchers have recently shifted to automating aspects of the error analysis process and providing a systematic approach to analyzing a model's performance. These approaches are presented as error analysis tools (Rajani et al., 2022; McMillan-Major et al., 2022; R Menon and Srivastava, 2024; Gauthier-melancon et al., 2022; Tenney et al., 2020; Grace et al., 2023; Yuan et al., 2022; Wu et al., 2019) or Slice Detection Models (SDMs) (Hua et al., 2023; d'Eon et al., 2022; Sohoni et al., 2020; Eyuboglu et al., 2022). Error analysis tools provide a user-interface that allows practitioners to closely examine their systems and SDMs partition the data to "slices", aiming to identify those partitions on which the model underperforms, without the need for explicitly labeled subgroups.

These tools and models typically involve grouping the data points according to some human-understandable concept (e.g., gender, race). Clustering textual data requires them to be converted to a vector representation, like contextual embeddings, which gained popularity with the rise of pre-trained language models. SDMs and error analysis tools frequently use contextual embeddings when developing their frameworks (Rajani et al., 2022; McMillan-Major et al., 2022; R Menon and Srivastava, 2024; Hua et al., 2023; d'Eon et al., 2022; Sohoni et al., 2020; Eyuboglu et al., 2022).

Embeddings of text from neural networks encode information that can go beyond the label and these interpretable features or subclasses are not

[1]We publicly release all the code, models, and data needed to reproduce our results https://github.com/wetey/cluster-errors

[2]https://perspectiveapi.com/

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop), pages 368–380

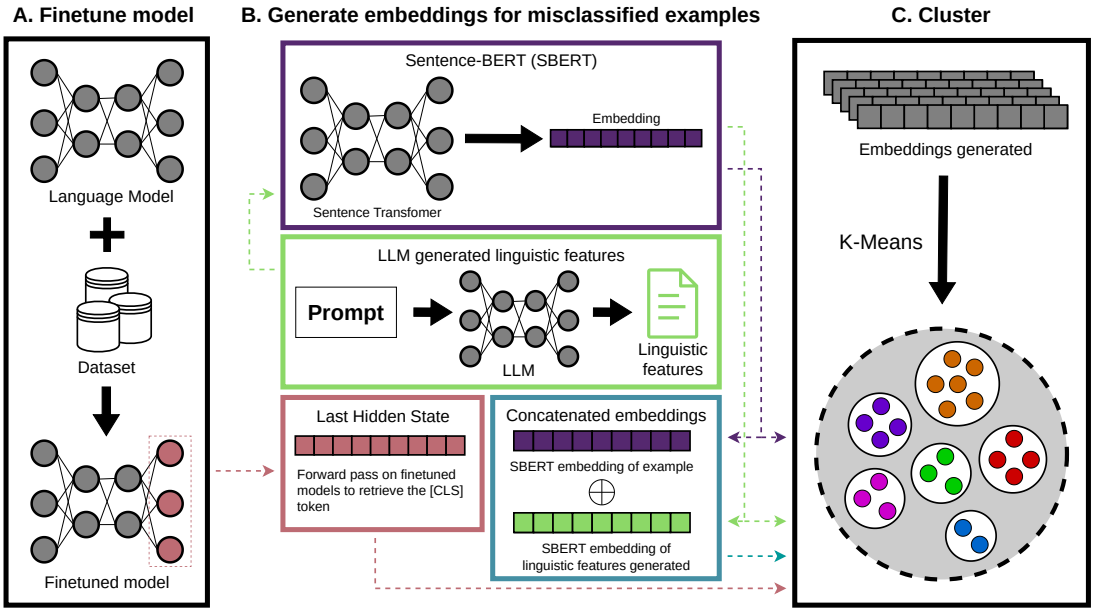


Figure 1: Overview of the methodology followed in the paper. **A.** Two Pretrained language models are finetuned on offensive language datasets (one on English and one on Arabic). **B.** We take the misclassified examples and generate embeddings to then cluster. We experiment with four types of embeddings: **(1)** **Last Hidden State (LHS)** are generated by extracting the `[CLS]` token from the last layer of the finetuned models. **(2)** **Sentence-BERT (SBERT)** are generated by running a sentence transformer model trained to generate semantically meaningful sentence embeddings. **(3)** **Linguistic features** are generated by prompting an LLM to generate linguistic features for the example, then the generated features are encoded using the same models as in embedding type **2**. **(4)** **Concatenated embeddings** are generated by concatenating the embeddings from **2** and **3**. **C.** The final step is running K-Means clustering on the generated embeddings.

always available with the dataset (Sohoni et al., 2022). In this work, we experiment with four types of embeddings of texts that were erroneously classified by offensive language detection models. Figure 1 summarizes the process used in this paper. We evaluate the embedding approaches to determine which leads to the most interpretable clustering and analyze what information about the underlying instances is represented by the embeddings. We find that for English, the two methods of clustering text using Sentence-BERT (SBERT) embeddings (Reimers and Gurevych, 2019) and concatenating those embeddings to embeddings of additional LLM-generated linguistic features yield the most human-interpretable clusters. Moreover, the clusters are primarily based on the group that was the target of the offensive language in the text. For Arabic, we find that clustering text using LLM-generated linguistic features yields the most human-interpretable clustering.

