Aryan Thakkar: 903754913/athakkar37

Prabhav Agrawal: 903855202/pagrawal99

Georgia Institute of Technology

CS 4365/6365: Intro to Enterprise Computing

Summer 2025

Trending-Term LLM Drift Explorer

Aryan Thakkar and Prabhav Agrawal

Aryan Thakkar: 903754913/athakkar37

Prabhav Agrawal: 903855202/pagrawal99

Table of Contents

Aryan Thakkar: 903754913/athakkar37

Prabhav Agrawal: 903855202/pagrawal99

1 Introduction

In today's digital world, language and culture are evolving at an extremely rapid pace. Social Platforms such as Reddit and BlueSky host millions of new posts each day, giving way to new slang, memes, political jargon, and trending hashtags all that have the potential to spread virally in hours. On the other hand, large-scale NLP models like GPT-3, BERT, LLaMA are all typically trained on data that is "frozen" at a particular point in time. As a result, these models treat fresh terms as out-of-vocabulary (OOV) or assign them low-weight embeddings, misinterpreting or altogether missing their intended meaning.

This rapid drift between "real-world" language usage and NLP model's internal vocabulary can end up having serious downstream consequences. For example moderation systems could fail to catch harmful slang, misinformation tools could overlook newly coined terms, or chatbots could simply misunderstand users. Outdated language knowledge in machine learning pipelines can very easily undermine performance, fairness, and safety.

This is where our project comes in, Trending-Term Drift Explorer. We intend to surface and quantify exactly how out of touch an NLP model is at any moment. By continuously harvesting the latest trending terms from Reddit and BlueSky, comparing them against a model's tokenizer, and scoring each term's novelty, we can reveal vocabulary gaps in real time, compare multiple models' coverage, and provide an interactive dashboard to track vocab drift over time.

2 Motivation

As language continues to evolve rapidly online, NLP models risk falling behind, especially when it comes to understanding emerging slang, euphemisms, or culturally shifting terms. This gap between real world usage and model comprehension is not just a linguistic issue, it has concrete consequences for safety, fairness, and functionality in AI systems. Our Project addresses these concerts by providing insight into how well these models handle this trending language.

Misinformation & Moderation: Fact-checkers and moderators have grown reliant on NLP pipelines to flag misinformation. If models are unable to parse new slang or certain "code words" then they could miss harmful narratives

Fairness & Bias Audits: Language shifts very often are representative of social changes. A phrase that had no ill intent a couple months ago could be used in a hostile manner today. Therefore users need to know when a model's vocabulary is not updated.

Model Retraining & Fine-Tuning: Data Science teams need a metric to decide when to fine-tune/retrain models. The drift scores our model will provide offer objective triggers for retraining.

Linguistic Research: Researchers looking into sociolinguistics will be able to see how quickly terms transition from obsolete to mainstream, and how NLP models lag behind.

AI Product Safety & UX: Voice assistants, chatbots, and autocomplete tools risk misunderstanding users or providing incorrect outputs if they can't recognize trending terms. Vocabulary drift metrics help product teams evaluate when model updates are needed to maintain a smooth user experience.

3 Related Work & Starting Point

Many studies have acknowledged model staleness, especially in LLMs which have been trained on fixed datasets. **Touvron et al. (2023)**, describes how models like LLaMA have become outdated because of their reliance on this static data. Even though there has been extensive research done on topics such as concept drift and factual knowledge obsolescence, little to no papers focus specifically on vocabulary drift, especially related to emerging slang, memes, and netspeak.

Other studies such as **Chang et al. (2023),** explore dataset drift in broader terms and introduce metrics for vocabulary and semantic drift. However, they analyse carefully curated benchmark datasets, and do not capture new and upcoming slang/memes from real time social media. Similarly, **Sun & Xu (2022)** and **Keidar et al. (2022)** examine slang and semantic change over time, but focus on long-term evolution using old data or dictionary definitions, rather than new terms gaining popularity online in a matter of hours.

Research has highlighted that LLMs like GPT-3, BERT, and LLaMA are trained on a large fixed dataset. This results in models that are not up to date with ongoing language evolution **(Touvron et al., 2023**). Some LLMs, however, are periodically updated/ fine-tuned, but these updates can be infrequent and are often opaque causing slang or emerging hashtags to remain unrecognized for months at a time **(Hombaiah et al., 2021)**. Most existing studies focus on performance degradation due to concept drift or outdated factual knowledge, but very few examine vocabulary related drift caused by new slang or coded language in real-time social platforms.

Benchmarking datasets such as GLUE and SuperGLUE also fail to reflect trending words or phrase usage. Tools like embedding projectors and token attribution methods give insights into model internals but are not designed to monitor how well models adapt to new vocab in a time-sensitive context.

In terms of tooling, platforms like **DRIFT (EMNLP 2021)** provide mechanisms to detect semantic shifts in scientific texts, but once again are not built to track slang or model misalignment related to new and emerging terms. Likewise, **Spataru et al. (2023)**
Define semantic drift in context of generation degradation over time but do not tackle vocabulary mismatch with user input.

In summary, there is:

- **No continuous, real-time measurement** of how emerging terms are tokenized or semantically interpreted by LLMs;
- **No public-facing platform** that shows how NLP models fall behind the evolving discourse on social platforms;
- **No operational scoring mechanism** that teams can use to determine when model retraining is needed based on vocabulary misalignment

Our project aims to fill this gap by continuously scraping BlueSky and Reddit, collecting trending phrases, analyzing their tokenization and semantic representation across major LLM, and scoring their drift. By doing this, we provide a new lens into model relevance and robustness. Our goal is to build an automated, reproducible system that identifies when models are out of touch.

4 Project Architecture and Methodologies

For this project, our ultimate goal was to provide 4 key artifacts:

1. **Trending-Term Drift Dashboard:** A dashboard allowing users, with access to a hugging face account and a Gemini API key to input any trending term they would like and view how it is tokenized, embedded, and understood by a set of popular LLMs. This tool will allow users to view drift scores and comparative metrics.
2. **Core Automated Data Pipeline:** In order to have our model up to date with the newest trending terms, we wanted to create a pipeline that could continuously scrape trending terms from Reddit and Bluesky, which would allow users to continuously scrape new trending terms and test model drift in the future. The pipeline scripts would allow for scraping and filtering of data to create effective datasets.
3. **Model Drift Sores:** We will provide a scoring methodology that will quantify vocabulary drift across the different models we are using. Scores will include token novelty, embedding distance, and prompt accuracy. These will then be visualized across different models to illustrate drift.
4. **Comparative Evaluation Report:** A written analysis of multiple LLMs, highlighting which ones are more resistant to vocabulary-drift and which fail to capture evolving terms. Additionally, commentary would be provided on the training type of the model as well and transformer architecture to demonstrate those impacts on model performance.

These artifacts served as a guideline for how we approached researching such an expansive topic. Our final workflow was separated into three separate tasks across the project timeline: **Data Scraping and Cleaning**, **Metrics Creation**, and **Model Metrics Visualization and Analysis**.

