Clothing Fit Prediction Based on Text Features

Karl Wang
UCSD
kawang@ucsd.edu

Chenfeng Wu
UCSD
chw357@ucsd.edu

Abstract

Rent The Runway is a popular clothing rental service where people can rent clothes for different occasions. In this project, we are going to use user reviews to predict size feedback of whether the rental cloth is fit, large, or small.

1. Dataset Analysis

We chose to use Rent The Runway dataset, which consist of user reviews and size feedback of the rented item.

1.1. Dataset Imbalance

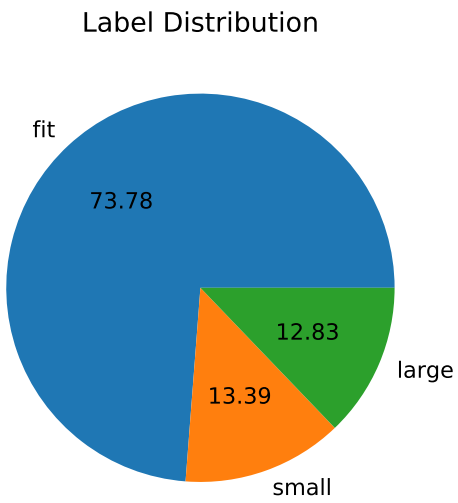

Figure 1. Percentage of different labels in the dataset

Out of 192,544 reviews, 73% of them have the label "fit". Most customers feel the rental dress is true to size. Therefor we need to perform re-sampling on the "fit" category to make sure each classes are similar in sizes during training. Otherwise, classifiers will just predict "fit" all the time.

1.2. Temporal Features

Now let's examine how user feedback changes over time to see if temporal features can help our prediction.

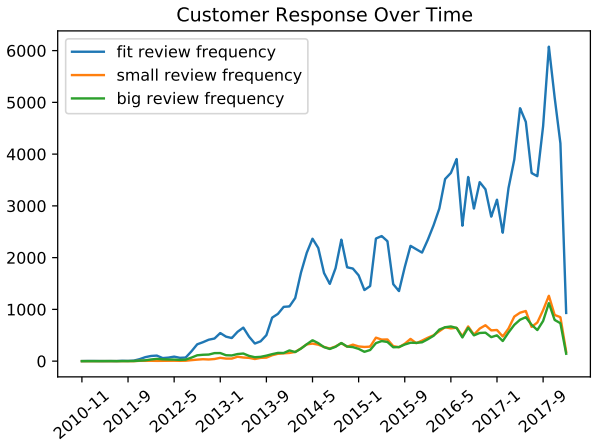

Figure 2. Frequencies of various feedback over time

Customers who felt their rentals are "small" or "large" increase modestly overtime. In contrast, numbers of customers who responded "fit" grows scientifically. Therefore, temporal features are less helpful for predicting "small" and "large" labels, since their trends are identical over time. This can be attribute to increase of traffic on the website. Since most customers feel their rentals are "fit", the number of fit responses are going to increase naturally.

1.3. Ratings as Feature

If there is a correlation between product ratings and size feedback, then we can use rating as a feature for our prediction. For instance, customers might tend to give higher ratings when the cloth is fit.

As we can see from from **Figure 3**, 90% of customers who feel their rental cloth is undersized of oversized gave the item 8 or 10 stars. Thus, star rating, although a good indicator of user experience, is not a good feature for predicting size feedback.

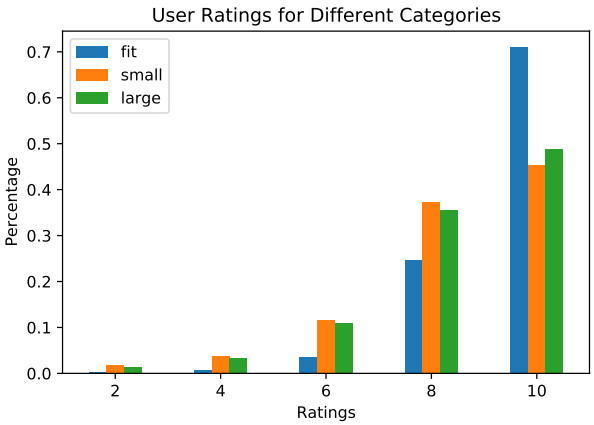

Figure 3. User ratings for various size feedback

1.4. Text Features

When users describe size feedback, the adjectives used in the sentence are often very informative of whether the cloth is fit, small, or large. Take the following example:

User feedback: small

> Lovely fabric, heavier than I imagined it would be (somehow I thought billowy, but its softer and stiffer than that) but it runs small around the rib cage. Pretty! Runs small.

