Robust Learning from Unfunny Sources

Fred Owens [*1]

Algorithm 1 DPLL Pseudocode

Input: random people p_1, p_2, funniest word vectors w_1, w_2, training words t_1, t_2
Insure $p_1(w_2) = 0$ and $p_2(w_1) = 0$
Average vector v_1 based on t_1
Average vector v_2 based on t_2
if $0 < (w_1 - w_2) \cdot (t_1 - t_2)$ **then**
  Return True
else
  Return False
end if

Abstract

Modeling an individual's sense of humor is a difficult problem for computers, not only because humor itself is poorly understood by theoreticians, but also because doing so would undoubtedly take a large amount of data from the individual. But, what if an individual's sense of humor could be learned based off others' senses of humor? Inspired by the work of (Gultchin et al., 2019), which demonstrated that aspects of humor can be be captured by Word Embeddings, as well as (Konstantinov & Lampert, 2019), which demonstrated techniques for learning robustly from unreliable data sets, this paper explores techniques for learning senses of humor from other senses of humor. Essentially, the algorithm described herein seeks to learn more from individuals with a similar sense of humor to the target individual, while minimizing learning impacts from dissimilar individuals.

1. Motivation

Research on applying Natural Language Processing techniques to humor has been slow to progress, partly because humor itself is not well understood, and partly because the computational task is daunting. Intuitively, we recognize that individuals have different senses of humor, and that some things are funny for different reasons.What might be less intuitive, however, is that individual words themselves, devoid of context, can be funny. However, (Engelthaler & Hills, 2018) (henceforth referred to as EH) showed exactly this, by conducting a crowdsourced study on five thousand words, and asking participants which words they found humorous. The results showed that there are words that are consistently found funnier than other words, such as "wanker" and "boondoggle". However, the conclusions that can be drawn from EH are limited: while it provides a useful starting place, the data is aggregated, and there is no actual data for words *individuals* found funny.

(Gultchin et al., 2019) (henceforth referred to as GE) picks up where EH left off. They conducted a crowdsourcing study of their own, which focused more on identifying specific senses of humor, and various "features" of humorous words, such as whether a word possess a scatological connotation, or juxtaposes unexpected terms. GE then analyzed the data through the lens of Word Embeddings (WEs), machine learning tools which capture the properties of words via a large vector. They showed that WEs were able to capture various humor features of words, and were able to represent a mean "sense of humor" vector for an individual by using these tools.

Meanwhile, (Konstantinov & Lampert, 2019) (KL) provide a framework for learning robustly from "corrupted" sources. The algorithm they describe uses a small reference data set to ascribe weights to a wide variety of data sets, based on how similar they are to the reference data. The model then learns from *all* the data sets, based on the weights ascribed. This paper implements an attempt at applying KL's algorithm to humor. By using a small sample of words one individual did and did not find funny as a reference, can we leverage more data to effectively predict what words this individual *would* find funny?

2. Related Work

2.1. Gultchin et al. (GE)

As discussed previously, (Gultchin et al., 2019) attempted to use WEs to draw conclusions from words deemed funny by asking 1,678 subjects. First, they started by determining 216 funniest words from the EH data set, based off participant responses. Then utilizing these 216 words, each subject was shown 6 randomly-selected words at a time, and picked the one they deemed funniest of the six, until they had seen all

216. The process was repeated for the 36 words deemed "funny", then for the 6 after that picked "funnier," resulting in a word that the individual thinks is funniest.

Although this method effectively winnows down the field of 216 into just 1 "funniest" word for each subject in just 43 clicks, the result is a fairly noisy data set. And, though it is tempting to think of the 36 "funny" words as being uniformly funnier than the other 180, that is not the case. Rather, this result means only that *the given funny word is funnier than five other randomly selected words*. Similarly, in the context of a 64-team single-elimination basketball tournament, it would be a mistake to say that every team that advanced to the round of 8 is definitely better than the 56 teams that did not (never mind the fact that the team that wins a given game is not always the better team). The implications of this method of collecting data will be discussed more in Section 5

In addition, these participants were also asked about different humor features the words possess: sound, scatological connotation, colloquialism, sexual connotation, juxtaposition, and insulting nature. GE used K-means clustering to find groupings of words based on how similarly they exhibited these different features, and found statistically significant demographic differences between the individuals who found these words funny. For example, those who found words with sexual connotations (like "dong") funny skewed disproportionately male and young, and those who found funny-sounding words (like "gobbledegook") funny skewed female and older.