2 Background

Ad-hoc approaches to understand model performance for NLP classification tasks involve manually grouping the errors and giving each group/cluster a label. The process of having humans provide the label is laborious and subjective, leading to results that are often not reproducible (Wu et al., 2019).

Recent works that propose systematic error analysis frameworks for NLP classification tasks use clustering algorithms like K-Means and hierarchical clustering to group misclassified instances in an attempt to understand where the model underperforms (Rajani et al., 2022; McMillan-Major et al., 2022; R Menon and Srivastava, 2024). Similarly, popular Slice Detection Models (SDM)s are based on Gaussian Mixture Models (a generalized version of K-Means clustering) (Hua et al., 2023; d'Eon et al., 2022; Sohoni et al., 2020; Eyuboglu et al., 2022).

A popular vector representation used to cluster the textual data points is the last hidden layer of deep learning models, because it contains the learned representation of the entire sequence of tokens. When using pre-trained language models based on the transformer architecture like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) the final representation of their `[CLS]` token

is commonly used. These vector representations are used for understanding model performance in error analysis tools and SDMs for NLP classification tasks (Rajani et al., 2022; d'Eon et al., 2022; Hua et al., 2023). Other works such as McMillan-Major et al. (2022) use Sentence-BERT (SBERT) embeddings as the representation of the data points. SBERT embeddings (Reimers and Gurevych, 2019) use Siamese network structures (Bromley et al., 1993) to build a sequence-level text representation, which shows improvements over previous state-of-the-art sentence embedding methods on Semantic Textual Similarity tasks.

Prior works focused on quantitative evaluation of groups of embeddings with limited evaluation of how the choice of embedding approach might impact the final result (Rajani et al., 2022; McMillan-Major et al., 2022; R Menon and Srivastava, 2024). In this work we leverage two embedding types that have been commonly used to perform error analysis, last hidden state embeddings and SBERT embeddings, to build representations of the misclassified examples. Moreover, we propose a new method of representing errors which uses LLMs to generate linguistic features present in the errors. We evaluate the interpretability of the clusterings and provide insights into the type information the embeddings hold.

3 Data and Models

3.1 Datasets

The English dataset we use is the Measuring Hate Speech (MHS) dataset (Kennedy et al., 2020). The dataset originally contained 135,556 total annotations of 39,565 texts (~ 3.42 annotations per text), including statements about 7 target groups (gender, religion, sexuality, origin, race, age, and disability). The dataset is sourced from Twitter (40%), Reddit (40%), and YouTube comments (20%) and was annotated by 10,000 Amazon Mechanical Turk workers. We converted the continuous hatespeech scores to categorical labels using the ranges suggested by the authors:[3] examples with hate speech scores that are lower than -1 are considered `supportive`, between -1 and 0.5 are `neutral`, and scores greater than 0.5 are `hatespeech`. We remove duplicate examples along with those that received fewer than three total annotations, and we drop the neutral class. After these steps, we were left with 12,289 examples with 7497 examples labeled as `supportive` and 4792 labeled as `hatespeech`. We use 85% of the dataset for fine-tuning and 15% for testing.

The Arabic dataset we use is the Levantine Hate Speech and ABusive (L-HSAB) dataset (Mulki et al., 2019). The examples are in Levantine Arabic and the original dataset has 5,846 instances, which were all sourced from Twitter and annotated by three native Levantine Arabic speakers. After removing duplicates we were left with 5,754 examples. The dataset has three labels: `normal` (3576 examples), `abusive` (1713 examples), and `hate` (465). We use 85% of the dataset for fine-tuning and 15% for testing.

3.2 Classification Models

We finetune DistilBERT base uncased (Sanh et al., 2020) on the English dataset using an NVIDIA RTX A6000 GPU with a learning rate of $1e-05$ for 5 epochs. The model achieved an accuracy of 89.3%.

Since we are working with dialectal Arabic rather than Modern Standard Arabic (MSA), we finetuned MARBERT (Abdul-Mageed et al., 2021), a language model pre-trained on dialectal Arabic. We used the same hardware and hyperparameters as stated previously. The model achieved an accuracy of 87.9%.

We perform a forward pass on the models to obtain the predictions on the test set and the last hidden state embeddings from the classifiers. The finetuning and inference took less than an hour for both English and Arabic. To better understand where the models underperform, we focus on the misclassified examples (196 English examples and 106 Arabic examples).

4 Clustering Errors

4.1 Text Embeddings

We use KMeans++ from SKlearn[4] to cluster the errors. To determine the optimal number of clusters, we plot the inertia against the number of clusters and identify the elbow. We experiment with four types of vector representations for the errors.

The first representation is the **last hidden state** (LHS) from the classifiers we finetuned.

[3] The ranges are listed on the HuggingFace Dataset card: https://huggingface.co/datasets/ucberkeley-dlab/measuring-hate-speech

[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

The second representation uses **SBERT embeddings** (Reimers and Gurevych, 2019). We use the `all-distilroberta-v1` model for English and the `distiluse-base-multilingual-cased-v1` model for Arabic. These models effectively balance size, speed, and performance.

The third representation is built by prompting an LLM to extract **linguistic features** (using zero-shot prompting) and uses the same SBERT models mentioned above to convert the features to a vector representation. The linguistic features add more information that is not explicitly mentioned in the text, which we hypothesize will help bring errors with similar hidden features together. We use Mixtral 8x7b (Jiang et al., 2024) to extract features of the English errors ($temperature = 0.90$). To extract features of the Arabic errors, we use the Command-R[5] model ($temperature = 0.80$). We opt to use open-weight, freely accessible LLMs without automated guardrails that prevent generation of offensive content.