4.1 Data Scraping and Cleaning

For data scraping and cleaning, we successfully implemented scraping pipelines for Reddit and Bluesky through their API tools to collect high frequency words used by their users. For reddit, we extracted trending posts from multiple subreddits and processed the content to isolate uncommon terms with known definitions and usage examples. For Bluesky, we developed a similar pipeline that scrapes the top 1000 high engagement posts based on post features, including likes, reposts, and replies. Both pipelines obtained the top N frequent words (where N is some positive integer determined by user needs) include profanity checking and output a cleaned dataset in csv format providing the terms, frequency count of the word, definition, and contextual sample. If a definition or example did not exist for a word, from the dictionaries API or if the definition was incorrect, the definitions and examples would have to be manually to ensure valid data.

4.2 Metrics Creation

In terms of drift evaluation, we focused on three complementary methods. First, we examined **token drift**, where we count how many tokens a model uses to represent a slang term. We found three main sources to calculate token count for a variety of models. Those sources were Tiktoken (OpenAI's open source library for tokenizing text for models like GPT-4o, GPT 4-Turbo, GPT 3.5, and more), Huggingface's `AutoTokenizer.from_pretrained(model_name)` function (allows you to use various Transformer models from Huggingface), and Google's Gemini `count_tokens(text: String)` function. To calculate the drift between two models, we want to calculate the ratio of the difference, meaning if one model tokenized a word using two tokens and the other model did so using three tokens, the drift would be |2-3|/max(2,3), resulting in ⅓ or .3333. As a result, we effectively defined the tokenization drift between two models as:

$$\frac{\left|T_1 - T_2\right|}{max(T_1, T_2)}, \text{where } T_n \text{ is model n's token}$$

Secondly, we calculated **semantic drift** analysis. Here, we researched how we could utilize Gemini Embedding API, and Hugging Face transformers to retrieve vector embeddings of the same word from different models. By computing the euclidean distance between the L2 norm of the embeddings, we were able to quantify how closely the models align in meaning. Larger distances indicate greater drift in understanding.

$$drift\_score = euclidean\_distance(e_1, e_2) / (||e_1|| + ||e_2||),$$

$$where\ ||e_n|| = \sqrt{(\Sigma e_{ni}^{2})}$$

Third, we explored **definition drift**, which involves prompting the models to define and use slang words in context. For instance, we asked a model to define "chopped" and compared its response to definitions from the reference definition created during the data pipeline stage. We then used SentenceTransformers (specifically the "all-MiniLM-L6-v2" model) to compute cosine similarity between definition embeddings and ROUGE-L scoring to measure lexical overlap. The final definition similarity score was calculated as:

$$definition\_similarity = (cosine\_similarity\_score + rouge\_L\_score) / 2$$

Where the cosine similarity captures semantic alignment between model-generated and reference definitions, while ROUGE-L measures the longest common subsequence overlap, providing both semantic and syntactic evaluation of definitional consistency across models.

4.3 Model Metrics Visualization and Analysis

For the final analysis, due to the computational limits, time requirements, and api rate limits to get embeddings, text generation, and token counts, we extracted 50 terms from each dataset sourced from both social media platforms, to create a collective 100 words to test our various drift metrics. For each model we calculated the token count, semantic drift, and definition drift for each possible pair of models. We use those results to create multiple visualizations using python's matplotlib library to understand different correlations and draw important insights.

5 Results & Analysis Discussion

This section presents our results related to vocab alignment and drift across multiple LLMs using real world slang, which we scraped from both BlueSky and Reddit. We analyze model behavior using 3 core metrics: **Token Count, Definition Similarity,** and **Embedding Drift**, providing a multi-faceted picture of how up to date these models are.

5.1 Correlation Between Tokenization and Definition Understanding

We begin by examining the relationship between how models tokenize a term and how well the model actually captures the term's meaning. In other words, we want to understand whether breaking a term into more subword tokens affects the model's ability to form a coherent, accurate embedding.

Figure 1 Plots each mode's average token count for trending terms (X-axis) against the average cosine similarity between the model's embedding of each term and a corresponding human-written reference definition (Y-axis).

In this plot we observe a strong negative correlation. This means that models that split trending terms into more tokens tend to produce embeddings that are less semantically aligned with the true meaning of the term. When a model splits a trending term into multiple unfamiliar subwords, it typically means the term wasn't seen during training. As a result, the model assigns it an embedding based on loosely related fragments, leading to poor alignment with the term's actual meaning.

Older and smaller models like gpt-2 and SmolLM2-135M averaged around 2.42-2.47 tokens per term and low semantic similarity scores, which were around 0.25-0.29. Meanwhile, newer models like Gemini-2.5-flash and Gemini-2.5-Pro tokenize terms more efficiently at around 1.91 tokens and achieve a lot better similarity scores. This suggests that they have been updated to better recognize and represent new or trending terms.

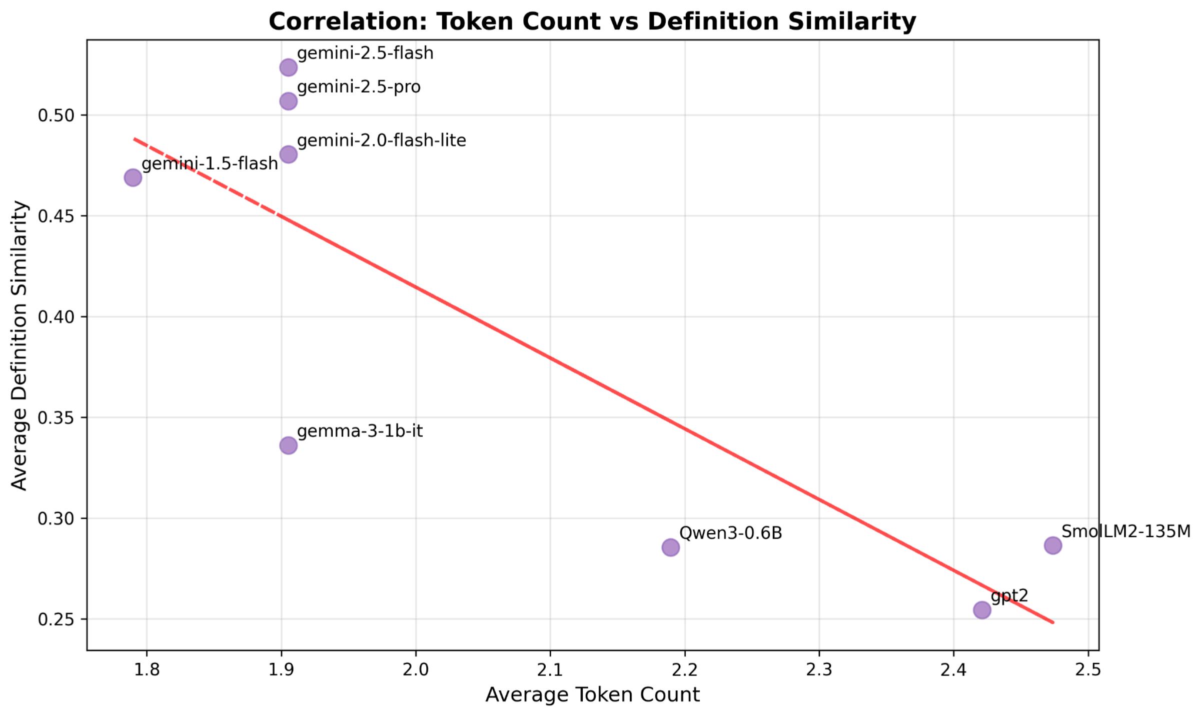


Figure 1: Correlation Between Average Token Count and Definition Similarity Across Models

5.2 Average Model Definition Similarity to Reference Definition

We also looked at how well each model could generate accurate definitions by comparing their outputs to a reference set we built manually. Figure 2 shows the average definition similarity for each model, combining both ROUGE-L and cosine similarity to measure how close the model's definition was to the reference. Higher scores mean the model's response was more in line with the actual meaning of the word.