For the purposes of this paper, perhaps the most significant finding relates to their attempt to predict "individual differences in word humor." The prediction algorithm finds disjoint sets of humor taste, essentially humor "opposites," where one individual's funniest word was not rated funny by the other. Then, for each set, using each subject's other 35 funny words as training data, and the funniest words as the test data, the model attempts to predict which individual found the word funniest, and demonstrated a reasonable amount of success (78.1%, whereas random guessing should see a 50% success rate). See Algorithm 2 for psuedocode.

There are two important takeaways from this result:

- Surprisingly, 60% of pairs exhibited this "opposites" quality; in addition to the demographic differences found by accident in the K-means clustering for humor features, this result further reinforces the heuristic that senses of humor among individuals are quite distinct.
- WEs are a word model for humor prediction: given a relatively small amount of funny words, a fairly simple training algorithm is able to learn a lot from them, and make accurate predictions.

Algorithm 2 GE Prediction

Input: random people p_1, p_2, funniest word vectors w_1, w_2, training words t_1, t_2
Insure $p_1(w_2) = 0$ and $p_2(w_1) = 0$
Average vector v_1 based on t_1
Average vector v_2 based on t_2
if $0 < (w_1 - w_2) \cdot (t_1 - t_2)$ **then**
  Return True
else
  Return False
end if

While the success of the model is undoubtedly noteworthy, one cannot help but wonder if more interesting humor predictions can be made based off this data set. How successfully could a model predict humor ratings for words besides *the funniest* one? Is it hard to predict words that are not funny? Could a model instead utilize the similar senses of humor to make predictions? These questions are explored in Section 3.

2.2. Konstantinov and Lampert (KL)

One of the fundamental problems with data science, machine learning, and AI, is the problem of data acquisition. Obtaining quality data is expensive and time consuming, as it generally requires a human. But without this curation, the model either will not have enough data to give meaningful results, or it could be susceptible to the influence of unreliable data. But what if the machine did the curation for you, by filtering out data that a human would deem irrelevant? (Konstantinov & Lampert, 2019) attempts to thread this needle in their algorithm.

Fundamentally, the KL algorithm attempts to minimize a loss function $\mathcal{L}$, just like many other machine learning algorithms. However, there are many features that differentiate this algorithm, and it is worth discussing them before beginning a critique.

The algorithm is based off a reference data set $\mathcal{S}_\mathcal{R}$, composed of m corresponding inputs $\mathcal{X}$ and outputs $\mathcal{Y}$. To train the model, N different data sets $\mathcal{S}_i$ are provided, of questionable validity. Let the number m_i denote the number of input/output pairs for each $\mathcal{S}_i$. The goal is to minimize a predictor h, which can be compared to the actual outputs of the test data set in order to see how well the model performs.

As for actual computations, KL determines weights α based on how similar they are to the reference data set. Weights are distributed such that $\sum_{i=1}^{N} \alpha_i = 1$ and $\alpha_i \geq 0$ for all i. Intuitively, the weights α must be greater than 0 and must add up to 1, in order to weight the data sets properly, similar to how in a probability density function, the individual probabilities for the entire sample space must add up to 1, and

no individual probabilities can be negative. But how do you measure the difference between the reference data set $\mathcal{S}_\mathcal{R}$, and weight them accordingly? KL defines the discrepancy between reference data set $\mathcal{S}_\mathcal{R}$ and a training data set $\mathcal{S}_i$ as:

$$d(\mathcal{S}_i, \mathcal{S}_\mathcal{T}) = \sup_{h \in \mathcal{H}} (\mathcal{E}_i(h) - \mathcal{E}_\mathcal{T}(h)) \tag{1}$$

Where $\mathcal{E}(h)$ is the expected risk of a predictor h in a hypothesis class $\mathcal{H}$, which is made up of possible predictor values. In plainer language, if all functions give similar outputs on two data sets, the data sets are similar.

