Predicting Tags to Simplify Product Architecture

Emma Tebbe and EJ Haselden

Abstract

In this paper, we approach the problem of simplifying file organization architecture in a novel way by utilizing tag data from the Digital Asset Management tool Brandfolder. We simplify tags, a text-based way of labeling files, by providing tag suggestions in three unique ways: a Gensim Word2Vec model, a trigram model, and a pre-trained BERT model. We found that the Word2Vec model performed best at an average 74% accuracy, and used that model in a functional implementation. Our functional implementation returned at least one valid suggestion for up to 88% of assets.

Introduction

Brandfolder, a Digital Asset Management tool, allows marketing professionals to store, organize, and disseminate materials that are vital to a company's brand. These assets might be anything from a JPG logo to raw video footage for a commercial to a PowerPoint template. It is similarly vital that these assets are easy to find: if not, Brandfolder is no better than DropBox. Brandfolder offers several architectures to organize assets, one of which is tags.

Tags are short strings that are directly attached to each of the aforementioned assets. They are added by users via open text entry and generally consist of single words or short phrases, such as "event" or "spring marketing campaign." Because tags are user-generated and do not conform to any set template, the corpus of tags can expand quite quickly to include many slight variations of essentially the same tag. There is currently no strategy implemented to reduce this tag sprawl, which results in a large number of tags that are each associated with just a single asset. A single-instance tag adds no value; it is merely a redundant second identifier (after the asset ID) that provides no association with other conceptually related assets.

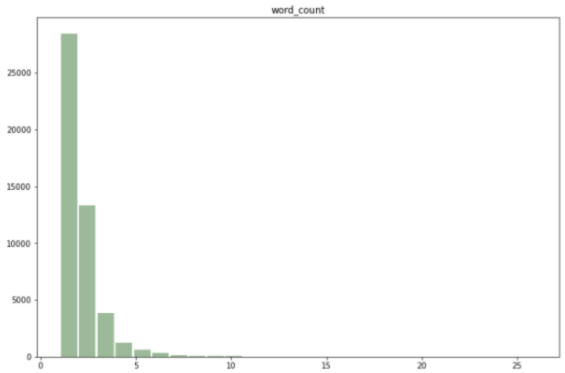

Figure 1: Count of tags by word count

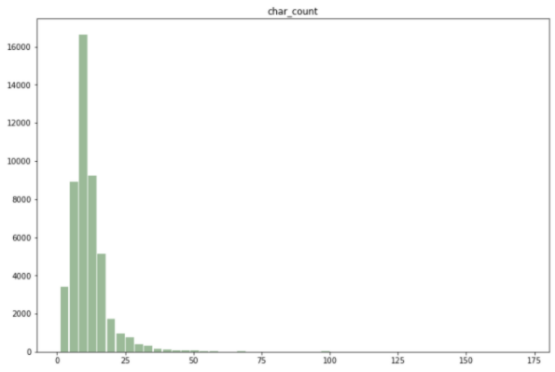

Figure 2: Count of tags by character count

This project uses data from the 11.5 million tags contained in the Brandfolder architecture. Across these tags, there are 48,230 unique tag texts. Tags in this dataset are text up to 26 words and 172 characters. The median tag is 1 word and 10 characters long. The figures below show the distribution of tags by word count (Figure 3) and character count (Figure 4).

Of the 48,230 unique tags, 14,779 are associated with only one asset. This constitutes 31% of all unique tag texts, and represents a significant opportunity for tag consolidation. Figure 3 shows the distribution of the number of assets per unique tag, limited to tags that are applied to fewer than 100 assets. The most prolific tag is applied to 433,279 assets ("Arts, Culture, and Entertainment"), while the mean is 239 and the median is 2.

The number of tags per asset is also generally low. The maximum number of tags per asset is 962, but the mean is 15, and the median is 13. Figure 4 shows the number of assets bucketed by the number of tags applied to those assets (limited to assets with fewer than 100 tags). Both this distribution and the distribution of number of assets per tag clearly illustrate the need for a method of simplifying the current tag architecture.

