AI–Driven Commerce: Integrating Men's Health Articles with Relevant Products Using Large Language Models

Kyle Sullivan
University of Pennsylvania
kylems@seas.upenn.edu

Abstract

Hearst's encompassing suite of magazine brands, each touching on a variety of topics, makes mapping articles to highly granular, hierarchical product taxonomies particularly challenging. This project aims to utilize a trio of models (rule-based, embeddings, and LLM) to categorize Men's Health articles by the products, items, and concepts they discuss. All these models were evaluated against seventy-two human-labeled articles, with label agreement between at least two-thirds of the taggers constituting ground truth. The rule-based model supplied a baseline performance marker but failed to tag articles as accurately as embeddings and LLM-based models, which had greater levels of agreement with the ground truth data than human annotators had among themselves. Ultimately, the best-performing model was a hybrid approach, using a more conservative subset of taxonomy categories chosen by the embedding model as input to the LLM.

1 Background

Hearst has partnered with Wharton Analytics to help them connect their magazine articles to a given, mutable product taxonomy for the Men's Health shop. These tags will enable targeted advertising of relevant products in a new online marketplace. In the future, Hearst hopes to adapt this model to additional magazines and growing taxonomies. Therefore, the final model must be both flexible and scalable.

As an extra hurdle, this tagging process differs for commerce articles (containing affiliate links), which require a narrower mapping, only picking out taxonomy categories matching products for sale. Hearst cannot sell advertisements competing against the article content, making missed commerce tags problematic. On the other hand, overtagging commerce articles limits Hearst's ability to sell ads for these wrongly included, likely highly related products. While neither outcome is desirable, Hearst suggested erring toward the latter case when balancing these tradeoffs. Contrarily, non-commerce articles need to be mapped to any taxonomy category mentioned, regardless of whether the corresponding product or concept is the article's primary focus or just casually mentioned. Since the supplied dataset does not make this distinction, a prior classification task is required to differentiate commerce from non-commerce articles.

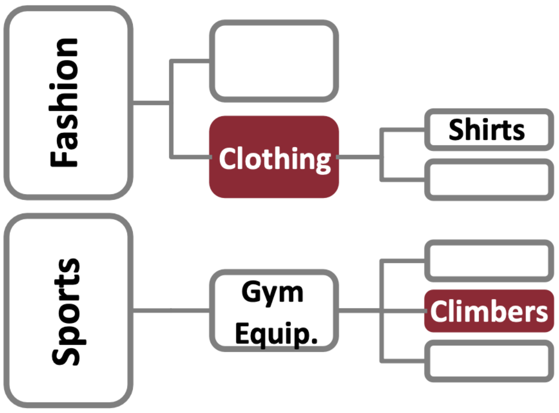


Figure 1: Simplified Taxonomy Hierarchy Diagram

2 Data

Hearst presented four datasets—just over 5.5k English-language web-scraped articles, separate article features, product information, and taxonomies. The feature data contained IDs, publishing dates, title and body text, URLs, formats (standard article, long-form article, listicle, etc.), and metadata. The metadata contained all the above information, with additional components that required further exploration. The discovery that a gallery section, present in over 85% of articles, contained product IDs mapping to Hearst's product dataset enabled more reliable identification of commerce articles than checking for populated product fields alone. Overall, this gallery finding increased the identified share of commerce instances within the 5.5k articles, which jumped from 25% to 36% of the dataset. Merging product data also unlocked additional product name and description information, which were labeled and appended at the end of commerce article bodies to aid the article's classification. Finally, the web-scraped data was more accurate than the feature dataset's body text, so the latter was dropped in favor of the former.