Using these discrepancies for the N data sets, the weights α are chosen using:

$$\min_\alpha \sum_{i=1}^{N} \alpha_i d_\mathcal{H}(\mathcal{S}_i, \mathcal{S}_\mathcal{T}) + \lambda \sqrt{\sum_{i=1}^{N} \frac{\alpha_i^2}{m_i}} \tag{2}$$

such that: $\sum_{i=1}^{N} \alpha_i = 1$ and $\alpha_i \geq 0$ for all i

where λ is a hyperparameter "selected by cross-validation on the reference dataset" (Konstantinov & Lampert, 2019). λ represents the tension between the fundamnetal problem being solved by KL: how to encourage learning from multiple sources. In their GitHub code, KL randomly generate several λ values, run shorter optimizations for all these values, then select the best-performing one and run a longer optimization for it. Essentially, a λ value of 0 indicates that *only* the data set with the lowest discrepancy is utilized (normally the reference data set itself, as it is useful to use it in training). However, as λ approaches ∞, the algorithm simply equally weights all data sets, based on how many samples they have. This corresponds to just trusting all the data given to the algorithm.

The most relevant results from KL concern an experiment they conducted using Amazon reviews of various products, from (McAuley et al., 2015), wherein each review is classified as either positive or negative. A book was then chosen to be the reference, with 100 randomly selected reviews to act as the reference data set $\mathcal{S}_\mathcal{R}$, with another 500 randomly selected reviews to act as the test data set. Reviews from other products were used as the training data, some of which were books, while others were unreliable (non-books). The inputs to the algorithm were represented by WEs and sentence embeddings, and the outputs were a binary label: positive or negative sentiment in the review.

What KL found was that their algorithm out-performed numerous other methods in predicting whether review sentiments for books will end up as positive or negative, including blindly trusting all data, and only using the reference data. However, the prediction accuracy for the KL algorithm converges to the performance of the reference data-trained model as the percentage of "corrupted" (non-book) samples increases. In other words, predictions get worse as data gets worse. This result makes sense: if the non-book reviews are taken with a healthy grain of salt, they undoubtedly contain some broad trends in sentence construction and word choice in general that can be extrapolated in order to use in predicting a specific type of review.

KL performs a similar experiment on identifying attributes in pictures of animals (i.e. if an animal is black). The model demonstrated a similar level of success: significantly out-performing other methods of prediction, but eventually converging to perform equally well as a model trained *only* on the reference data as the share of corrupted sources increases (in this case, the inputs were corrupted by deliberately manipulating the images, such as by blurring them or creating dead pixels).

There are a two important takeaways from KL:

- Too much garbage data can still be a problem, but can be mitigated with this algorithm. This is illustrated by the convergence between the KL results and the reference-only results.
- Predictions are very noisy, even with good models. In the Amazon example, the KL algorithm has an average classification error of 0.275 even with 0 corrupted sources.

One question begged by these experimental results is how well this algorithm can perform when the output spaces are not binary. Does the performance of the algorithm improve or get worse? Overall though, KL represents an important step in trying to decrease the importance in human labor in creating good models, and is utilized extensivley in this paper's implementation.

2.3. Kahng et al. (KA)

While the subject matter is interesting, (Kahng et al., 2019) is not quite as relevant to this paper's implementation (see Section 4) as the other two papers, and so less time will be spent discussing this paper. This research attempts to model a "virtual democracy," wherein voters are modeled according to a Mallows Model (MALLOWS, 1957) in the context of a Borda Count voting system (essentially a variation of ranked choice voting). The experiment seeks to generate a noisy model for individual preferences, as voters often have idiosyncratic preferences in democratic processes, but demonstrate that Borda Count provides a good model for accounting for this noise.