The user directly use "small" in the sentence, indicating the dress is in a bit smaller in size. Although the phrase "small", "large", or "fit" doesn't always appear in the sentence, other adjective associated with fitting, such as "comfortable", "snug", "loose" can be used as features for our prediction.

Another observation is stop words do not really contribute to sentiment on cloth fitting. **Figure 4** shows that removing stop words can help to produce n-grams that have more useful meaning. However, we should keep the stop words that negate the meaning of the sentence. For instance, the meaning of "this dress does not fit" would change dramatically if we remove the word "not".

Bigram	
Keep Stop words	Remove Stop words
(a, little)	(true, size)
(so, i)	(runs, small)
(the, top)	(little, tight)
(so, many)	(runs, large)
(able, to)	(great, dress)

Figure 4. Bigram: Before and after removing stop words

2. Prediction Task

The task is to use user reviews to predict if user feels the rental dress fits perfectly, is smaller, or is larger. Since labels are independent from each other, we cast the problem as a classification problem. Given text features, classify as class 0 (small), class 1 (fit) or class 2 (large).

2.1. Evaluation Metric

We decide to use balanced error rate $\frac{1}{3}\sum_{classes}(1 -$ true positive rate$)$.

2.2. Baseline

The baseline model we choose to use is logistic regression as classifier and uni-gram (adjective only) as features.

2.3. Model Evaluation

We implemented train, validation, test pipeline. 20% of data are used for validation and parameter tuning and 20% of data are used for testing. We select the models that have best validation performance to run on the test set.

2.4. Feature Engineering

We only use text features for the predictive task, as we found text features contain the most relevant information. Then we prepare training features using the following steps:

1. Remove stop words except the stop words that negate meaning such as "not","wouldn't",etc.

2. Only include n-grams that contain at least one adjective in the bag of words. We decide to use bag of words (BoW) method for word embedding because we think the most popular words can define text document feature.

3. Use multiple n-gram strategy:

- uni-gram
- uni-gram + bi-gram
- uni-gram + bi-gram + tri-gram
- uni-gram + bi-gram + tri-gram + 4-gram

The reason for using multiple n-gram strategy is we think this approach can capture more context, making features highly contextual.

4. Encode word vectors using TF-IDF score to help capture relevance of words in review texts.

2.5. Model Selection

We chose models that are suitable for classification, such as KNN, Naive Base, Logistics, SVM. These kinds of models produce discrete value output and each output label belongs to a class. Regression model will not be suitable for this task because the notion of small, fit, large is on an arbitrary scale and labels cannot be represented as continuous

values. If the mapping small→0, fit →1, large →2, regression model (using MSE loss) would make the assumption that predicting a fit review small is the same as predicting it as large, but these two are different classes.

3. Model

3.1. Proposed Feature Representation

We preprocessed the corpus to remove all stop words. Then our model takes in vectors of TF-IDF of n-gram that contains at least one adjective word as features. We used combination of the top uni-gram, bi-gram, tri-gram and four-gram as the dictionary. For each n-gram, we separated the reviews of fit, small, and large, then we take the union of the top fit n-gram, top small n-gram, and top large n-gram. After calculating the inverse document frequency, we acquired the TF-IDF for each word in each document based on if the word is one of the words in the dictionary.

Algorithm 1: Popular word set using union

```
popular word set = {} ;
for n-gram=1, 2, 3, 4 do
    a := top fit n-gram[:top k] ;
    b := top small n-gram[:top k] ;
    c := top large n-gram[:top k] ;
    top n-gram = a ∪ b ∪ c;
    popular word set = popular word set ∪ top
      n-gram;
end
```

3.2. Alternative Feature Representations

Aside from TF-IDF, we tried bag of words counting (term frequency). We also tried preserving the stop words instead of filtering them. Finally, as top fit n-gram, top small n-gram and top large n-gram contains many mutual n-grams. We tried to use symmetric difference, which only preserve the n-grams that are unique among all labels. We thought that this would make each label's features more distinctive from each other.

In total, we have 8 feature representations to compare: term frequency vs TF-IDF, remove stop words vs preserve stop words, and symmetric difference vs union.

3.3. Proposed Model

We used multinomial naive bayes as our model to conduct text classification. Naive bayes assumes features are mutually independent to each other. In our case, this means that our model assumes that position of each word does not matter. This assumption mostly works in our case as we only preserve n-grams that contain at least one adjectives.