The supplied taxonomy is highly granular and hierarchical—four tiers (see *Figure* 1 for a mini diagram). Throughout the modeling process, the structure was flattened, with all level tags considered potential matches. The goal was to map to the lowest (most specific) level tags possible, which proved challenging. For instance, an article on dumbbells could be classified as *Sports and Outdoors*, *Weight Training*, *Free Weights*, or *Dumbbells*—each being a subset of its precursor. While some articles support tags at the lowest terminal nodes, less specific connections should include appropriate higher-level categorizations. With 174 current terminal tags and 211 total (flattened) tags, this represents a sizeable multiclass classification problem. The terminal tags are often very similar, making the task even more difficult. For example, the taxonomy includes *Men's Shoes* and *Men's Active Shoes*, *GPS Watches* and *Smart Watches*, and *Men's Board Shorts* and *Men's Swim Bottoms*. Making these distinctions was hard for human annotators, highlighting the intricacy of the classification process.

3 Modeling

3.1 Rule-Based

Serving as the initial model explored, rule-based classification is simple and limited. The idea is to search each article, checking if taxonomy tags appear verbatim, with any findings returned as matches. The biggest problem with this approach is that it's very rigid. Synonyms or slightly different wordings go undetected. Despite these restrictions, a rule-based model was primarily explored to establish a performance baseline, upon which other more elaborate methodologies could be compared.

A few variations of rule-based models were attempted, including unigram, bigram, and full-tag comparisons (after removing unnecessary words from tags such as "Men's"). The unigram and even bigram models were too noisy. Applying these techniques to long, multiword labels such as *Ab and Core Training Equipment* generated unreliable results—similarly, investigations into using GPT-generated synonyms as additional keywords failed to improve performance. Ultimately, simple full-tag searches were used as the baseline model in later evaluations.

	Commerce Percentile	Non-Commerce Percentile	Min Cosine Similarity for Normalization	Cosine Similarity Tag Threshold	Z-score Tag Threshold
V1	99	95	0.25	0.45	1
V2	100	100	0.25	0.45	1
V3	100	100	0.3	0.4	1
V4	100	100	0.35	0.45	1.5
V5	100	100	0.45	0.45	1
V6	99	95	0.45	0.45	1
V7	95	90	0.45	0.45	1
V8	90	90	0.45	0.45	1
V9	80	80	0.45	0.45	1

Figure 2: Embedding Versions Table

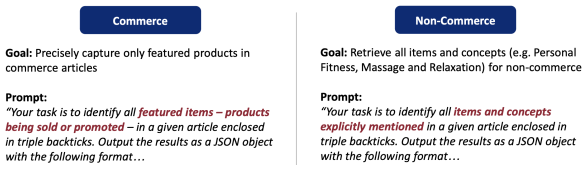
3.2 Embeddings

The primary advantage of embeddings over rule-based models is their ability to capture contextual meaning and generalize to synonyms. They convert words to numeric representations, effectively capturing underlying text semantics. Passages are transformed into large numerical vectors corresponding to spatial points. These high-dimensional coordinates group similar words together and encode term-to-term relationships. As a result, article text embeddings can be compared to taxonomy embeddings via cosine similarity. Anything scoring above a chosen threshold is then returned as a match.

A crucial factor when running an embedding model is selecting the optimal context window size. In an early approach, Word2Vec was used to encode individual words, but this approach removed all context, thereby limiting the benefits to term generality. Subsequent approaches, however, used OpenAI's embedding model (*text-embedding-3-small*) to convert larger swaths of text into vectors. Brief experiments revealed that encoding entire articles or even paragraphs at once distilled too much information into a single point in space. For instance, paragraphs mentioning a variety of clothes scored poorly against these individual tag embeddings. While paragraphs proved too large and individual words too small, sentence-level embeddings performed well in preliminary tests, leading to their exclusive use in more extensive evaluations later.

Calculating embeddings at the sentence level necessitates an additional layer of aggregation. Rather than averaging scores or taking the max across all sentences, a more flexible approach was opted for. Cosine similarity scores above a chosen percentile cutoff (typically around 0.9, with the option to set different values for commerce articles) were averaged, with tags returned as matches when they exceeded predetermined cosine similarity or z-score thresholds (in regards to the distribution of tag scores, with each aggregated across all sentence-level scores using the above method). As a further layer of customization, these z-scores were calculated on a subset of the tags. Typically, only tags with scores above some small positive fraction were used in the normalization calculation to avoid numerous irrelevant tags from boosting the z-scores of semi-related categories. See *Figure* 2 for a detailed breakdown of the various embedding hyperparameter choices used in subsequent evaluations.