Unsurprisingly, the Gemini models came out on top. Gemini-2.5-flash and Gemini-2.5-pro had the highest scores, 0.524 and 0.507 respectively, which tracks given their size and recent updates. These models have over a billion parameters and clearly seem tuned to handle new slang and evolving language better than the others.

On the other hand, GPT-2 had the lowest score at 0.254. This makes sense since it's a much older model and mainly built for text generation. It behaves more like an advanced n-gram model—good at predicting next words but not so great at capturing deeper meaning, especially with terms it wasn't trained on. What's interesting is how SmolLM2-135M performed. Despite being tiny compared to the others, it still beat GPT-2 and was about on par with Qwen3-0.6B, scoring 0.286. That shows that even super lightweight models can perform above their weight when designed or fine-tuned well. Gemma-3-1b-it landed somewhere in the middle at 0.336, which makes sense since it's instruction-tuned and probably more focused on following prompts than defining slang.

Overall, this graph shows that newer and bigger models generally do better at defining trending terms, but it also highlights how model purpose and training strategy matter just as much as size.

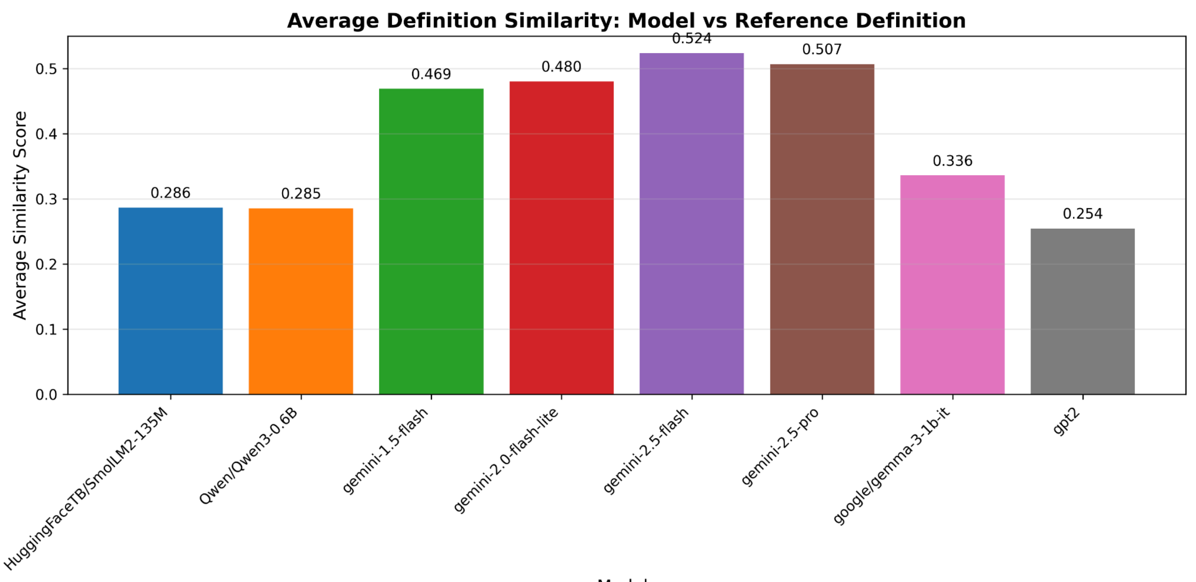


Figure 2: Average Definition Similarity between Model and Reference Definition for Each Model

5.3 Model Definition Drift

While Section 5.2 focused on how well each model matched a human written reference definition, here we explore how closely each model aligns with other models in terms of semantic understanding. Here we are measuring definition drift, or the average cosine distance between how each model and its peers embed trending terms. A higher drift indicates less agreement with others about what a term means, which may reflect outdated vocab, weak generalization, or divergence in training objectives.

5.3.1 Gemini-2.0-flash-lite (Figure 3)

This model shows consistently low drift against other Gemini models, with values around 0.10, and moderate drift with newer baselines like Gemma-3b (0.203). It diverges the most with GPT-2, however compared to the rest of the Gemini model's divergence to GPT-2 it is lower.

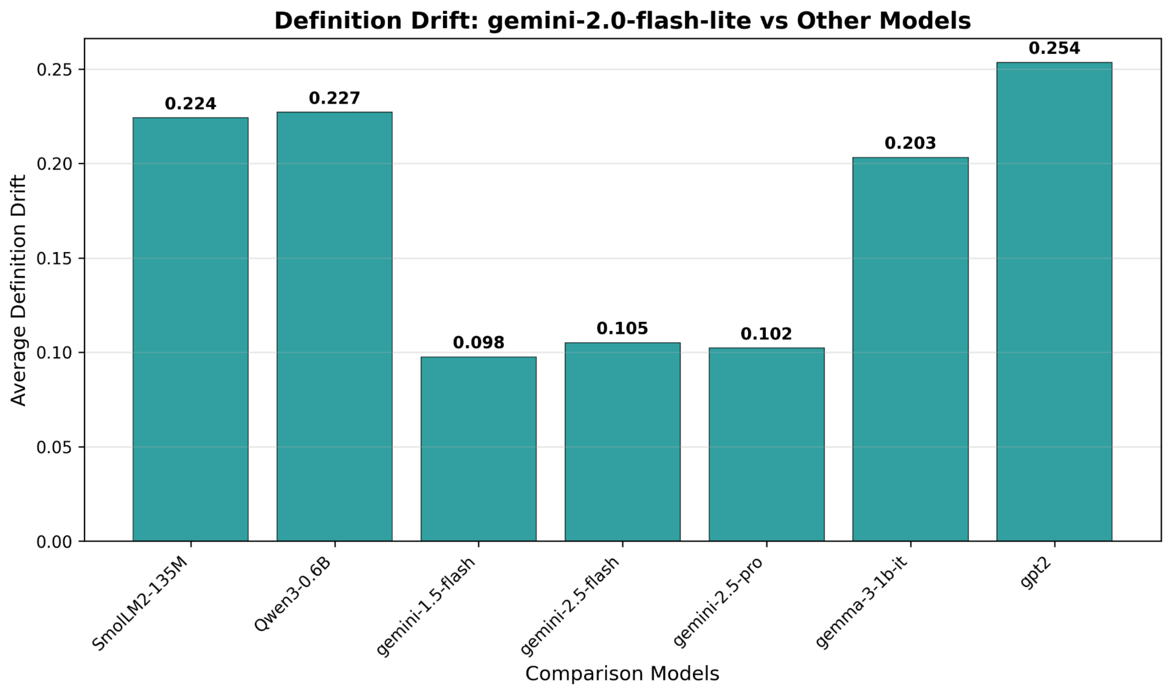


Figure 3: Definition Drift for Gemini-2.0-flash-lite vs Other Models

5.3.2 Gemini-2.5-flash-lite (Figure 4)

Gemini-2.5-flash shows the smallest drift across the Gemini family and the lowest pairwise drift at 0.078 with Gemini-2.5-pro. It's largest divergences are, again, with GPT-2 and Qwen3, reinforcing how disconnected older/smaller models are from modern embeddings