We use a 4-bit quantized version of Mixtral 8x7b because the full model is too large to run on the available hardware. It took approximately 1.5 hours to generate all the features. For Command-R we use Cohere's trial API.[6] Figure 2 displays an example of linguistic features generated by Mixtral 8x7b and Command-R as well as the prompts used (for the full linguistic features generated see Table 4).

The last representation we experiment with is **concatenating** the SBERT and linguistic feature embeddings. We use the same embeddings generated from the second and third representations. This approach includes a representation of the actual errors as well as the extra information the linguistic features provide.

4.2 Evaluation

Our method for evaluating the clustering is inspired by prior work on topic model evaluation (Chang et al., 2009). In that work, the five most probable words from a given topic t are presented to the annotator, in addition to an "intruder", which is a word with low probability for topic t, but high probability for a different topic. The words are shuffled, and the annotator is tasked with identifying the intruder. If the intruder is correctly identified, it implies that the topic is semantically coherent.

[5] `https://cohere.com/command`

[6] `https://docs.cohere.com/reference/about`

Representation	Number of Clusters	
	English	Arabic
LHS	8	7
SBERT	20	7
Features	16	7
Concatenated	21	9

Table 1: Number of clusters chosen for each representation-language pair using the elbow method.

In our work, annotators were shown questions that included four examples from a cluster and an "intruder" example that did not belong to the cluster and were asked to identify the intruder. To generate these questions, we randomly pick four examples from a cluster without replacement and then pick a random intruder from a different randomly selected cluster. Once an intruder is picked it is removed from the list of potential intruders to be used in future questions. We generate questions for all the text representations we described previously for both English and Arabic.

We recruited fluent speakers of English (2 Males, 9 Females, and 2 Non-binary; 8 ages 18-24 and 5 ages 25-34) and Levantine Arabic (3 Males and 4 Females; 2 ages 55-64, 2 ages 45-64, 1 age 34-44, and 2 ages 18-24) to annotate. We did not provide the annotators any additional details about the task beyond what was written in the questions. Every question was annotated three times in order to avoid ties. Before beginning the annotation process, the annotators were warned about potentially offensive content and informed that they were free to stop annotating at any point. Figure 3 and Figure 4 show an example of a question for evaluating the English and Arabic clustering respectively.

5 Results

5.1 Qualitative Analysis of Clusters

Table 1 shows the number of clusters we used for each vector representation for Arabic and English.

In the case of LHS clustering, we notice that the partitions are based on the (incorrectly) predicted label for both English and Arabic. Table 3 summarizes the percentage of data points with a predicted label that matches the primary label found in their cluster. We see that 96.59% of English and 91.43% of Arabic data points have a label that matches the majority predicted label in their cluster (per-cluster predicted label distributions can be found in Figure 5 and Figure 6). The reason for this outcome is

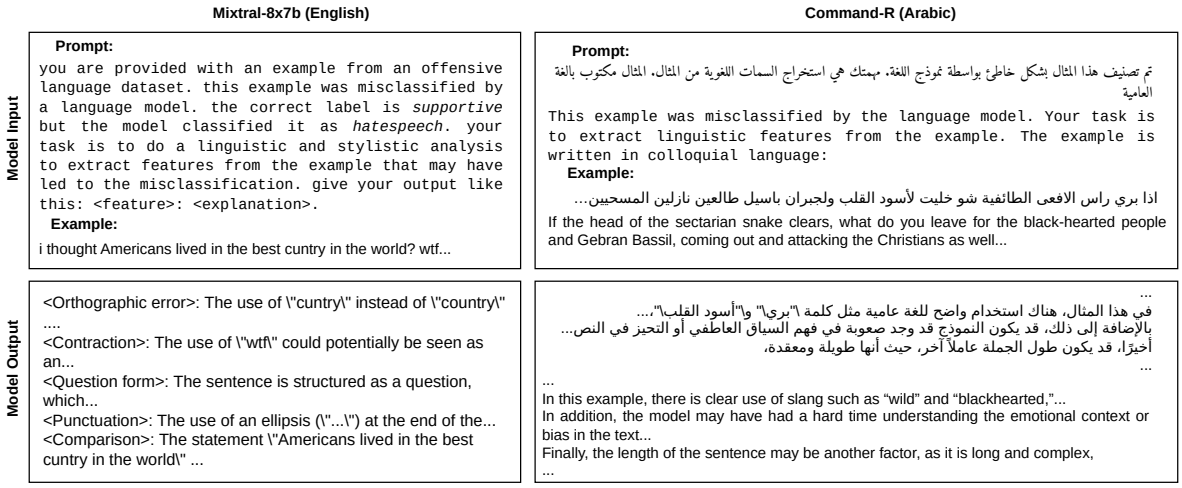


Figure 2: Example of LLM generated linguistic features.