With such a highly granular taxonomy, an additional filter was implemented to remove similar, highly correlated tags. Since cosine similarity scores were calculated over small, sentence-level windows, it was possible to determine if tags scored well over the same short portions of the article. The lower-scoring tag was dropped if the correlation between the two terms exceeded a very conservative cutoff (0.98) and the cosine similarity score between both tags was greater than another marker (0.9). Due to these high bars, only a few tags were removed in the test data. Nevertheless, this marked an initial attempt to address highly similar taxonomy categories.


Commerce

Goal: Precisely capture only featured products in commerce articles

Prompt:
*"Your task is to identify all **featured items – products being sold or promoted** – in a given article enclosed in triple backticks. Output the results as a JSON object with the following format…*

Non-Commerce

Goal: Retrieve all items and concepts (e.g. Personal Fitness, Massage and Relaxation) for non-commerce

Prompt:
*"Your task is to identify all **items and concepts explicitly mentioned** in a given article enclosed in triple backticks. Output the results as a JSON object with the following format…*

Figure 3: LLM Commerce vs. Non-Commerce Prompts

3.3 Large Language Model

While embeddings excel at capturing semantic matches, they lack the refinement of large language models, which can more reliably distinguish between similar, related categories. These large language models (LLMs) are fundamentally text generators. Given system and user prompts, they respond based on the most probable (depending on hyperparameters adjusting randomness) subsequent sequence of tokens learned from their enormous training data. They are fed entire articles—the title, body, and any associated product information (names and descriptions)—along with the taxonomy categories and a set of instructions. Using this information, they generate the most likely ensuing text. Given the task of identifying corresponding tags, the LLMs output a final set of matches.

As a leader in the large language model space, OpenAI provides a variety of LLM tiers with mixed pricing and efficiency tradeoffs—an ideal starting point for this project. GPT 3.5 Turbo (*gpt-3.5-turbo-0125*) and GPT 4 Turbo (*gpt-4-0125-preview*) were used exclusively; however, with OpenAI updating models regularly, there will likely be cheaper, more powerful offerings in the near future. Together, these supplied two different budget options, with GPT 3.5 Turbo being significantly more affordable and GPT 4 Turbo being much more powerful. All experiments were run with both to see if the added price was justified for this assignment.

Considered as much an art as a science, the key task associated with LLMs is to fine-tune the optimal set of instructions. This process, known as prompt engineering, involves designing the overall assignment layout and carefully selecting the words used in the commands. LLMs are highly responsive to subtle phrasal tweaks, particularly in complex problems. The success of these applications often hinges on the quality of the instructions provided.

GPT 3.5 Turbo and GPT 4 Turbo introduce a new JSON mode, which requires the LLM to present its responses in JSON format. This feature enforces a problem-specific configuration and facilitates easier result parsing. Furthermore, this structure helped leverage GPT's output token log probabilities to generate confidence scores. Given the context, these log probabilities represent the LLM's likelihood of generating a particular token next. A binary output system, therefore, can be used to obtain prediction probabilities of the tagging results. They are derived by mapping categories to 0/1 integers and checking GPT's corresponding token probabilities. Using words like "Found" or "True" was avoided as they could lead to incorrect probability calculations, given that different cases (e.g., "FOUND" or "TRUE") could also be considered meaningfully likely (non-zero probability) occurring tokens, skewing the associated probabilities.