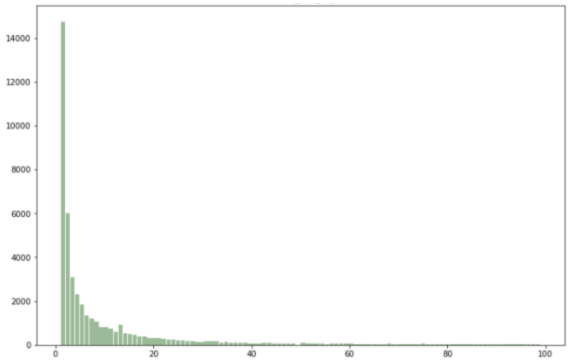

Figure 3: Count of tags bucketed by number of assets with that tag. Limited to tags with less than 100 assets per tag.

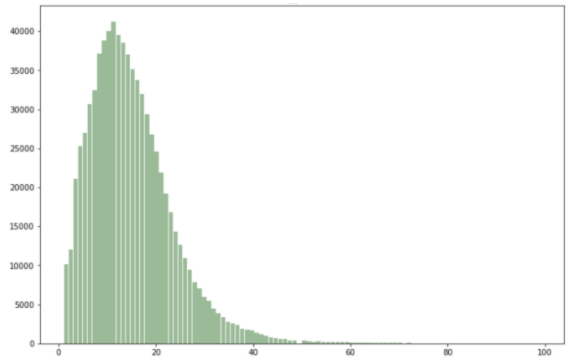

Figure 4: Number of assets bucketed by the number of tags associated with each asset. Limited to assets with fewer than 100 tags.

Our model is built to reduce the number of tags associated with a single asset. We leverage a Word2Vec model, a trigram model, and a BERT model to develop an effective way of suggesting alternative and additional tags for an asset based on the tags already associated with that asset. We envision these suggestions 1) being used by a moderator to prune the tag corpus and/or 2) being presented to users as prompts in the tag-entry interface, in hopes that they would choose an existing common tag over a rare bespoke variant.

This paper uses key terms that are defined as follows:

- **tag:** a single descriptive string applied to one or more assets
- **tag list:** the set of all tags applied to a single asset
- **rare tag:** a tag that is applied to only one asset
- **common tag:** a tag that is applied to two or more assets

Background

We believe our approach to this particular problem to be novel, and as such we found little evidence or literature indicating that others had used similar tools and strategies to the same end. We did see significant parallels, however, when comparing our efforts to the development of search query prediction models. Most notably, predicting search queries is dependent on the corpus of searches that users have already performed, much in the same way that predicting tags is dependent on the corpus of tags that users have already associated with given assets. In

both cases, the goal is effectively to reproduce a user's summary of a particular concept or artifact (making tag prediction a sort of inverse search query) .

Tags and search queries are also similar in number of words and characters, so we hoped that using simple n-gram models may prove as effective for tag suggestions as others had found them for predicting queries.[1] We were somewhat less optimistic about using pre-trained BERT, although it was shown to be a powerful tool for query prediction under certain circumstances.[2] We were skeptical because while query predictions are structured and order-dependent, tag lists are order-agnostic, thereby reducing the usefulness of BERT's attention mechanism. Tag suggestion also differs from search query suggestion due to a lack of rephrasing behaviors.[3] Ideally, tags each represent an entirely unique concept, and are most efficient when there is less conceptual overlap.

However, although there are major differences in problem statements, one can understand tag suggestion and search query suggestion as two sides of a coin: if tags are well-assigned then the searches performed within a Brandfolder will retrieve more effective and relevant results.

Methodology

In order to simplify the tag architecture, we pre-processed each unique tag text. This pre-processing involved lowercasing, stripping, removing punctuation, lemmatizing, removing stop words, and converting all digits to a uniform token. Tag simplification accomplished by any of this simple text processing will be more efficiently handled by the Brandfolder product code; this model will simplify tags based on their deeper meanings.

After processing, "rare" tags were labeled. 31% of all tags are associated with only one asset, so we used this as the criteria for tag rarity. Unique tag texts were then used to train a word2vec model. We used the entire texts instead of tokenizing phrases since the tag texts as a unit are what matters most, rather than the individual words composing the tag texts. Additionally, when using the model to return similar values, we want the word2vec model to return a text that already exists in the corpus of tags, instead of returning a single word that is contained within another tag. If the model were to return a single word that is contained within another tag but is not an existing tag itself, the model would not be contributing to simplifying the tag architecture.