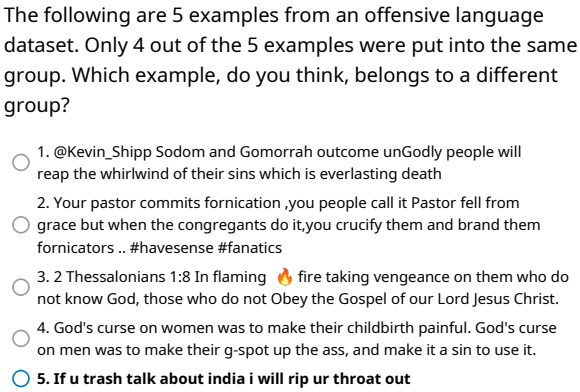


Figure 3: Example of an English clustering evaluation question, with the intruder highlighted in bold. Choices 1-4 are about religion and the last choice (the intruder) is targeting a specific origin.

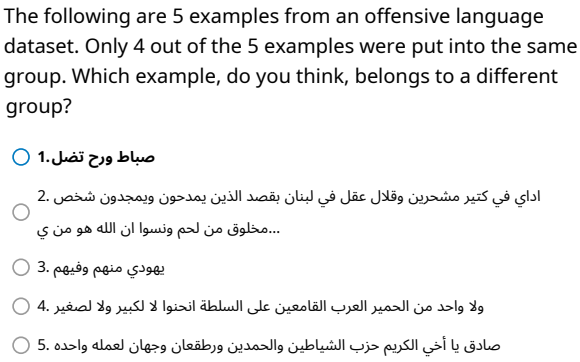


Figure 4: Example of an Arabic clustering evaluation question, with the intruder highlighted in bold. All the choices aside from the intruder mention a certain demographic. The first choice (the intruder) does not mention a specific demographic, the second is about Lebanese people, the third is referring to Jews, the forth is targeting Arabs, and the last choice mentions sects in Islam.

intuitive: since the LHS embeddings are extracted from the finetuned classifier, we expected a representation of the label to be embedded in the vector. Moreover, we notice SBERT clustering of the English errors is based on the target group, which we are able to verify because the target groups are annotated in the dataset (refer to Figure 7 for the target group distributions). We do not notice any pattern in the Arabic SBERT clustering. Moreover, for both languages, the clusterings of LLM generated linguistic features are not based on the target group or predicted label.

5.2 Human interpretability of clusters

We use accuracy to evaluate the clustering and define it as the percentage of questions where the majority of the annotators select the correct choice. Table 2 shows the results of the human evaluation on the clustering task. Humans achieved above baseline accuracy for all the text representations for English. The best performance was on SBERT and the concatenated embeddings, for which both approaches have human accuracy of 67.65%. We expected annotators to perform the best with SBERT embeddings because the clustering was primarily based on the targeted group in the text, which is often easier to identify based on keywords in the text. We find that using only the linguistic features did not improve the evaluators' performance.

Annotators were able to correctly identify the intruder only 27.78% of the time with LHS embeddings; this is particularly meaningful as LHS embeddings have been used in prior work on error analysis, such as the SEAL system (Rajani et al.,

Representation	Accuracy	
	English	Arabic
Baseline	20%	20%
LHS	27.8%	15.8%
SBERT	**67.6%**	12.5%
Features	34.3%	**31.6%**
Concatenated	**67.6%**	17.6%

Table 2: Evaluation results of clustering task.

Representation	% with majority label	
	English	Arabic
LHS	**96.59%**	**91.43%**
SBERT	65.76%	53.81%
Features	68.37%	50.51%
Concatenated	64.82%	51.61%

Table 3: Percentage of data points with a label that matches the majority label of their cluster.

2022). Lastly, the evaluators had an accuracy of 34.39% when choosing the intruder for the linguistic features clustering. We computed agreement using Cohen's Kappa (Cohen, 1960), and average scores ranged from 0.176-0.538 (detailed agreement results can be found in Table 5).

We use the same method to evaluate the Arabic clusters. Out of the four text representation approaches tested, only clustering the features yielded performance above the baseline (20%), with evaluators correctly identifying the intruder 31.58% of the time. A possible explanation for the improved performance is that the linguistic features are in MSA which is what the SBERT model is trained on.

The clusters of Arabic SBERT embeddings were the least human interpretable with accuracy of 12.5%, which indicates that SBERT embeddings using `distiluse-base-multilingual-cased-v1` may not yield meaningful embeddings for this task. There is a slight increase in performance with LHS embeddings, where evaluators had an accuracy of 15.79%. Lastly, the addition of the linguistic features slightly improved the clustering interpretability over only clustering SBERT embeddings. The accuracy of identifying the intruder with the concatenated embeddings was 17.65%. Average agreement Kappa score ranged from 0.130-0.314 (detailed agreement results can be found in Table 5).

6 Conclusion

Contextual embeddings are frequently used as a vector representation of textual data when performing error analysis. In this work, we evaluate four types of text representations of erroneously classified text in the context of offensive language in English and Arabic. We find that SBERT clustering provides the most human-interpretable clustering of English text, with each cluster focusing mainly on one target group. For Arabic we find that the SBERT embeddings of LLM generated features give the most interpretable clustering and the only approach to have above baseline performance. We notice the clustering of LHS in both English and Arabic is based on the predicted label. This paper builds on a growing area of research in error analysis for offensive language detection and provides insights into what information about the errors is encoded in their representation. Future work should explore other clustering algorithms and the effects of them on the interpretability and usefulness for error analysis, as well as automatic methods to generate informative labels about the clusters.