After lengthy experimentation, different prompts were used for commerce and non-commerce articles. In the former case, GPT was asked to identify all featured items—things being sold or promoted. In the latter instances, it was more broadly told to find any items or concepts explicitly mentioned. This distinction reflects the different goals for each article type laid out in *Figure* 3. In particular, including "concepts" in non-commerce prompts aided the retrieval of more general, non-terminal taxonomy categories such as *Massage and Relaxation*, *Personal Fitness*, *Fashion*, and *Beauty and Grooming*, which wouldn't be considered "items." After describing the types of things to find, a JSON object with the tags in the desired format was included along with instructions to swap out 0 values (which were pre-loaded) for 1s if the category appeared (or was featured) in the article. GPT was told the article would be enclosed in triple backticks to clarify its start and end positions. Then, the article was given as a user message.

As an additional version, another prompt format also requested text citations for the found items/concepts. Instead of just mapping categories to 0/1, each included "found" and "citation" sections, which would be mapped to 0/1, and a brief passage from the text highlighting the items. All "found" categories required a citation. The hope was that this would force GPT to consider why things were being tagged while providing a level of explainability that could be useful in auditing the results when selecting the final article tags used in practice.

Most articles only touch on a narrow scope of subjects, with many of the supplied taxonomy categories clearly irrelevant. Rather than pass in all tags to the LLM for every article, another implementation sought to narrow the search by first performing an embedding tag-preselection process. Slightly more conservative embedding model hyperparameters were chosen to avoid missing correct tags. Nevertheless, this two-step approach significantly reduced the number of tags shown to the LLM in most cases. The proposed benefits were twofold. First, this would greatly reduce costs since the LLMs were set up to return all input categories with binary found indicators. This large output is significant since the text generation cost is several multiples higher than the input cost. Secondly, this limited selection would hopefully reduce input noise and opportunities to hallucinate classifications.

Finally, it's important to note that GPT 3.5 Turbo and 4 Turbo are non-deterministic, meaning they don't always produce the same results across runs. Even with the temperature and top p set to 0 each time and using seed parameters to attempt similar sampling, the models output slightly different results. These variations often manifest via different prediction probabilities, some of which may affect the final tags. While this can make model evaluation challenging and potentially leave an impression of unreliability or lacking robustness, these three hyperparameters and tag preselection offer the best solution for now.

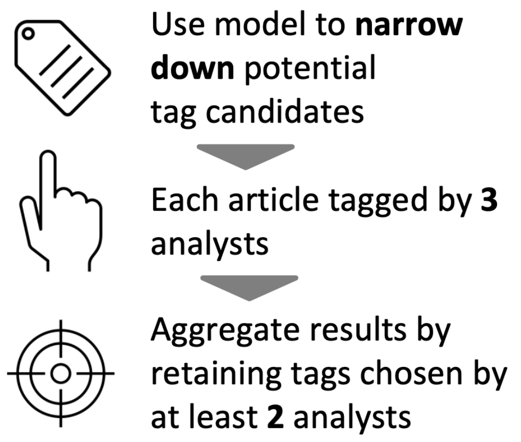
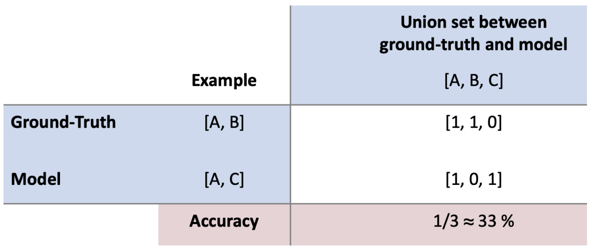

Use model to **narrow down** potential tag candidates
Each article tagged by **3** analysts
Aggregate results by retaining tags chosen by at least **2** analysts

Figure 4: Gold-Label Generation via Human Annotation Process

3.4 Human-Labeled Data

One of the most significant project obstacles was the lack of labeled data. Without gold labels, model evaluation and subsequent tuning would be extremely limited. To remedy this constraint, team members manually tagged a subset of articles with three humans assigned to each of the seventy-two articles. While more data points would have been preferable, time constraints and prioritizing quality over quantity prevented further manual tagging. Since the taxonomy is lengthy, very conservative subsets of tags were shown to labelers as potential options. However, this was only a guide. All annotators also had access to the complete taxonomy in its initial hierarchical form, with similar subcategories grouped for convenience. A mix of commerce and non-commerce articles was chosen, including a blend of article types (standard, long-form, and listicles). These hand-labeled articles generated ground truth labels, defined by tag agreement between at least two-thirds of the annotators. Rather than including all human-selected tags as gold labels, this agreement criteria helped prevent incorrect potential over-tagging from polluting the evaluation results. See *Figure* 4 for a diagram of the process.