Algorithm 2: Popular word set using symmetric difference

```
popular word set = {} ;
for n-gram=1, 2, 3, 4 do
    a := top fit n-gram[:top k] ;
    b := top small n-gram[:top k] ;
    c := top large n-gram[:top k] ;
    top n-gram = (a ∪ b ∪ c) - (a ∩ b ∩ c) ;
    popular word set = popular word set ∪ top
      n-gram;
end
```

Below is the equation of naive bayes:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\text{Loss} = \sum_{j=1}^{N} 1 - \delta(y_j - \hat{y}_j)$$

where y represents the label, x represents the features, and n represents the number of features. $P(y)$ is the prior of the label, $P(x_i \mid y)$ is the conditional distribution of feature x_i given the label y (in our case, we use multinomial distribution). Naive bayes classifies the data by finding the label with the maximum posterior probability $P(y \mid x_1, \ldots, x_n)$ given the features. To optimized our classifier, we tried to minimize the loss function, which indicates the total number of incorrect classifications.

3.4. Alternative Models

Besides naive bayes, we also tried SVM with linear kernel, logistic regression, decision tree, and K nearest neighbor as alternative classification models. Because we used combination of uni-gram to four-gram, the number of features is around 6000. Although this was not a problem for naive bayes model, it was a scalability issue for the alternative models. To solve this problem, we decide to do dimensionality reduction. Because the training data is a large sparse matrix, we used singular value decomposition to preserve the top 100 principle components. The reduced training data significantly speeds up the training time for the alternative models.

3.5. Strength and Weakness

Naive bayes is highly efficient and intuitive. It is easy to implement and scale well with large dataset. The downside is that naive bayes assumption about mutually independence of features hardly hold true. However, even when the assumption does not hold true, naive bayes still work quite well in practice.

Logistic regression works well when the decision boundary is linearly separable. It can be regularized to avoid over fitting. Nevertheless, it does not work well when the decision boundary is non-linear.

SVM can use non-linear kernels to model non-linear decision boundary. However, using non-linear kernels is very computational intensive. Therefore, beside linear kernel, SVM with non-linear kernels do not scale well with big data.

Decision tree uses hierarchical structure to classify features, and the structure provides a non-linear decision boundary. The downside is decision tree are very prone to over fitting.

K nearest neighbor is able to model non-linear decision boundary, highly intuitive and easy to regularize using number of neighbors. However, it does not perform well on high dimensional data, and it is the most computational intensive model among all the models here.

4. Literature Reviews

4.1. Dataset Survey

The Rent The Runway dataset we used came from Prof. McAuley's repository of public dataset, which is most likely obtained via web scraping. The original purpose of the dataset is to study how to build a personalized size recommender system for online shopping [5]. Due to sizing variation from different brands, customers are often frustrated when they find out the item they just purchased/rent is not a comfortable and appropriate fit. Therefore, researchers try to devise an algorithm that can automatically provide accurate and personalized fit guidance to ensure customers can find perfect fit every time. And this Rent The Runway dataset was originally used for this purpose.

Other reviews datasets, such as amazon review [3], are being used to study sentiment analysis and opinion mining. The main objective is to take a piece of review text to predict whether it's a positive review or a negative review [1]. Such works are mostly based on sentiment analysis as positive reviews often reflect approval and satisfaction, whereas negative reviews reflect unhappiness and dissatisfaction.

4.2. State of the Art

Rather than using Rent The Runway dataset to build a size recommender system, as Misra et al did [5], we used the dataset to build a language model to predict customers' feedback on size of the cloth. Thus we looked into the current state of the art research on language modeling and semantic analysis.

4.2.1 Bi-directional LSTM

Language understanding is contextual. The meaning of a word heavily depends on words that surround it. Nevertheless, traditional LSTM architecture only processes input sequence in one direction, which means only previous words are considered but not words that follow. To make the model fully contextual, researchers train two LSTM networks [1], one take input from left to right and another one in reverse direction. Such bi-directional networks out perform traditional network on next sentence prediction and reading comprehension based Q&A (i.e. SQuAD [7]).

4.2.2 BERT: Bi-directional Transformer

Similar to bi-directional LSTM, BERT [2] also employs bi-directional architecture to harvest the full power of left and right context. With the addition of attention mechanism in transformer, BERT is able to pay attention to key words in sentences, which helps to produce more accurate representations. As a general language representation model, BERT is trained on massive corpus (Wikipedia, books, etc) using masked words training. To perform any specific prediction task, one just need to fine tune BERT model on a labeled dataset related to the task.

4.2.3 ELMo: Context Based Word Embedding

Contrary to Word2Vec [4], which gives fixed word embedding using a pre-trained 'lookup' dictionary, ELMo [6] uses bi-directional LSTM to create dynamic embedding depending on the context. Take the example of the sentence "I read the book yesterday". If we used fixed word embedding, "read" would be interpreted as present tense. ELMo, on the other hand, takes in the entire sentence and would interpret "read" as past tense because based on the context, "yesterday" refer as the past.