Limitations

We found that a major limitation when it came to working with Arabic was the lack of language models pre-trained on dialectal Arabic. The SBERT model we used as well as the LLM are only trained on Modern Standard Arabic (MSA). Dialectal Arabic is very different from MSA in the way words are spelled, the way that sentences are structured, and has a different lexicon. In addition, we experiment on one dataset and one model per language. The examples in the datasets are not representative of all the types of offensive language for English or Levantine Arabic. Moreover, human interpretability is only one way to measure a clustering's quality, future work should explore other ways to evaluate the choice of embedding for error analysis.

Ethical Considerations

This work aims to add to the ongoing research on error analysis for NLP and offensive language detection. We adhere to the intended usage guidelines of the models and datasets set by the developers of the models and datasets. In addition, annotators were warned about potentially being subject to offensive content and were informed they could stop annotating at any point. No information that

could potentially expose the identity of the annotator was collected and they could opt out all of the demographic questions if they wished.

7 Acknowledgments

We thank the reviewers for their feedback, and all the annotators, without whom this work would not have been possible. We also thank Dr. Alycen Wiacek and Dr. Lanyu Xu at Oakland University, who served as thesis committee members, for their feedback and questions that helped polish this work.

References

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7088–7105, Online. Association for Computational Linguistics.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Greg d'Eon, Jason d'Eon, James R. Wright, and Kevin Leyton-Brown. 2022. The spotlight: A general method for discovering systematic errors in deep learning models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 1962–1981, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. 2022. Domino: Discovering systematic errors with cross-modal embeddings. *Preprint*, arXiv:2203.14960.

Gabrielle Gauthier-melancon, Orlando Marquez Ayala, Lindsay Brin, Chris Tyler, Frederic Branchaud-charron, Joseph Marinier, Karine Grande, and Di Le. 2022. Azimuth: Systematic error analysis for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 298–310, Abu Dhabi, UAE. Association for Computational Linguistics.

Marie Grace, Jay Seabrum, Dananjay Srinivas, and Alexis Palmer. 2023. OLEA: Tool and infrastructure for offensive language error analysis in English. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 209–218, Dubrovnik, Croatia. Association for Computational Linguistics.

Camille Harris, Matan Halevy, Ayanna Howard, Amy Bruckman, and Diyi Yang. 2022. Exploring the role of grammar and word choice in bias toward african american english (aae) in hate speech classification. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 789–798, New York, NY, USA. Association for Computing Machinery.

Wenyue Hua, Lifeng Jin, Linfeng Song, Haitao Mi, Yongfeng Zhang, and Dong Yu. 2023. Discover, explain, improve: An automatic slice detection benchmark for natural language processing. *Transactions of the Association for Computational Linguistics*, 11:1537–1552.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arxiv:2401.04088 [cs].

Chris J. Kennedy, Geoff Bacon, Alexander Sahn, and Claudia von Vacano. 2020. Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application. *Preprint*, arxiv:2009.10277 [cs].

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *Preprint*, arxiv:1907.11692 [cs].

Angelina McMillan-Major, Amandalynne Paullada, and Yacine Jernite. 2022. An interactive exploratory tool

for the task of hate speech detection. In *Proceedings of the Second Workshop on Bridging Human–Computer Interaction and Natural Language Processing*, pages 11–20, Seattle, Washington. Association for Computational Linguistics.

Julia Mendelsohn, Ronan Le Bras, Yejin Choi, and Maarten Sap. 2023. From dogwhistles to bullhorns: Unveiling coded rhetoric with language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15162–15180, Toronto, Canada. Association for Computational Linguistics.

Hala Mulki, Hatem Haddad, Chedi Bechikh Ali, and Halima Alshabani. 2019. L-HSAB: A Levantine Twitter dataset for hate speech and abusive language. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 111–118, Florence, Italy. Association for Computational Linguistics.

Rakesh R Menon and Shashank Srivastava. 2024. DISCERN: Decoding systematic errors in natural language for text classifiers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19565–19583, Miami, Florida, USA. Association for Computational Linguistics.

Nazneen Rajani, Weixin Liang, Lingjiao Chen, Margaret Mitchell, and James Zou. 2022. SEAL: Interactive tool for systematic error analysis and labeling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 359–370, Abu Dhabi, UAE. Association for Computational Linguistics.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *Preprint*, arxiv:1910.01108 [cs].

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence, Italy. Association for Computational Linguistics.

Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. 2020. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In *Advances in Neural Information Processing Systems*, volume 33, pages 19339–19352. Curran Associates, Inc.

Nimit S. Sohoni, Jared A. Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. 2022. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Preprint*, arXiv:2011.12945.

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118, Online. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy. Association for Computational Linguistics.

Mengzhou Xia, Anjalie Field, and Yulia Tsvetkov. 2020. Demoting racial bias in hate speech detection. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 7–14, Online. Association for Computational Linguistics.

Jun Yuan, Jesse Vig, and Nazneen Rajani. 2022. isea: An interactive pipeline for semantic error analysis of nlp models. IUI '22, page 878–888, New York, NY, USA. Association for Computing Machinery.

A Linguistic features example

Table 4 shows an example of LLM generated features. The English features were generated by Mixtral-7x8b and the Arabic features were generated by Command-R.

B Predicted label distributions

Figure 5 and Figure 6 show the predicted label distribution for each cluster. The LHS clustering for Arabic (see Figure 6) and English (see Figure 6) is primarily based on the predicted label. We do not observe a similar distribution in the other clusterings, every cluster has examples from every label.