	Example	Union set between ground-truth and model
		[A, B, C]
Ground-Truth	[A, B]	[1, 1, 0]
Model	[A, C]	[1, 0, 1]
	Accuracy	1/3 ≈ 33 %

Figure 5: Accuracy Diagram

4 Evaluation Methodology

Three evaluation criteria were used to assess model performance: standard accuracy, AUC, and Krippendorff's Alpha. The human annotators, however, were only evaluated using Krippendorff's Alpha, which assesses inter-rater agreement.

4.1 Standard Accuracy

The first evaluation metric considered is an iteration of standard accuracy that computes the quotient of the number of correct tags divided by the total number of tags. Being a one-to-many classification problem, defining "correct" and "all" tags is more complex. "Correct" tags were defined as any labels in both the model and ground truth outputs—their intersection. On the other hand, "incorrect" tags were considered labels present in only one of the outputs. Thus, "all tags" included the tags across both outputs—their union. See *Figure* 5 for a basic example.

In summary, this simple accuracy measures the set intersection divided by the set union. This approach penalizes over-tagging by growing the denominator and penalizes under-tagging by limiting the size of the numerator. Additional versions that credited non-tag agreement (things excluded in both models) were considered but not reported, as they led to artificially higher scores.

4.2 AUC (Area Under the ROC Curve)

AUC metrics analyze prediction tradeoffs between sensitivity and specificity by calculating the area under the receiver operative characteristic (ROC) curve. It "shows the tradeoff between the true positive fraction (TPF) and false positive fraction (FPF) as one changes the criterion for positivity." (Hajian-Tilaki, 2013) For multiclassification, this process takes prediction scores for tags as inputs, along with the gold-label data. It then returns a value between 0 and 1, indicating how well the model distinguishes between different categorizations. Models should, at the very least, exceed 0.5, which represents performance on par with random guessing. (Çorbacıoğlu and Aksel, 2023) Only the LLM model was assessed using this evaluation metric, and it came with a few caveats. First, when using the embeddings to preselect input tags, all excluded tags had their prediction scores (probabilities) set to 0. Second, AUC calculations cannot accommodate instances with no tags despite this being a perfectly valid option for Hearst if none are found. Therefore, an additional "No Tags" category was made retroactively—the associated prediction probabilities were calculated based on the probability of all other tags being zero.

4.3 Krippendorff's Alpha

Finally, Krippendorff's Alpha evaluation metric enabled relative comparisons between models and humans. It is a "reliability coefficient developed to measure the agreement among observers... [to determine] how much the resulting data can be trusted to represent something worthy of analysis." (Krippendorff) This method works across multiple categories and various observers. Consequently, it helps evaluate model performance against the gold labels and human tagger agreement. Since the ultimate goal is to produce a model capable of human performance, this metric is the most valuable, as it uniquely allows for such a comparison.

4.4 Handling Parent Tags

Due to the tags' hierarchical nature, there were a couple of ways to deal with parent tags. In the standard approach, tags were compared as is, without adjustments. However, two other variations were used and often preferred. The first removed all parent tags when one of their children – a more precise tag – was also found. Conversely, the second approach added all parent tags to the selected matches. If their child matched, they naturally did too. This second version necessitated a minor prediction probability adjustment, where parent tags simply took the probability of their highest matching child. Both also altered the gold-label data generation process, as these alterations were applied to annotator tags before searching for two-thirds agreement among human labelers.