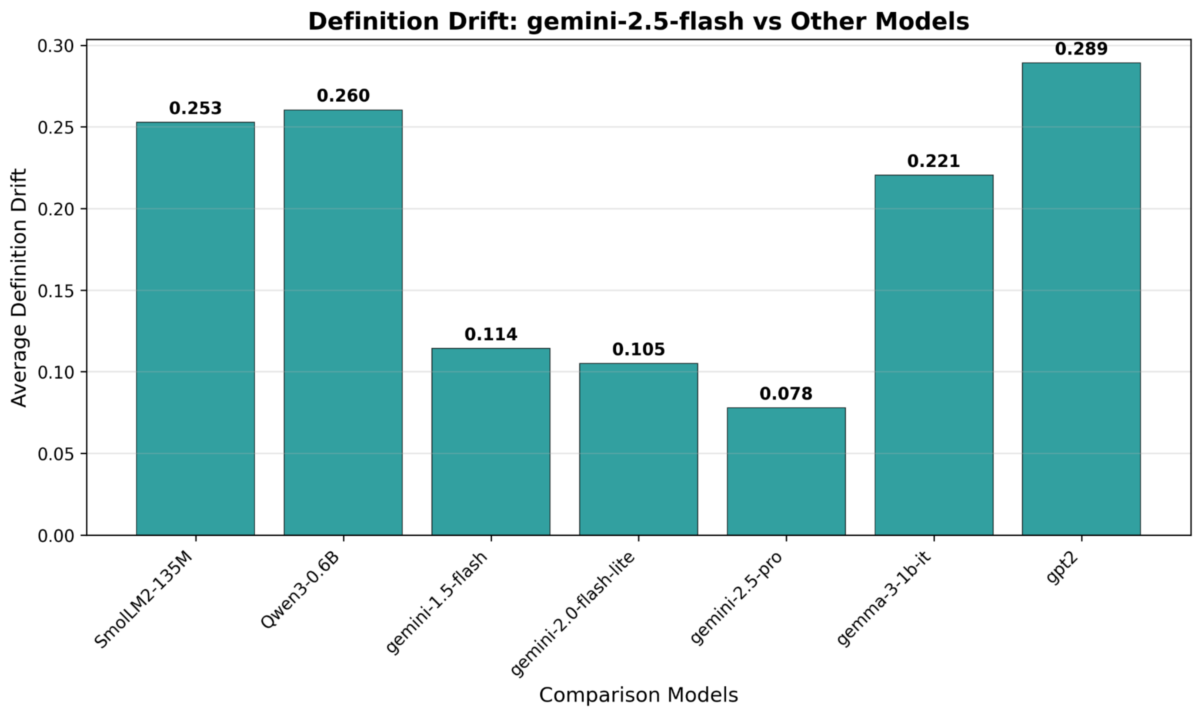


Figure 4: Definition Drift for Gemini-2.5-flash vs Other Models

<u>5.3.3 Gemini-1.5-flash (Figure 5)</u>

Despite being the earliest Gemini model in this analysis, Gemini-1.5-flash shows strong alignment with the later Gemini versions. However it can be seen that there is a higher drift with older models like GPT-2. This indicates that Gemini-1.5 is well integrated into the Gemini family's embedding space, but has already diverged significantly from earlier-generation models.

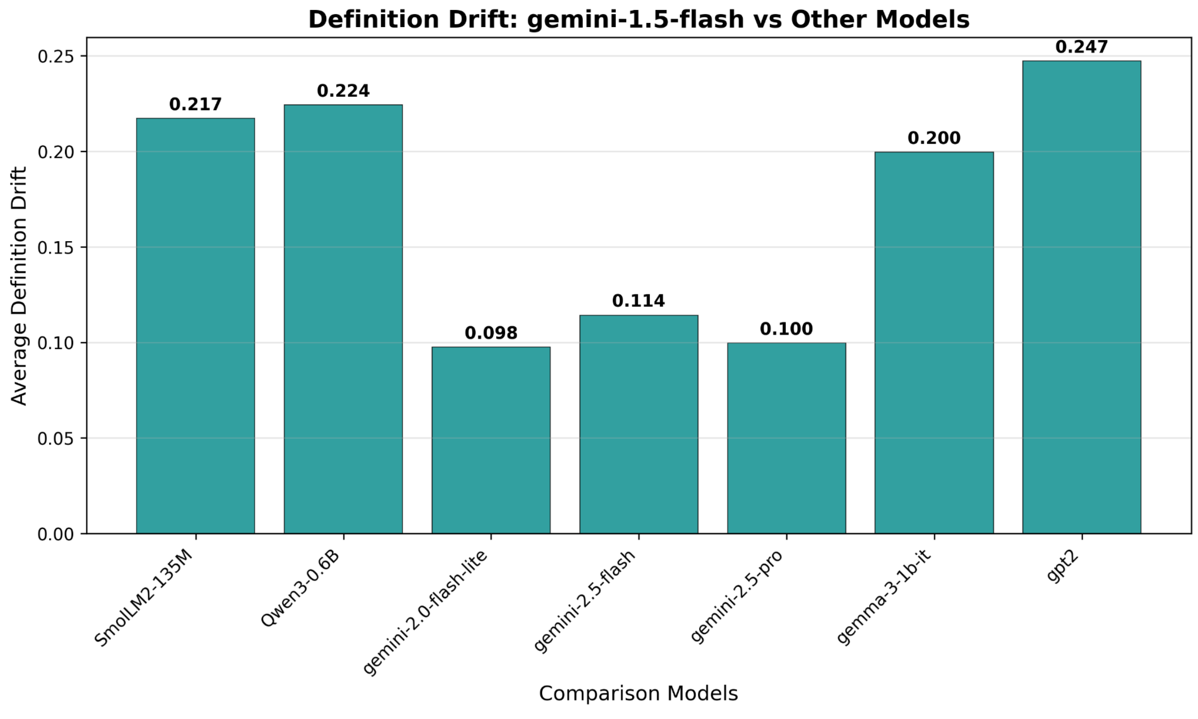


Figure 5: Definition Drift for Gemini-1.5-flash vs Other Models

5.3.4 SmolLM2-135M (Figure 6)

SmolLM2 aligns most closely with GPT-2 and Qwen3, drifting substantially from all Gemini models (0.22-0.25). Its position suggests it occupies a shared, older semantic space that may struggle with newer or evolving terms.