We then structured the data such that each asset is associated with an array of 50 values containing the lemmatized values of the tags on that asset, and "None" for any empty values. Only 0.3% of assets have more than 50 tags, so while we did lose some information on those few assets, we gained efficiency compared to the 962-length array needed to contain all tags for all assets.

Functional Suggestion Test

We identified the need for a functional test that would accurately evaluate any type of model with respect to our specific task, namely the production of valid tag suggestions. We quickly

[1] https://arxiv.org/pdf/1507.02221.pdf
[2] http://ceur-ws.org/Vol-2621/CIRCLE20_06.pdf
[3] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.9419&rep=rep1&type=pdf

realized that standard metrics for model performance were poor predictors of suggestion quality. The W2V model, for instance, showed increasing loss with each epoch while the number of valid predictions steadily improved.

We therefore employed a test function that analyzed each model's set of suggestions given our existing tag corpus. For each asset, we selected a single tag at random from its list of applied tags. We then prompted each model to propose suggestions based on that tag alone. If any of those suggestions exactly matched any of the other tags applied to the asset, we consider the suggestion to be valid (considering it was at some point in time applied to the asset by a human user). See Figure 5.

We then scored each model based on the number of assets for which it proposed a valid suggestion, divided by the total number of assets tested. We did not weight our results based on multiple valid suggestions per asset; each is either pass or fail.

The method by which we extracted suggestions varied by model, but the input format to the Suggestion Test was identical across all models, ensuring parity in our model-to-model comparison. We also set a global parameter for the number of suggestions per test, so that was also held constant across models.

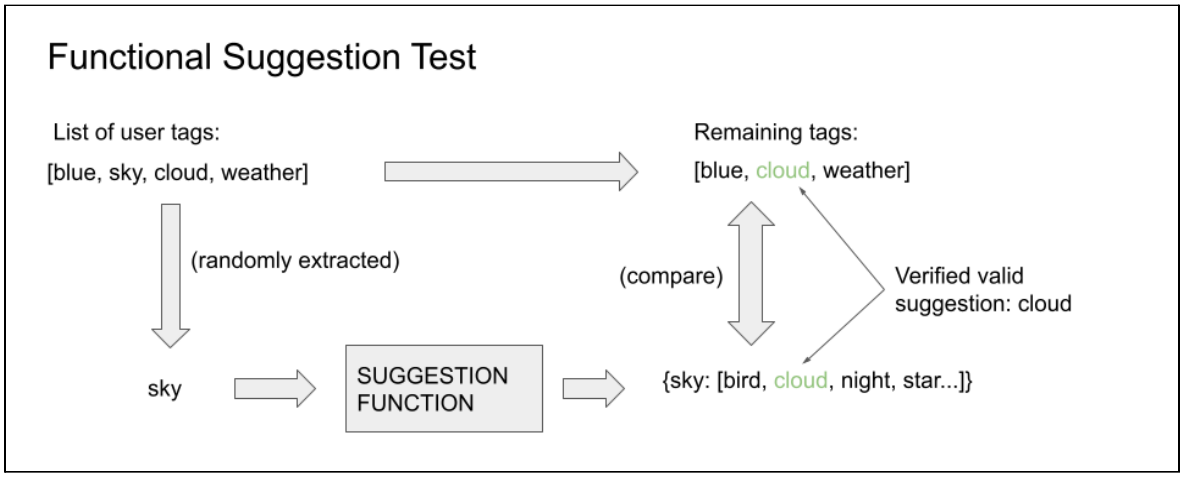

Figure 5: Functional Suggestion Test

Models

With preprocessing complete, we first implemented a simple Gensim Word2Vec model trained on the corpus of tags, using entire tags as words. For evaluation, we used the model's built-in most_similar function to return a set of strings most closely related to each test tag. We tuned this model (epochs, window, vector size) to optimize its performance on the Functional Suggestion Test.

Next, we trained a trigram model on a sample of the assets in the full dataset, treating the tag list for each asset as a "sentence". We retained the "<s>" and "</s>" markers, even though the sets of words are not true sentences, to ensure that tags from one asset did not bleed into the next. As with the W2V model, we trained the trigram model on the processed training data and used it to predict a set of tags based on each tag applied to an asset in the test data.