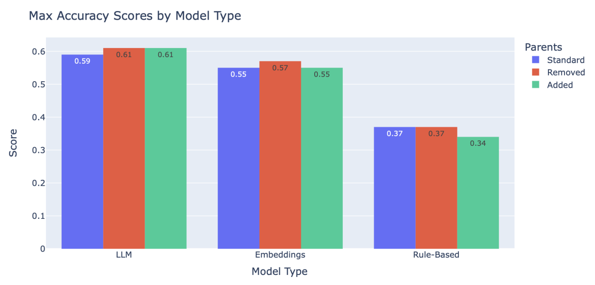


Figure 6: Max Accuracy by Model Type

4.5 Standard Accuracy

The standard accuracy scores in *Figure* 6 aligned with model expectations, with the LLM scoring highest and the rule-based approach falling well short of the other two. Seeing how well the embeddings performed relative to the LLM was a little surprising. While scores around 60% seem low, they reflect the difficulty of selecting among numerous, often very similar tags. With most of these results, relative comparisons are the most informative. Interestingly, the "adding parents" approach typically underperformed in relation to the "removing parents" technique, as the bigger union set grew the denominator faster than the numerator's additional parent tag matches.

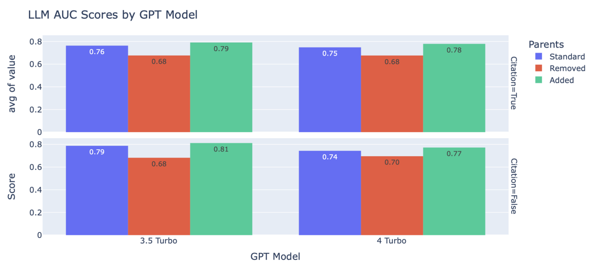


Figure 7: Average LLM AUC by GPT Model

4.6 AUC

The AUC values are moderate in *Figure* 7, just breaking 0.8 in the best case, placing them at the bottom of the "considerable" scoring range. (Çorbacıoğlu and Aksel, 2023) However, putting these values into perspective without computations for the other models is difficult. The more significant takeaway is that GPT 3.5 Turbo kept up with its more powerful GPT 4 Turbo sibling in this test, often outperforming it. An important note: these results represent the average performance when feeding the model embedding preselected tags to help narrow the scope. V3 through V9 were run twice for each GPT model (once for each citation/non-citation prompt approach).

4.7 Krippendorff's Alpha

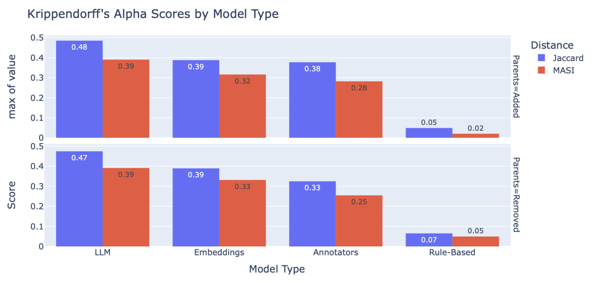


Figure 8: Krippendorff's Alpha Scores by Model Type

4.7.1 All Model Type Comparison

Evaluating all model types, including the annotator approach, emphasizes the difficulty of the classification task. It's important to note that the annotator score presented in *Figure* 8 is between all three human labelers, showing Krippendorff's Alpha level of agreement between people tasked with manually tagging these articles. The LLM posted a substantially higher level of agreement with the gold-label data than the annotators had among themselves. The embedding model also produced higher scores than the humans, showing stronger agreement with ground truth than the labelers had between their chosen tags. Interestingly, human scores were much higher in the "parents added" approach, whereas other models experienced minimal improvement, with the rule-based model's score even suffering a dip.