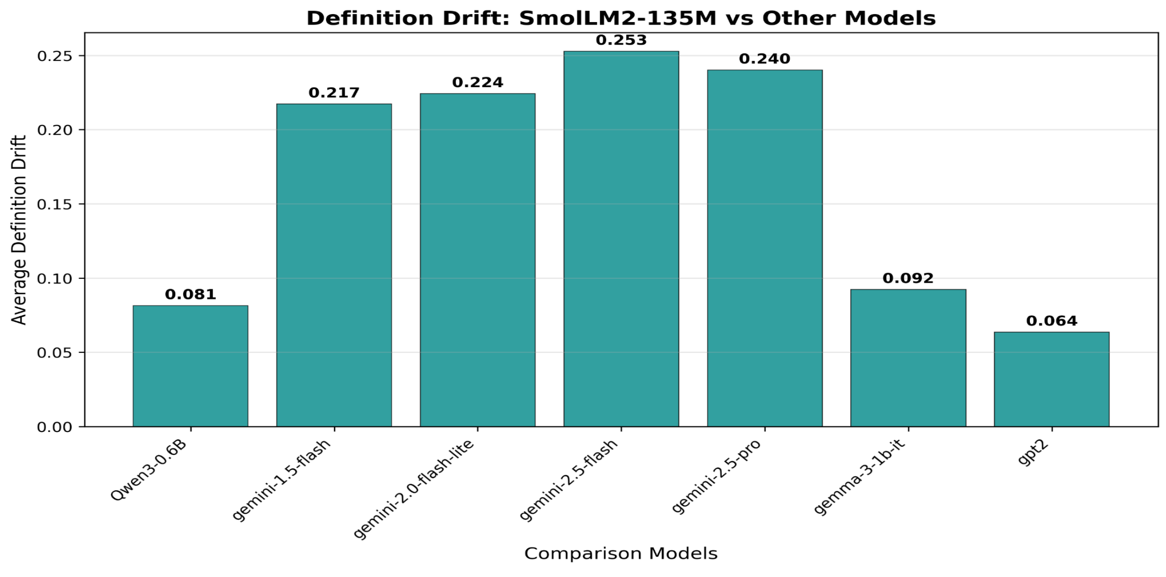


Figure 6: Definition Drift for SmolLM2 vs Other Models

5.3.5 Qwen3-0.6B (Figure 7)

Qwen3's alignment is almost identical to that of SmolLM2's. It shows low drift with GPT-2 and SmolLM2 but diverges sharply from Gemini-2.5 models. Interestingly, it performs slightly better in matching human-written definitions than its drift would suggest. This could indicate that it may capture meaning differently despite being semantically distant from the larger models.

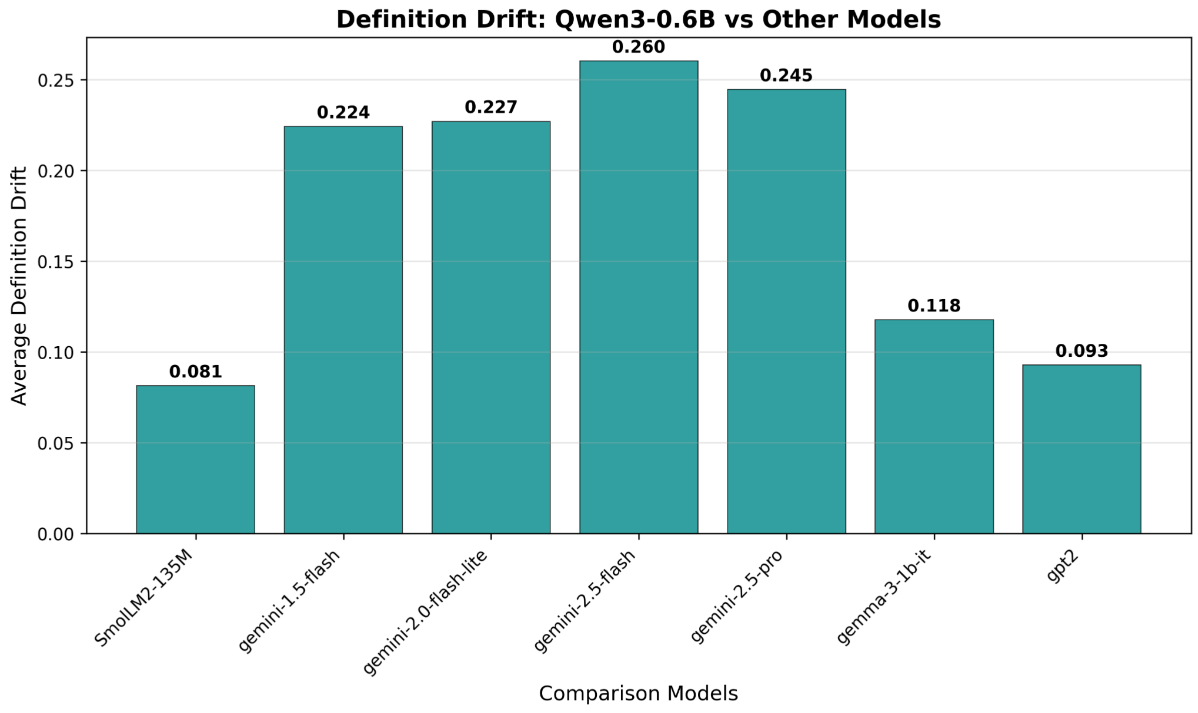
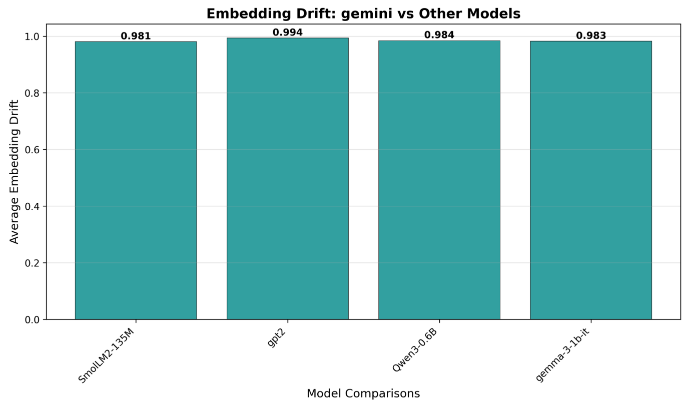
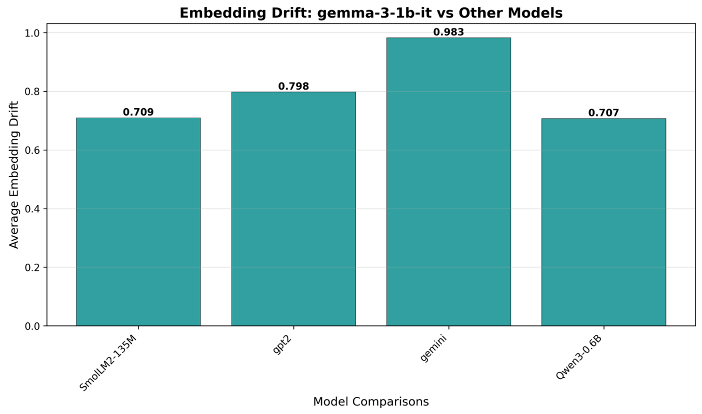


Figure 7: Definition Drift for Qwen3-0.6B vs Other Models

5.4 Model Semantic Embedding Drift

To evaluate how consistent each model's word embeddings are compared to others, we looked at the average embedding drift across all pairwise model comparisons (figure 8.). Across the board, Gemini had the highest average embedding drift, scoring over 0.98 against every other model, including SmolLM2-135M (0.981), GPT-2 (0.994), Qwen3-0.6B (0.984), and Gemma-3-1b-it (0.983), which tells us that Gemini's internal understanding of trending terms is significantly different from the rest. On the other side, SmolLM2-135M showed much closer alignment with smaller models like Qwen3-0.6B (0.725) and Gemma-3-1b-it (0.709), while still staying under 0.81 even with GPT-2. Interestingly, Gemma-3-1b-it and Qwen3-0.6B had the lowest drift with each other at 0.707, suggesting that their embeddings for trending terms are the most similar. GPT-2, while older, still maintained decent consistency with Qwen and Gemma, averaging around 0.79–0.80, but drifted heavily from Gemini. These results suggest that even though newer models like Gemini perform well on definition tasks, they encode term meanings much differently than older models. .