In KA, they represent voters as having a true preference on a decision as σ_i*, and predict this voter as having a preference σ_i, which is based on a Mallows distribution from σ_i*. However, this "true" preference is obviously unknowable, and so the paper goes to great lengths to attempt to predict

Algorithm 3 Humorous Word Prediction

Inputs: Reference Voter $\mathcal{V}_\mathcal{R}$, training voters $\mathcal{V}_\mathcal{T}$
Form reference set $\mathcal{S}_\mathcal{R}$
for i in $\mathcal{V}_\mathcal{T}$ **do**
  Form training data sets $\mathcal{S}_i$
  Compute d_i as in Equation 1
end for
Select α based on Equation 2
Minimize weighted predictor loss
Return predictors

this "true" preference. The paper demonstrates that Borda Count, fused with this Mallows Model representation, is useful for predicting results of democratic processes before they occur.

3. Solution Approach

The solution herein attempts to apply the KL method of learning robustly from untrusted sources to a wholly different context than the ones explored in their paper: humor prediction. Blindly trusting the data of many different individuals is problematic, because as discussed in Section 2.1, senses of humor are quite distinct. However, it seems inefficient to let this information go to waste, and just training the model based on one individual's data seems like a recipe for overfitting. These joint insights suggest that applying KL (see Section 2.2) could prove useful, as this algorithm has demonstrated efficacy in learning from divergent sources of data.

The algorithm for the solution is the same as algorithm described by KL (see Algorithm 3 for pseudocode). However, here the goal is to model an *specific* individual's sense of humor, rather than a more generic algorithm for detecting positivity in Amazon reviews. Therefore, the target individual in question acts as the reference data set $\mathcal{S}_\mathcal{R}$, and the other subjects' results act as the unreliable data. For simplicity, the solution does not attempt to differentiate the different tiers of funny words.

For a data set $\mathcal{S}$, inputs $\mathcal{X}$ are the WEs, and the outputs $\mathcal{Y}$ are the binary classification of whether the subject found the word funny or not (i.e. whether the voter filtered this word from the round of 216 into the round of 36). Intuitively, the goal is for the algorithm to recognize individuals with similar senses of humor to the target, and weight their results accordingly in training. Similarly, the algorithm should be able to filter out those with dissimilar senses of humor (the demographic trends discussed in Section 2.1 suggest that there are, in fact, "types" of senses of humor).

4. Implementation

As the aforementioned solution approach (Section 3) relies on fusing together the algorithms and methods described in KL and GE, so too does the implementation rely on combining KL and GE's code. KL and GE are both available in public GitHub repositories[1], but modifications to both code bases were necessary in order to get them to work with each other, and a substantial amount of original code was written in order to facilitate data loading and testing. Code comments and a README file clarify this code division further.

While GE uses a variety of WE models, and conducts different experiments to see which capture humor best, for the purposes of this implementation only the FastText wiki news model[2] was used. This model represents words via 300-dimensional vectors trained on Wikipedia. The decision to use this WE model was based primarily on ease of coding, but GE showed that all of the models they looked at were able to similarly capture humor, so a negligible expect on results is expected.

KL's code utilized sklearn and tensorflow to do operations: tensorflow for optimizing the loss function 2 via gradient descent, and sklearn for conducting logarithmic regression to build linear models. KL's code was largely used for implementing the logic of the algorithm.

5. Evaluation

To verify the implementation of the algorithm described in Algorithm 3, an experiment was conducted for 15 randomly selected basis voters. For each basis voter, 10 unfunny training words, 10 funny training words, 20 unfunny testing words, and 20 funny testing words were randomly selected (according to the basis voter), and 200 other subjects were also used as training data (using those same 10 unfunny and funny training words). Though a larger training set of words would obviously be preferable (KL used 100 Amazon reviews, each possessing much more information than a single word), this was about as much data as could be utilized, considering that each subject only identified 36 funny words.

The results from this experiment do not demonstrate the model's success at predicting humor (Figure 1). While the average error rate of 0.459 is marginally better than what would be expected for totally random guessing, the result is clearly very noisy and unsatisfying, as there are numerous results which are *worse* than random guessing! Related to this unsatisfying result is the model's tendency to **always**

[1] https://github.com/limorigu/Cockamamie-Gobbledegook and https://github.com/NikolaKon1994/Robust-Learning-from-Untrusted-Sources