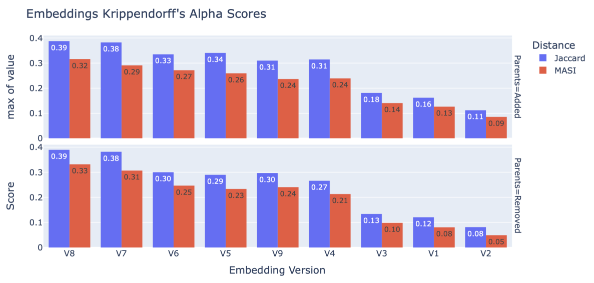


Figure 9: Embeddings Krippendorff's Alpha Scores

4.7.2 Embedding Scores Across Different Hyperparameter Choices

Evaluations were run across all hyperparameter variations shown in *Figure* 9. Large percentile thresholds produced the worst results, tagging anything with a single, high-scoring sentence. These outlier scores could result from unusual sentences or undetected non-article data collected while web-scraping. Lower cutoffs hovering around the 90th percentile performed much better, especially when paired with stricter cosine similarity tag thresholds and higher minimum normalization qualifying scores. Dropping the normalization component entirely through equality of these two cutoffs worked best. Anything scoring below a 0.45 cosine similarity was unlikely to be a match, and the lower bar led to many similar item tags.

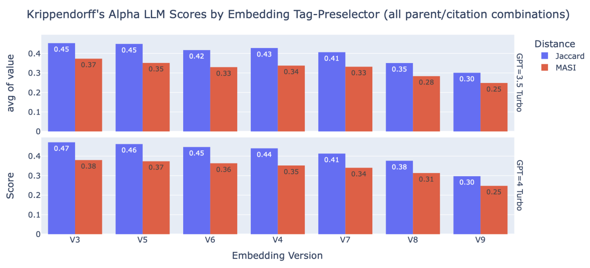


Figure 10: LLM Krippendorff's Alpha Scores Across Different Embedding Tag-Preselectors

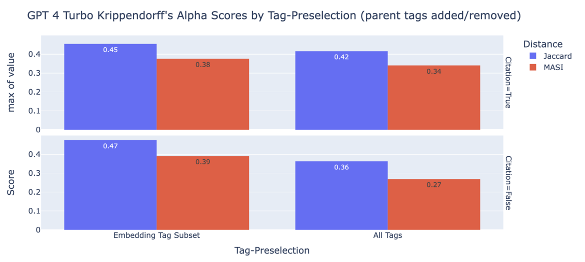


Figure 11: LLM Krippendorff's Alpha Scores Comparing Tag-Preselection vs. All-Tag Inputs

4.7.3 LLM Scores Across Different Tag-Preselectors

Despite being evaluated across nearly identical embedding hyperparameter versions, each acting as a tag-preselector, the LLM's behavior was markedly different (see *Figure* 10). The embedding model excelled with relatively lower percentile cutoffs, while the LLM preferred higher values. This choice comes with a set of tradeoffs. Although higher thresholds capture briefly mentioned items, they also tend to accumulate more noise since they don't require consistently strong tag scores across multiple sentences. In fact, with the percentile values set to one hundred, the tag preselector only considered the maximum scoring sentences for each tag. Unlike the embedding model, the LLM could filter this out.

Moreover, tag-preselection variants outperformed their all-tag counterparts as seen in *Figure* 11. Interestingly, this gap was even more pronounced without using the citation method. Counterintuitively, requesting this additional task across many categories also failed to produce better results after limiting the input tags via pre-selection.