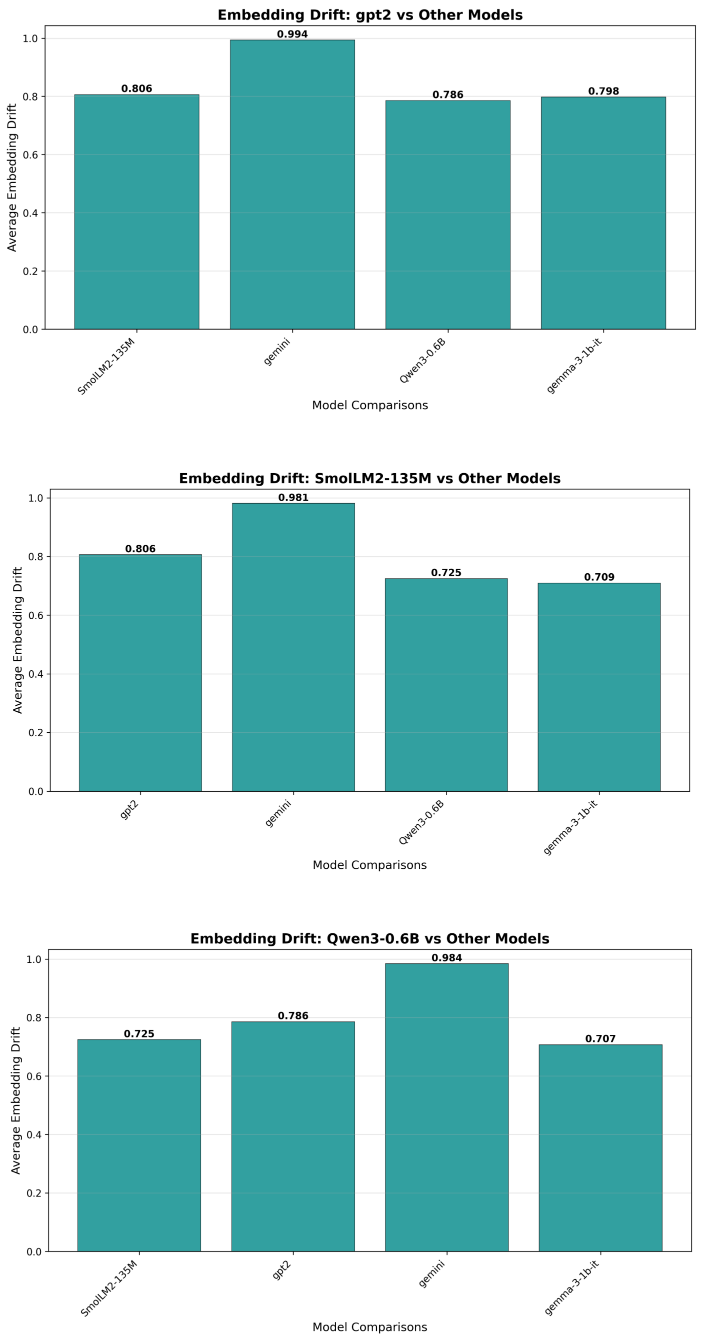


Figure 8. Mode Drift Comparisons for Each Model Between Other Models

5.5 Model Token Counts and Drift

The heatmap below (figure 9.) shows the token drift matrix across 10 different models, with warmer colors indicating greater divergence in token count behavior. A few trends stand out immediately. First, Gemini-1.5-flash and SmolLM2-135M show the highest drift relative to the rest of the group, especially with each other (0.25), suggesting they tokenize trending terms in very different ways. On the opposite end, models like GPT-3.5-turbo, GPT-4, and Qwen3-0.6B show near-zero drift among themselves, reflecting stable and consistent tokenization strategies. The GPT-4o family also shows slightly better token efficiency than earlier GPTs, while Gemini-2.5-flash stays closer to mid-range models but still drifts more from the larger GPTs. Overall, this visualization reinforces that while newer models have converged somewhat on tokenization norms, outliers like Smol and early Gemini variants still behave quite differently.