Finally, we used a pre-trained BERT model to generate potential suggestions. As with the trigram model, we are creatively defining “sentences” here to repurpose the model’s output for our task. To leverage BERT’s pre-existing knowledge of word meaning, we used a modified masked-word test by prompting the model with a two-word string consisting only of the tag to be tested and a [MASK] token. The model would then attempt to “complete” the “sentence” by returning a single string.

We then evaluated each model’s performance using the Functional Suggestion Test.

Results and Discussion

We tested all models on a subset of 1000 assets to determine average performance. We initially tested on larger subsets, but found no significant impact on the validity rate for any model between 1000 and 10000 test assets. We found that model performance varied consistently with the number of suggestions generated for each test, so we tested each model on a range of suggestion sizes. All three models performed best when they provided 35 suggestions per test tag.

For the Word2Vec model, we obtained an average of 79% accuracy across all suggestion sizes. However, since this is a human-generated dataset, it is conceivable that more than 74% of the suggested tags from this model are relevant to the assets they are suggested to.

With our trigram model, obtained an average 40% accuracy. The difference in accuracy between the Word2Vec model and the trigram model may suggest that tags are assigned to assets based on their meaning more than the context of other tags.

The BERT model yielded perhaps the most surprising results. Given that we provided minimal context for each test, that our corpus was order-agnostic, that many of our tags comprised esoteric terms and term groupings, and that the model we used was not trained on our corpus in any way, we expected very poor performance. In spite of those factors, however, the pre-trained BERT model obtained an average accuracy of 58%, which was mid-range for our experiment.

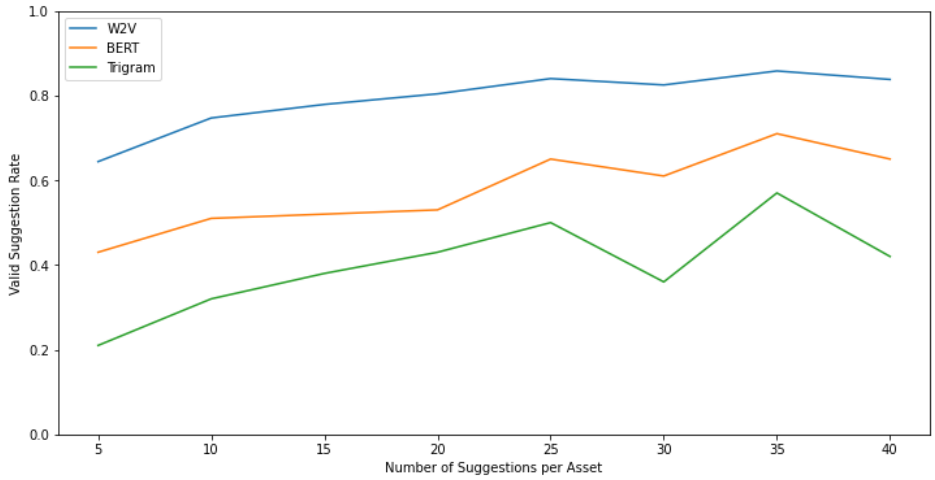

Figure 6. Valid Suggestion Rate by Model and Number of Suggestions

Implementation Example

In order to implement our model to provide a list of valid suggestions for a given set of tagged assets, we removed all rare tags from the corpus and retrieved suggestions from the best-performing Word2Vec model for each asset. This returned a large set of tags, including duplicates. We then removed all rare tags from this set of suggested tags and counted the number of times each unique suggested tag appeared in the set of suggestions. This allowed us to rank suggested tags based on a weight of how often they were suggested. Ultimately, our suggestion function returned the list of original tags, the list of suggested tags, the duplicates found in the suggested list that were in the list of original tags, the list of new suggestions, and the list of new suggestions ranked by weight. While this does not take one tag and return suggestions to replace that single value, it is potentially more powerful, as it takes into account the context of all of the tags already associated with that asset.