[2] https://fasttext.cc/docs/en/english-vectors.html

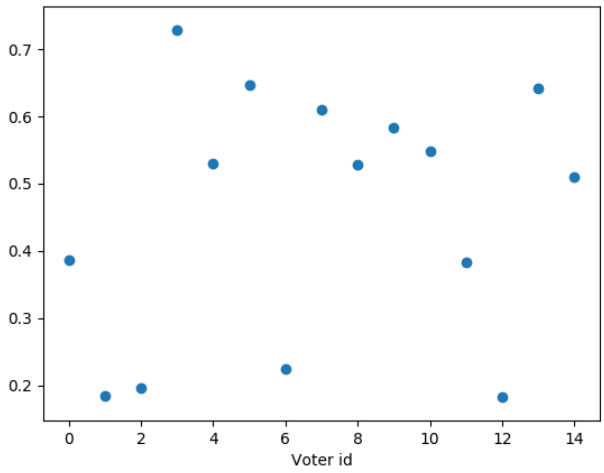

Figure 1. Results for applying the Algorithm 3 to 15 different individuals, for each utilizing 10 funny training words and 10 unfunny training words from 200 random subjects. The model was tested using 20 funny words, and 20 unfunny words. The **X-axis** refers to the Voter ID in question, and the **Y-axis** refers to the error prediction rate.

select 0 as value of the hyperparameter λ.

Why does the model always select the value zero for λ, which means the model only considers the data set most similar to the clean reference data set? Since the reference data set was included in the training data (as was also done by KL), this means that the model took the naive approach of training using *only* the reference data set, which defeats the purpose of all the complicated weighting and optimization. There are a number of possible explanations, but the most compelling are related to the noise in the data.

Recall from Section 2.1 the discussion on the noisiness of the data. The words each subject identified as funny were only compared to five other random words, not the other 180 words deemed "not funny. Therefore, it's highly likely that, given a different combination of words to vote on, a very different set of funny words would have been identified. And, since each participant in the GE crowdsourced study was forced to choose one word from a random set of 6 words, it would be virtually impossible for two people to have the same set of funny words, *even if they had the exact same sense of humor*. This could result in high discrepancies (Equation 1), leading the model to prefer a λ value of 0.

In a similar vein, one important finding from EH discussed in GE was that "the vast majority of words are not found to be funny." However, the data set for this experiment utilizes only the *funniest* words, and in the context of all words, these words are much funnier than other words. Yet,

Table 1. Funny Frequencies

'wazoo'	0.19
'poo poo'	0.14
'fufu'	0.20
'dickheads'	0.20
'douchebaggery'	0.37
'bushwhacked'	0.18
'hoo hah'	0.22
'thwacked'	0.12
'priggish'	0.11
'crack whore'	0.19

our model treats these relatively funny words as being not funny (the score is either a 1 or a 0, whereas in the context of all words, the scores should be more like 0.9 vs 0.8). Therefore, another likely explanation for the preference for a 0 value for the hyperparameter λ is a classic case of overfitting. As there are no truly unfunny words to compare against, the model overreacts to the noise in the data sets, mistaking certain qualities in the WEs as being indicative of unfunniness. In reality, it is highly likely that individual would have identified the word as funny, given a different random starting six words.

To illustrate this phenomenon of noise in funny word identification, I conducted a separate small experiment. I randomly selected one participant, randomly selected 10 funny words (out of 36), and saw what share of voters agreed that the word was "funny." The results are in Table 1. These results show that there are substantial amounts of subjects that find these words funny. Yet, out of the 1,678 subjects, 170 found *none* of these words funny! Clearly this data set is quite noisy. But what happens if a separate experiment was conducted, one that utilized fewer "corrupted" sources? After all, recall in Section 2.2 that KL's algorithm converged to perform only as well as a model trained only on the reference data when the number of corrupted sources became numerous enough. With these results in mind, I conducted a second experiment, one which utilized only 10 other individuals to act as training data sets, as opposed to 200. After running this experiment for 5 randomly selected subjects, the results were, unfortunately, similar (See Figure 2). The mean classification error was 0.38675. While nominally a smaller error, this result is likely just due to noise, as 40% of the classification errors are worse than blind guessing. The results of this experiment suggest that the problem is, in fact, more fundamental (due either to noise, or some sort of bug in the code).