C Target group distributions

Figures 7a to 7d show the percentage of each target group per cluster. It is important to note that some examples may belong to multiple target groups, hence the total percentage for some of the clusters is more than 1. In addition, no misclassified examples belonged to the `disability` target group, hence it is not included in the figures.

We observe in the LHS clustering (see Figure 7a) every cluster has examples from almost every target group, whereas the distribution for SBERT clustering (see Figure 7b) shows there is a majority target group per cluster. Figure 7c and Figure 7d shows the target group distribution for linguistic features clustering and concatenated embeddings clustering respectively. Their distribution is more similar to the SBERT distribution.

D Annotator agreement on clustering task

Table 5 shows the average Cohen's Kappa score (annotator agreement) for the human evaluation of the clustering.

Example	LLM generated features
i thought Americans lived in the best cuntry in the world? wtf...	*<Orthographic error>: The use of "cuntry" instead of "country" may have confused the model, as it is a non-standard spelling that could be indicative of a variety of different tones or emotions, including hate speech.* *<Contraction>: The use of "wtf" could potentially be seen as an informal and colloquial expression, which might be associated with a more supportive tone rather than a hateful one.* *<Question form>: The sentence is structured as a question, which might be interpreted as seeking information or clarification rather than expressing hate speech.* *<Punctuation>: The use of an ellipsis ("...") at the end of the sentence could indicate a sense of uncertainty or confusion, which is more in line with a supportive tone.* *<Comparison>: The statement "Americans lived in the best cuntry in the world" could be seen as a positive comparison or boast, rather than a derogatory statement towards any particular group of people.* *Overall, the combination of these features might have led the model to misclassify the sentence as hate speech, when in fact it is more likely to be supportive in tone.*
اذا بري راس الافعى الطائفية شو خليت لأسود القلب ولجبران باسيل طالعين نازلين المسحيين If the head of the sectarian snake clears, what do you leave for the black-hearted people and Gebran Bassil, coming out and attacking the Christians as well...	يبدو أن النص يحتوي على لغة عامية وشتائم، وقد يكون محتواه عاطفياً للغاية، مع وجود إشارات إلى الطائفية والتحيز الديني. هذه السمات قد تجعل من الصعب على النموذج اللغوي تصنيف النص بشكل صحيح، حيث يمكن أن تكون العاطفة واللغة العامية تحديات صعبة للنماذج اللغوية لتفسيرها والتعامل معها. بالإضافة إلى ذلك، قد يكون المحتوى الطائفي أو التحيز الديني حساساً للغاية ويمكن أن يؤثر أيضاً على قدرة النموذج على فهم السياق والتصنيف بدقة *The text appears to contain slang and expletives, and its content may be highly emotional, with references to sectarianism and religious bias. These features may make it difficult for a linguistic model to classify text correctly, as emotion and slang can be difficult challenges for linguistic models to interpret and deal with. In addition, sectarian content or religious bias may be very sensitive and can also affect the model's ability to accurately understand context and classification.*

Table 4: Example of linguistic features generated using LLMs.