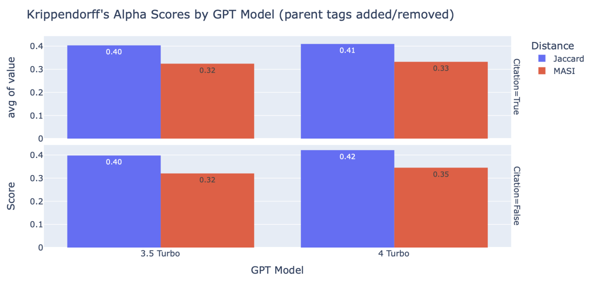


Figure 12: LLM Krippendorff's Alpha Scores by GPT Model

4.7.4 LLM Scores by GPT Model

The performance discrepancy between GPT 3.5 Turbo and GPT 4 Turbo was fairly minimal (see *Figure* 12). Evaluation across all tag preselection processes reveals slightly higher scores for GPT 4 Turbo. The poorer showing for the citation setup was surprising, but the gap is small. In preliminary experiments, citations sometimes caused models to make unsupported connections, such as tagging romance novels as *Beauty and Grooming*. Perhaps the added space for output text in the outgoing JSON occasionally confuses the model.

5 Scalability

Hearst's goal is to eventually implement this solution across more magazine/web properties, necessitating an affordable, efficient, scalable solution. The embedding models are incredibly cheap at $0.02/1M tokens. (ope, a) The most expensive component comes from GPT 4 Turbo, which costs $10/1M input tokens and $60/1M output tokens, although switching to 3.5 Turbo would lower these figures considerably to $0.50/1M input tokens and $1.50/1M output tokens—in practice, using GPT 4 Turbo to tag all 72 articles with preselection cost around $2. This is still very affordable at less than $300 for 10,000 articles.

Regarding efficiency, the embeddings processed 1k articles in about 8 minutes, while GPT 4 Turbo with tag-preselection often clocked in around 45 seconds during its best configurations (tagging 72 articles). Both models were parallelized using 90% of the OpenAI Tier 5 plan's maximum capacity. (ope, b) These speeds were satisfactory to Hearst.

Finally, scalability is also very strong regarding periodic taxonomy updates and adaptation to other marketplaces. Sentence-level embeddings only need to be generated once. Getting the embeddings for new tags, along with their cosine similarity scores in relation to the sentences, is a much faster process. Similarly, recalculating z-scores is very efficient. Finally, only articles for which new taxonomy categories make it through the embedding tag-preselection process must be rerun on the LLM. Furthermore, these additional runs will only take the new tags as inputs, thus reducing the output generation costs. Overall, the entire process is affordable, efficient, and easily adaptable.

6 Next Steps

One of the more interesting ideas that emerged later in the project timeline was making two separate calls to the LLM—the first to return a list of all items in the article and the second to match them to a subset of the taxonomy. This setup produced promising results in limited tests, but further evaluation is necessary to determine if it's a superior solution. One potential downside could be lost context since the second call only sees the first output list, not the original article.

Additionally, experimenting with slightly smaller embedding context windows could prevent products from being lost in long sentences. For example, one article read: "There's a lot going wrong for Ken in this episode, but one thing going absolutely right is the $25,000 gold pendant he pairs with a plain Maison Margiela T-shirt and Jacques Marie Mage sunglasses (the pendant also made an appearance in 'Too Much Birthday')." The t-shirt is buried within mentions of other clothing items and broader statements. Capturing all items lumped into this single sentence would prove difficult under the current sentence-level implementation. However, a model that breaks up long sentences with slightly overlapping, rolling windows may produce better tags.

7 Conclusion

Overall, the goal of scalably reproducing human-level tagging performance was achieved using embeddings in conjunction with LLMs. The former narrowed the scope of tags fed to the latter, which showed impressive performance as measured by Krippendorff's Alpha. This flexible, cost-effective approach provides a solid entry point for Hearst to grow additional marketplaces using AI to map relevant ads to articles.

References

a. OpenAI Pricing.

b. OpenAI Tiers.

Karimollah Hajian-Tilaki. 2013. Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian Journal of Internal Medicine*, 4(2):627–635.

Klaus Krippendorff. Computing Krippendorff's Alpha-Reliability.

Şeref Kerem Çorbacıoğlu and Gökhan Aksel. 2023. Receiver operating characteristic curve analysis in diagnostic accuracy studies: A guide to interpreting the area under the curve value. *Turkish Journal of Emergency Medicine*, 23(4):195–198.