6. Conclusion

The implementation based off KL, which attempted to robustly learn from an assortment of untrusted data sets, failed

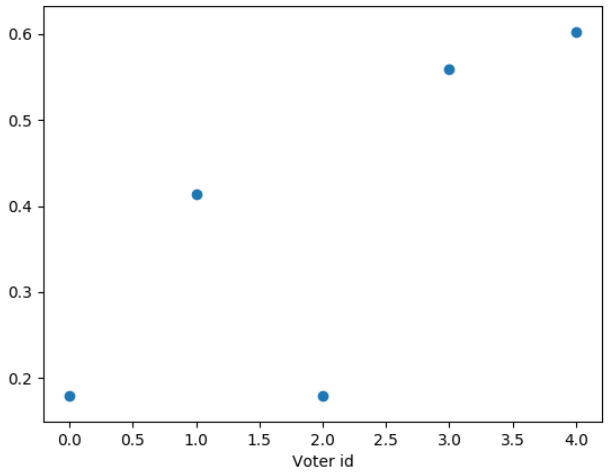

Figure 2. Results for applying the Algorithm 3 to 5 different individuals, for each utilizing 10 funny training words and 10 unfunny training words from 10 random subjects. The model was tested using 20 funny words, and 20 unfunny words. The **X-axis** refers to the Voter ID in question, and the **Y-axis** refers to the error prediction rate.

to adequately utilize the untrusted data to make better predictions in the context of humor, the prediction results being just barely better than blind guessing. This result is especially unsatisfying given that GE showed a relatively high rate of success in their "disjoint set" testing, though that task was much simpler than the one described in Section 3. However, KL showed a high degree of success on prediction tasks which were arguably more complicated, predicting the sentiment of entire Amazon reviews, as opposed to the "humor sentiment" of single words. As discussed in Section 5, this result could be due to noisy data, overfitting, user error, or some combination thereof.

There are a number of possible improvements and ways forward for improving on this implementation described in Section 3:

- Develop a voting model to vote on which word is funniest, creating a "virtual democracy" of funny words (see Section 2.3). The Mallows Model could prove especially useful, given the noisy data set, and having a virtual system of voting utilizing Borda Count represents an alternate implementation of the techniques discussed in this paper.
- In light of the lengthy discussion on the noisiness of the data set in Section 5 and Section 2.1, replicating this experiment with a less noisy data set could utilize different results. Having individuals directly rank words in a non-random fashion, as well as including more unfunny words to guard against overfitting, could provide a more useful data set for this experiment.
- With respect to the failure of applying the KL algorithm to this specific data set, perhaps a revision of the loss function 2 would be able to more effectively utilize the data from multiple sources.
- Modify the algorithm described in Section 3 such that the outputs are not just binary classifications of funny or not funny. GE describes multiple tiers of funny words: perhaps the model's performance could be improved if a 3 score represented the funniest word for an individual, and a score of 2 represented the 6 very funny words.

References

Engelthaler, T. and Hills, T. T. Humor norms for 4,997 english words. *Behavior Research Methods*, 50(3):1116–1124, Jun 2018. ISSN 1554-3528. doi: 10.3758/s13428-017-0930-6. URL https://doi.org/10.3758/s13428-017-0930-6.

Gultchin, L., Patterson, G., Baym, N., Swinger, N., and Kalai, A. Humor in word embeddings: Cockamamie gobbledegook for nincompoops. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2474–2483, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/gultchin19a.html.

Kahng, A., Lee, M. K., Noothigattu, R., Procaccia, A., and Psomas, C.-A. Statistical foundations of virtual democracy. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3173–3182, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/kahng19a.html.

Konstantinov, N. and Lampert, C. Robust learning from untrusted sources. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3488–3498, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/konstantinov19a.html.

MALLOWS, C. L. NON-NULL RANKING MODELS. I. *Biometrika*, 44(1-2):114–130, 06 1957. ISSN 0006-3444. doi: 10.1093/biomet/44.1-2.114. URL https://doi.org/10.1093/biomet/44.1-2.114.

McAuley, J. J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. *CoRR*, abs/1506.04757, 2015. URL http://arxiv.org/abs/1506.04757.