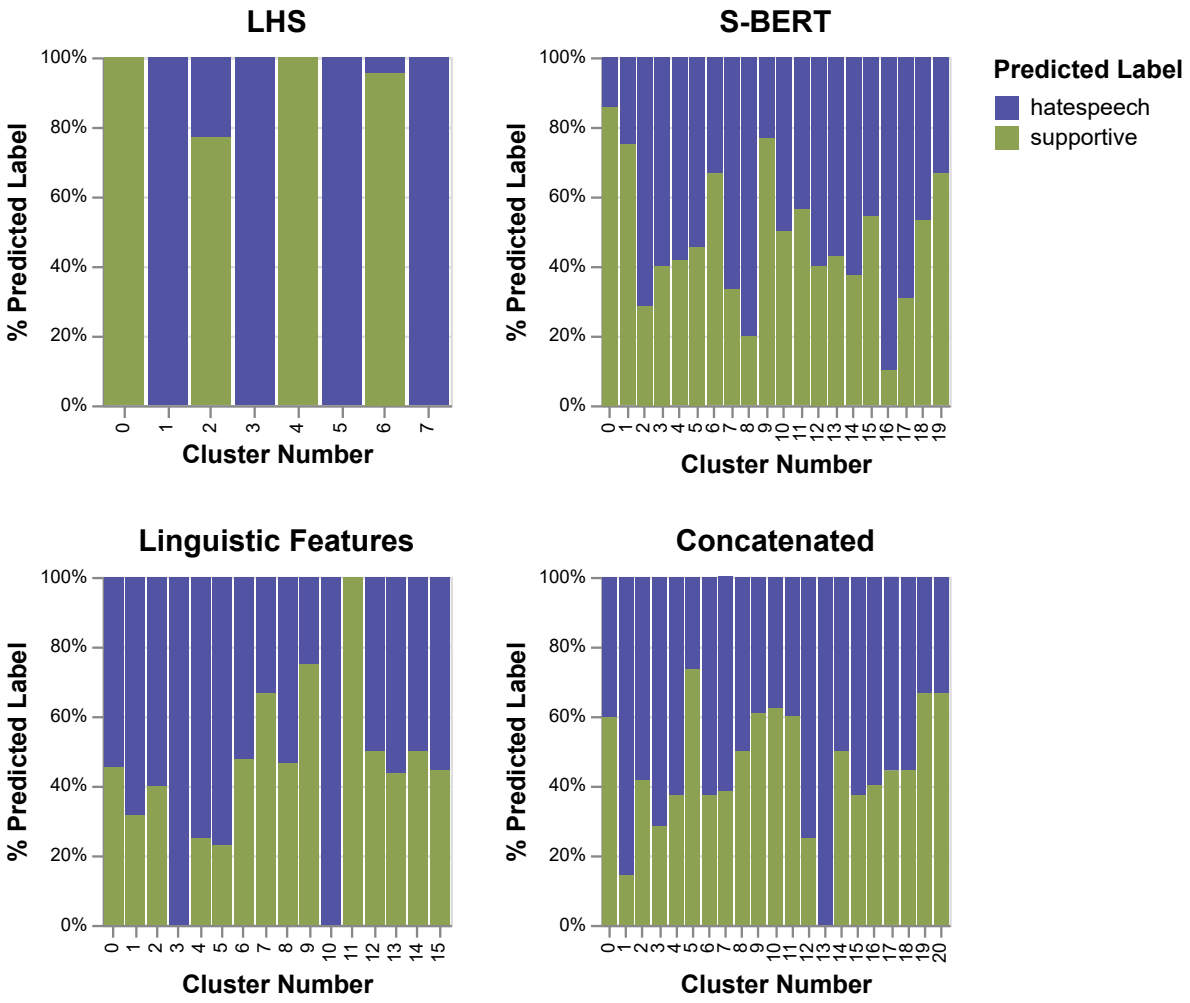


Figure 5: Predicted label distribution for English clusters.

Survey	Average Kappa
English LHS	0.176
English SBERT	0.484
English linguistic features	0.232
English concatenated embeddings	0.538
Arabic LHS	0.199
Arabic SBERT	0.130
Arabic linguistic features	0.231
Arabic concatenated embeddings	0.314

Table 5: Cohen's Kappa score between annotators on error intrusion task for evaluating the clustering.

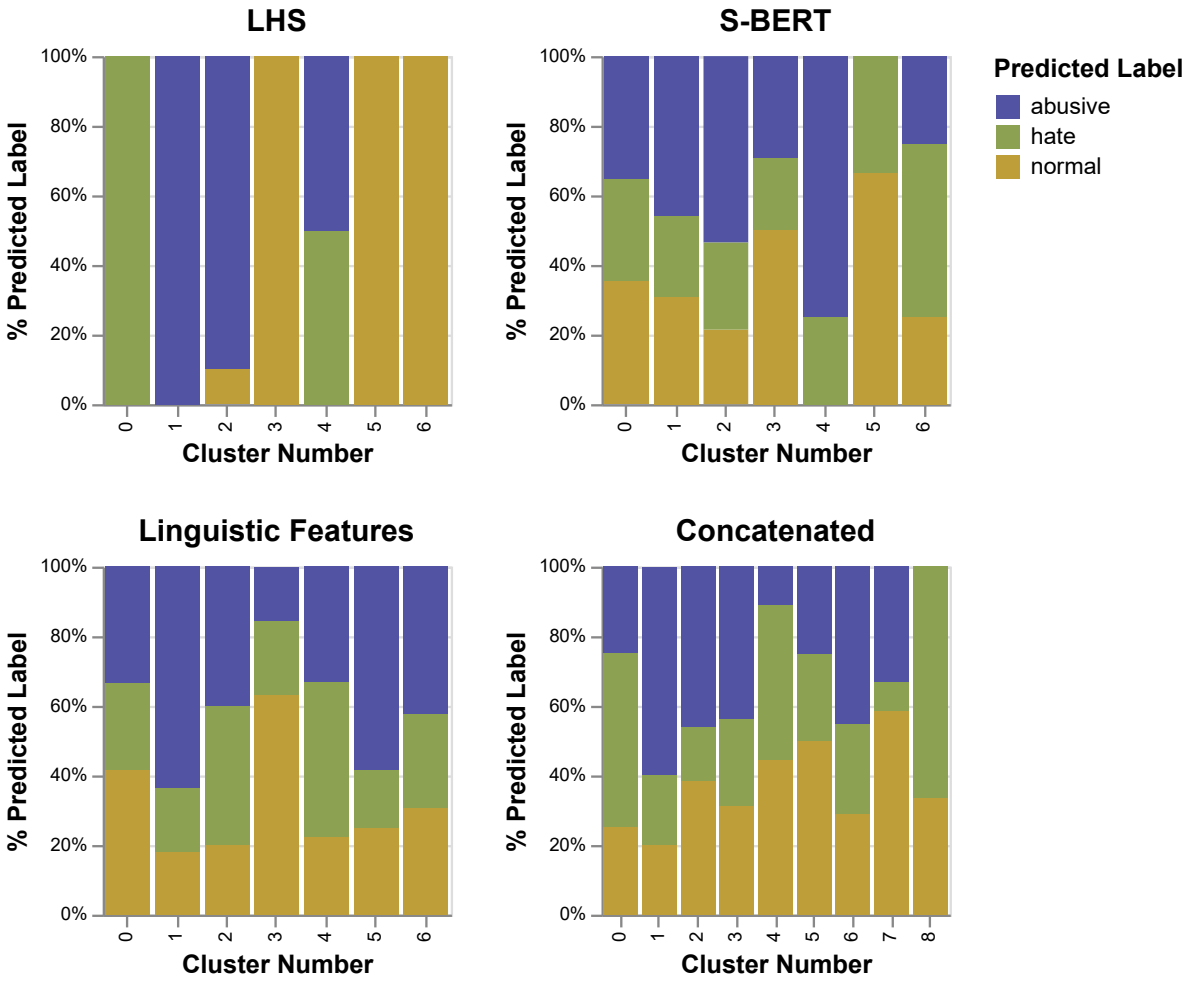
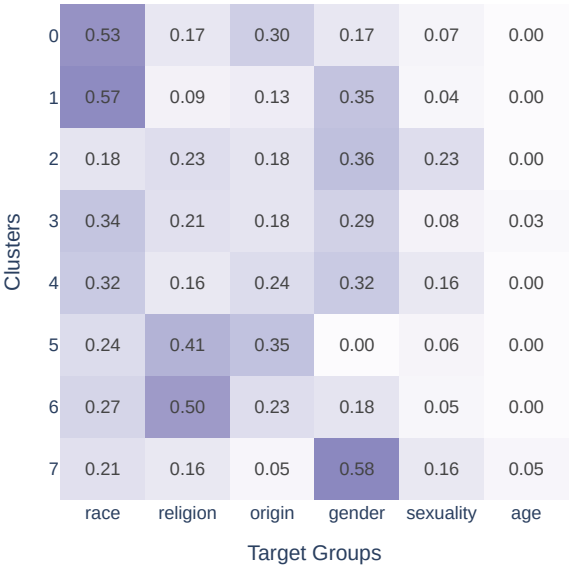
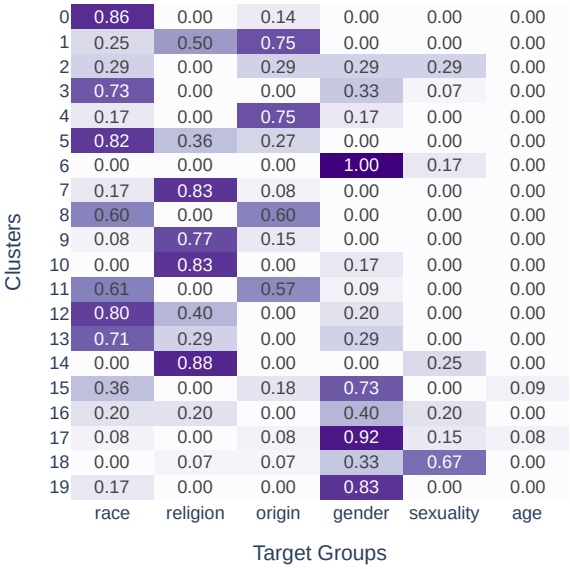