The following is an example of a suggestion return:

```
ORIGINAL TAG LIST:  ['film industry' 'arts culture and entertainment' 'Choreography' 'Dance'
 'celebrities' '775621438' 'awards ceremony' 'BFselects_FTP'
 'Entertainment' 'Fashion Design' 'Performing Arts']

SUGGESTIONS:  ['fashion', 'music', 'entertainment', 'purple', 'event', 'film industry', 'magenta',
'blue', 'red', 'pink', 'fashion', 'film industry', 'music', 'entertainment', 'blue', 'red', 'purple',
'fashion', 'music', 'entertainment', 'purple', 'event', 'film industry', 'magenta', 'blue', 'red',
'pink', 'pink', 'purple', 'event', 'red', 'magenta', 'entertainment', 'fashion', 'fashion', 'music',
'entertainment', 'purple', 'event', 'film industry', 'magenta', 'blue', 'red', 'pink', 'pink', 'event',
'purple', 'entertainment', 'red', 'magenta', 'fashion', 'music', 'entertainment', 'purple', 'event',
'film industry', 'magenta', 'blue', 'red', 'pink', 'fashion', 'music', 'entertainment', 'event',
'purple', 'film industry', 'magenta', 'red', 'blue', 'fashion', 'music', 'entertainment', 'purple',
'event', 'film industry', 'magenta', 'blue', 'red', 'pink', 'film industry', 'red', 'fashion', 'blue',
'pink', 'fashion', 'music', 'entertainment', 'purple', 'event', 'film industry', 'magenta', 'blue',
'red', 'pink', 'fashion', 'music', 'entertainment', 'purple', 'event', 'magenta', 'film industry',
'musician', 'concert', 'fashion', 'music', 'entertainment', 'purple', 'event', 'film industry',
'magenta', 'blue', 'red', 'pink', 'color image', 'topix', 'bestof', 'horizontal', 'vertical', 'blazer',
'film industry', 'fashion', 'music', 'entertainment', 'purple', 'event', 'film industry', 'magenta',
'blue', 'red', 'pink', 'fashion', 'sleeve', 'fashion', 'music', 'entertainment', 'purple', 'event',
'film industry', 'magenta', 'blue', 'red', 'pink', 'entertainment', 'musician', 'music', 'singing',
'concert', 'purple', 'event', 'magenta']

DUPLICATES:  ['entertainment', 'film industry']

LEGITIMATE SUGGESTIONS:    ['fashion', 'music', 'purple', 'event', 'magenta', 'blue', 'red', 'pink',
'fashion', 'music', 'blue', 'red', 'purple', 'fashion', 'music', 'purple', 'event', 'magenta', 'blue',
'red', 'pink', 'pink', 'purple', 'event', 'red', 'magenta', 'fashion', 'fashion', 'music', 'purple',
'event', 'magenta', 'blue', 'red', 'pink', 'pink', 'event', 'purple', 'red', 'magenta', 'fashion',
'music', 'purple', 'event', 'magenta', 'blue', 'red', 'pink', 'fashion', 'music', 'event', 'purple',
'magenta', 'red', 'blue', 'fashion', 'music', 'purple', 'event', 'magenta', 'blue', 'red', 'pink',
'red', 'fashion', 'blue', 'pink', 'fashion', 'music', 'purple', 'event', 'magenta', 'blue', 'red',
'pink', 'fashion', 'music', 'purple', 'event', 'magenta', 'musician', 'concert', 'fashion', 'music',
'purple', 'event', 'magenta', 'blue', 'red', 'pink', 'color image', 'topix', 'bestof', 'horizontal',
'vertical', 'blazer', 'fashion', 'music', 'purple', 'event', 'magenta', 'blue', 'red', 'pink',
'fashion', 'sleeve', 'fashion', 'music', 'purple', 'event', 'magenta', 'blue', 'red', 'pink',
'musician', 'music', 'singing', 'concert', 'purple', 'event', 'magenta']

BY WEIGHT:    [('fashion', 15), ('purple', 15), ('event', 14), ('magenta', 14), ('red', 14), ('music',
13), ('blue', 12), ('pink', 12), ('musician', 2), ('concert', 2), ('color image', 1), ('topix', 1),
('bestof', 1), ('horizontal', 1), ('vertical', 1), ('blazer', 1), ('sleeve', 1), ('singing', 1)]
```

Conclusion

In this paper, while attempting to identify the optimal model type for a tag-consolidation task, we discovered that the simplicity of Word2Vec embeddings represented the most efficient tool for our lightweight suggestion framework. Given the surprising performance of the BERT model that was not trained on our corpus at all, however, we believe that it could outperform the Word2Vec model in a more fully realized implementation. Further work would undoubtedly focus on fine-tuning the BERT model on some permutation of our existing corpus to determine the maximum potential of that model in this context.