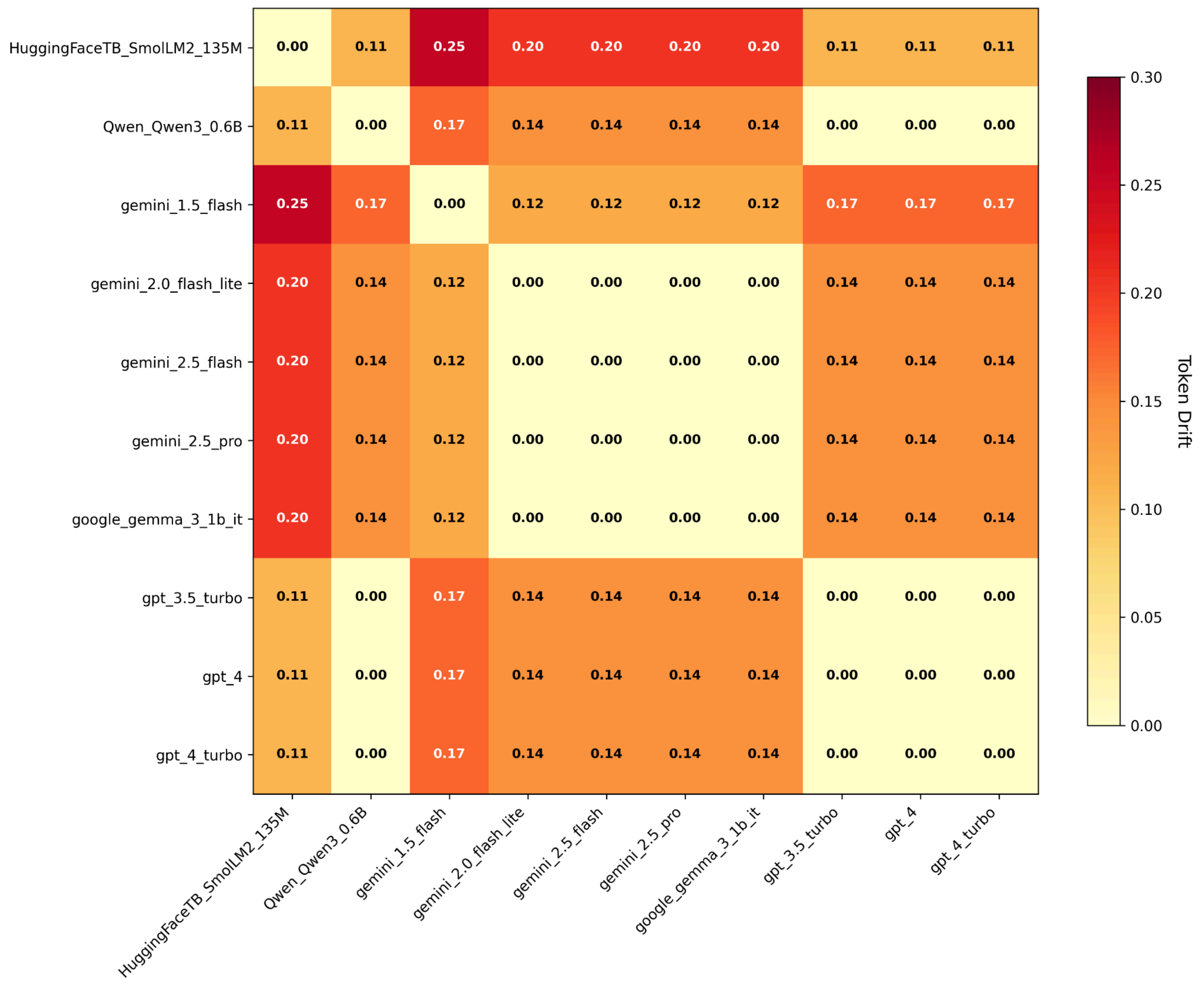


Figure 9. Token Drift Heatmap Between Various Models

5.6 Domain Specific Model Comparisons

Gemini Models (Token Count + Definition Similarity)

Across Gemini models, we noticed a clear improvement in both tokenization efficiency and semantic understanding as the versions advanced. Gemini-1.5-flash had the lowest average token count (1.79) but also the weakest definition similarity (0.469). Starting from Gemini-2.0-flash-lite and onward, the token count leveled out at 1.91, while definition scores improved significantly, peaking with Gemini-2.5-flash at 0.524. This suggests that while tokenization stabilized, newer versions continue to improve at capturing the actual meaning of trending terms.

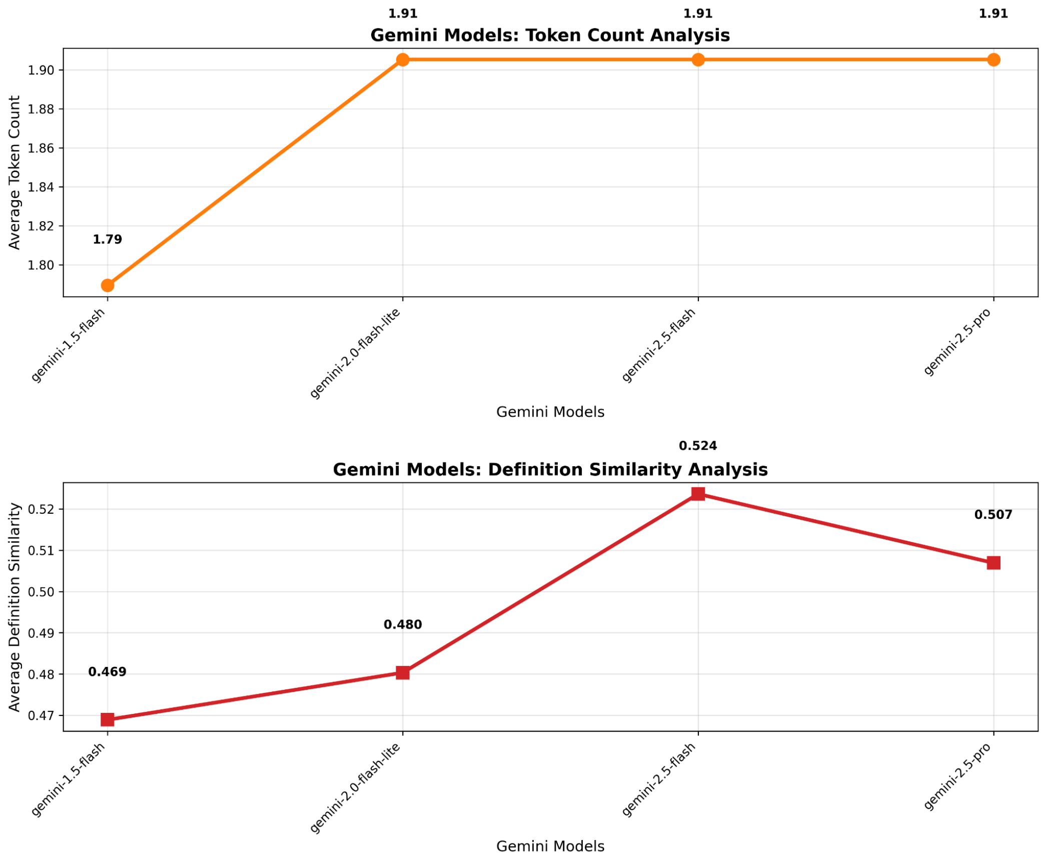


Figure 10. Average Token Count and Definition Similarity for Gemini Models

<u>GPT Models (Token Count Over Time)</u>

In the GPT family, we observed a steady downward trend in average token count as models evolved. GPT-2 had the highest token count at 2.42, showing it splits words into more subword units, which often weakens semantic alignment. From GPT-3.5-turbo to GPT-4o-mini, the token count dropped to around 2.09, indicating that newer models have a larger vocabulary and are getting better at compact tokenization.

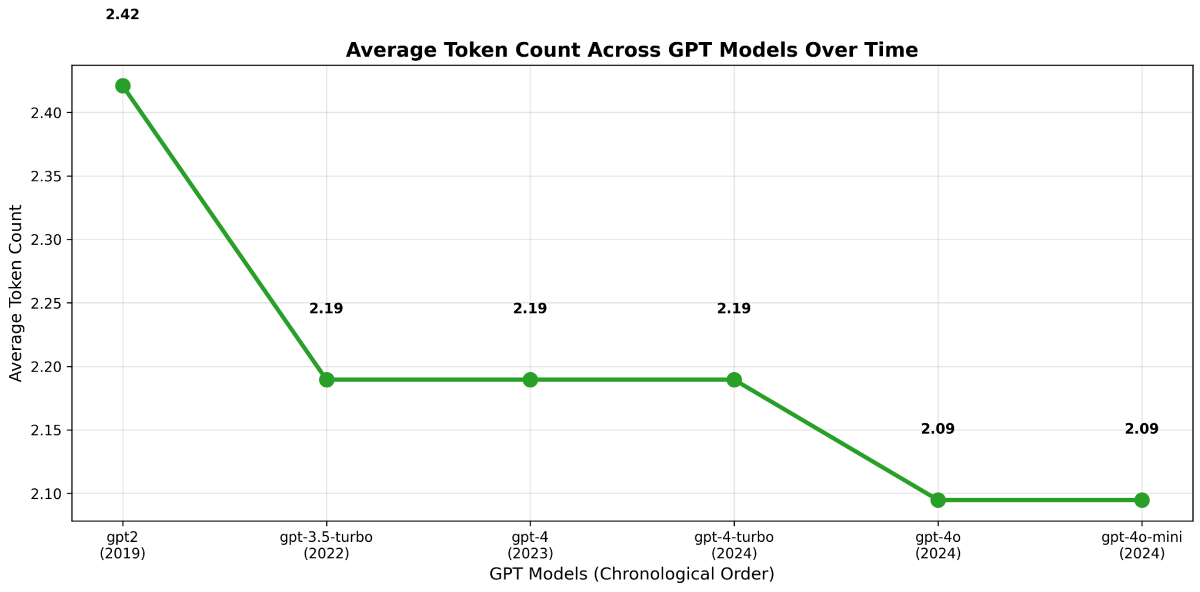


Figure 11. Average Token Count for GPT Models over Time

6 Blockers

During the course of our project, we faced several key obstacles as described below. However, we were able to effectively maneuver around them by adapting our project and continuing to move forward.