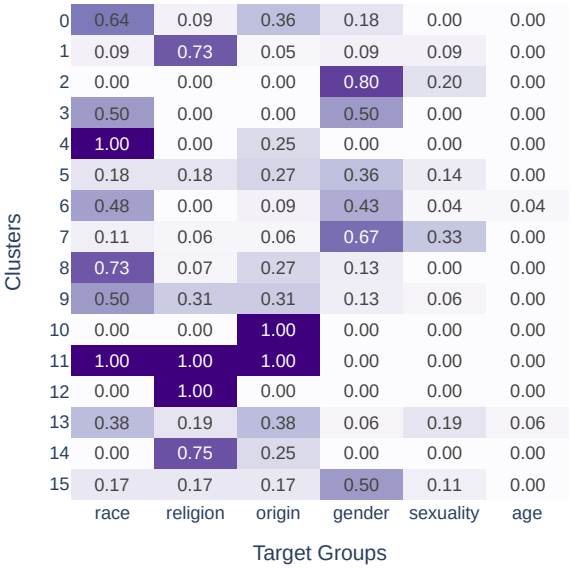
Figure 6: Predicted label distribution for Arabic clusters.



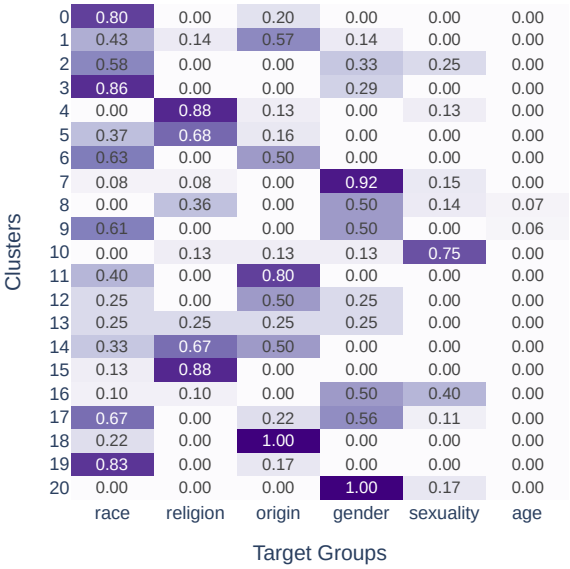
(a) Last Hidden State



(b) Sentence-BERT (SBERT)



(c) Linguistic features



(d) Concatenated (SBERT and Linguistic features)

Figure 7: Target group distribution for each embedding representation.