4.3. Ours vs SOTA

Our feature representations and models differ vastly compare to those of state of the art methods. We use mainly popularity of words as features based on counting without taking the meaning of words into consideration. SOTA tends to use word embedding to differentiate each word and take the semantic meaning of each word as features. In fact, SOTA methods use neural network to acquire the word embedding, where as we simply use counting.

After acquiring the features, SOTA methods use another neural network to classify based on the features. We simply use classic, old fashioned classification algorithms, such as naive bayes, logistic regression, SVM and etc. Although SOTA methods are much more effective than ours, our model might be more efficient as it does not involve computational intensive neural networks.

5. Results

Figure 5. Validation BER of the models against different feature representations. Note: `sym diff` means symmetric difference.

5.1. Feature Representations

From **Figure 5**, we conclude that using union top n-gram categories, preserving stop words and using TF-IDF is the most effective feature representation, as it has the lowest balanced error rates across all models. It is also interesting to see that using TF-IDF vs TF makes little difference. Preserving stop words seems to improve the error rate comparing to removing stop words. Finally, using union of top n-gram categories seems to be more effective that taking the symmetric difference of top n-gram categories.

5.2. Model Performance

After choosing union, preserve stop words, TF-IDF as our feature representation and choosing the best parameters for applicable models according to validation BER, below is the testing BER.

models	BER
decision tree	0.427437
KNN	0.433382
logistic regression	0.368269
naive bayes	0.338752
SVM	0.367676
baseline	0.402164

The testing BER is similar to the validation BER. According to the validation and testing BER, naive bayes is the most effective model among all models, which means our proposed model is correct. Logistic regression and SVM provides very similar BER across all feature representations. It is interesting to note that logistic regression with combination of uni-gram to four-gram has a 9% lower BER than that of the baseline, which uses logistic regression with uni-gram.

5.3. Model Parameters

The main model parameters are class priors $P(y)$ and feature probability $P(x_i \mid y)$. The class priors are all around $\frac{1}{3}$, and that is to be expected because we sampled the data to be balanced. After sorting the feature probability vector for a given class, we discovered the largest value corresponds to n-grams that are most informative about that class. The following table consists the top n-grams that describe each class the most. From the table we observed the top n-grams are mostly uni-grams and bi-grams, and they make intuitive sense.

fit	small	large
('many', 'compliments'), 'many', ('was', 'perfect'), ('a', 'little'), ('true', 'to', 'size'), ('true', 'to'), 'little', 'true', 'comfortable', 'great'	'many', 'gorgeous', 'comfortable', ('runs', 'small'), ('so', 'i'), 'short', 'great', ('a', 'little'), 'little', 'small'	('too', 'big'), ('runs', 'large'), 'smaller', ('so', 'i'), 'comfortable', 'great', ('a', 'little'), 'little', 'big', 'large'

5.4. Conclusions

Below is a table that compares combination of uni-gram to n-gram using naive bayes, union of top n-grams from three categories, preserve stop words, and TF-IDF.

up to n-gram	BER
1	0.419936
2	0.353967
3	0.342363
4	0.338752

The result of combining multiple grams is clearly more effective than using just uni-gram, yet there is a diminishing return. The improvement from up to 3-gram to up to 4-gram is minuscule, but the number of features increases from around 2000 to around 6000.

Our proposed feature representation was incorrect, as preserving stop words is more effective than removing stop words across most models. We suspect this is because by removing stop words, tri-gram and four-gram contexts no longer makes sense. Hence, tri-grams and four-grams might be highly unique per review, thus not contributing toward distinctive features for each label.

The proposed model succeeded because the main assumption of hold true for naive bayes. Since we only consider adjective n-grams, features are mostly mutually independent. In addition, since naive bayes was highly efficient, we do not have to do dimensionality reduction to train the model (in fact, the reduced data, which contains negative

values, didn't work because naive bayes only accepts positive values). On the other hand, dimensionality reduction might actually hurt alternative models due to lost of information.

We cannot conclude that SVM and logistic regression failed as they followed immediately behind naive bayes in terms of BER. It might be that these two models are slightly worse due to dimensionality reduction. Decision tree and KNN, on the other hand, clearly failed as they do not even beat the baseline that used uni-gram and logistics regression.

In conclusion, our proposed feature representation was not as effective as we expected but the proposed model is fairly successful both in terms of BER and efficiency.

References

[1] Muhammad Abulaish, Jahiruddin, Mohammad Najmud Doja, and Tanvir Ahmad. Feature and opinion mining for customer review summarization. In Santanu Chaudhury, Sushmita Mitra, C. A. Murthy, P. S. Sastry, and Sankar K. Pal, editors, *Pattern Recognition and Machine Intelligence*, pages 219–224, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[5] Rishabh Misra, Mengting Wan, and Julian McAuley. Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 422–426, New York, NY, USA, 2018. Association for Computing Machinery.

[6] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.