1. **Twitter API Cost**

During the research phase of the project, we found out that the Twitter API actually costs to use, which would make it extremely hard for us to scrape a high number of posts. We quickly pivoted and decided to use BlueSky as our second social media platform.

2. **Tiktoken Provided Little Context into Model Drift**

While researching possible metrics to use to differentiate different models by performance (e.g. calculate model drift), we were originally only going to use tokenization drift. However, we realized for short length slang terms, tokenization would not provide that much variance. This prompted us to pursue metrics including semantic similarity and definition similarity, where these features when combined with tokenization drift would allow for better insights.

3. **Cosine Similarity for Semantics Embedding Similarity**

Another issue we faced is that when we created embeddings using different models and found the cosine similarity between them, the results were often close to zero. As a result, it took numerous attempts of trial and error, starting with padding the vectors and finding the euclidean distance, and eventually finding the magnitude of the vectors, and finding the difference between the two magnitudes to see how large of a difference it was.

4. **Storing and Using Transformer Models with a High Number of Parameters**

To use a specific transformer from Hugging Face, it was required to download the model and store it on your computer’s cache. Models under 1 billion parameters would take up to 15gb of space while models over that could take up hundreds of gigabytes, limiting us in the number of models we could use. Additionally, using the text generation feature for large models was also limited due to the sheer amount of time it would take to load and use the model.

7 Conclusion

This project helped us understand how well different language models keep up with evolving vocabulary. Using token drift, semantic drift, and definition drift as our core metrics, we were able to measure how each model handled trending terms across multiple dimensions.

Our results showed that newer models like Gemini-2.5-flash and GPT-4o consistently performed better across all three metrics, recognizing and representing new language more accurately. In contrast, older models like GPT-2 struggled with both tokenization and meaning, often breaking terms apart or misunderstanding them entirely. One surprising finding was that SmolLM2-135M, despite its small size, performed more consistently than expected in both token and semantic drift. This reinforced the idea that model performance is shaped by more than just parameter count.

Overall, the drift scores and visualizations we built made it clear which models are best equipped for real-world language—and which ones are starting to fall behind.

8 Future Work

While our system successfully measures vocabulary drift in trending terms and compares performance across multiple LLMs, there are several directions in which we would like to explore.

1. Expand to More Domains and Languages

Our analysis focused on English slang from Reddit and BlueSky, but vocabulary drift happens globally and across specialized domains. Future versions could expand to different languages, or maybe even different social media platforms. Additionally, from a model standpoint, having funding to have more computing power and explore more models would allow us to study models like OpenAI's GPT, Anthropic's Claude, Meta's Llama, and much more.

2. Real-Time Monitoring and Alerts

Currently, our data scraping and drift scoring pipeline runs offline. We aim to build a real-time version that continuously takes in trending terms and sends automated alerts when drift exceeds a certain threshold. This could help teams know exactly when to retrain their models or apply patch updates.

9 Final Deliverables

1. Data Pipeline
    a. Bluesky
    b. Reddit
2. Metrics Creation and Visualization
3. Dashboard

10 Metaskills

Over the course of this project, we developed a wide range of skills spanning numerous fields.. Below, we summarize the key metaskills that shaped our growth:

1. **Data Engineering for Evolving Platforms**

Scraping live data from social platforms like Reddit and BlueSky was far from plug-and-play. We had to create scripts that handled API authentication, rate limiting, and profanity filterings—while also enriching the output with contextual examples and validated definitions (often which for some slang terms, official definitions did not exist). This process taught us how transformer noisy data is not a one stop process and requires multiple iterations of perfection and manual work.

2. **Metric Design and Evaluation**

Rather than relying on pre-existing benchmarks, we had to invent new drift metrics from scratch that would help evaluate how "out-of-date" a model was in terms of vocabulary. This included designing formulas for token count change, semantic drift via embedding vectors, and definition-level similarity. Constructing these metrics taught us how to be intentional about what we measure and how to ensure metrics actually reflect real-world performance gaps. A lot of research went into studying various metrics to ensure they provided variance in their output and allowed for effective differentiation.

3. **Data and Toolchain Integration**

We worked across a diverse stack that included PRAW, Hugging Face, Gemini APIs, SentenceTransformers, OpenAI's Tiktoken library, and Python visualization tools. Learning to move between and understand these tools and more importantly, stitching them together into something meaningful was a major takeaway.

4. **Debugging Complex Workflows**

Whether it was catching errors in our definition drift scores or resolving inconsistencies in the scraping pipeline output, we ran into plenty of situations where things didn't behave as expected. These moments helped us become more comfortable debugging at the level of both code and logic, identifying where the issue stemmed from and how to pivot from there.

5. **Insightful Extraction from Visualizations**

Working with 100+ term-model pairs and multiple scoring metrics created a sea of numbers. Visualization became our tool for turning that chaos into clarity. We learned to research ideas we didn't know to answer questions like: What does a higher token count mean for a model? How do certain models cluster in performance?

6. **Working with Ambiguity**

There was no textbook on how to measure trending-term LLM drift. The open-ended nature of our project forced us to take initiative, experiment with different approaches, and adapt when early ideas didn’t hold up. Learning to be comfortable with ambiguity required turning the project into a puzzle, requiring complex research and out-of-the-box thinking.

11 References

1. Chang, K. W., Linzen, T., & Zettlemoyer, L. (2023). *Characterizing and Measuring Linguistic Dataset Drift*. Proceedings of ACL 2023. https://arxiv.org/abs/2305.17127
2. Hombaiah, U., Liu, P., & Lin, H. (2021). *Time-Aware Language Modeling for Tracking Concept Drift in Social Media*. Proceedings of NAACL 2021.
3. Keidar, D., Dagan, I., & Goldberg, Y. (2022). *Slangvolution: A Causal Analysis of Semantic Change and Frequency Dynamics in Slang*. arXiv preprint arXiv:2203.04651. https://arxiv.org/abs/2203.04651
4. Spataru, A., Choudhury, M., & Narayan, S. (2023). *Know When to Stop: Detecting Semantic Drift in Neural Text Generation*. arXiv preprint arXiv:2404.05411. https://arxiv.org/abs/2404.05411
5. Sun, Y., & Xu, W. (2022). *Tracing Semantic Variation in Slang*. arXiv preprint arXiv:2210.08635. https://arxiv.org/abs/2210.08635
6. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., … & Scao, T. L. (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv preprint arXiv:2302.13971. https://arxiv.org/abs/2302.13971