Misspelling Detection from Noisy Product Images

Varun Nagaraj Rao, Mingwei Shen
Product Assurance, Risk, and Security (PARS)
Amazon.com
{varao, mingweis}@amazon.com

Abstract

Misspellings are introduced on products either due to negligence or as an attempt to deliberately deceive stakeholders. This leads to a revenue loss for online sellers and fosters customer mistrust. Existing spelling research has primarily focused on advancement in misspelling correction and the approach for misspelling detection has remained the use of a large dictionary. The dictionary lookup results in the incorrect detection of several non-dictionary words as misspellings. In this paper, we propose a method to automatically detect misspellings from product images in an attempt to reduce false positive detections. We curate a large scale corpus, define a rich set of features and propose a novel model that leverages importance weighting to account for within class distributional variance. Finally, we experimentally validate this approach on both the curated corpus and an out-of-domain public dataset and show that it leads to a relative improvement of up to 20% in F1 score. The approach thus creates a more robust, generalized deployable solution and reduces reliance on large scale custom dictionaries used today.

1 Introduction

Misspellings often occur on printed descriptions, warranty cards and advertisements associated with products despite the ubiquitous presence of spell checkers. Although these misspellings are often due to human typographical errors (Kukich, 1992), or by non-native writers who are confused by language intricacies, they may also be deliberately introduced by malicious elements to deceive stakeholders (Gong et al., 2019). A single inadvertent or intentional misspelling can propagate to large amounts of inventory. As a consequence and much to the chagrin of sellers, misspellings cost millions in lost online sales (Coughlan, 2011) and breaks customer trust.

This paper aims to automatically and accurately identify misspellings from product images in order to improve customer experience. We loosely *define* a misspelling as "incorrectly spelled ubiquitous words and proper nouns which are very similar to their correct forms". Example images and words are present in Figure 1 and Table 1 respectively. As can be seen, product images predominantly contain many

Figure 1: Product Images after OCR processing

Misspellings	knder, Wnite, Excepional, accommodat, PROFESSINALS, FLAWLBSS, harmfiil, ckytime, remaimin, ELOW, Replocemen, favoeite, HONOURAS, limietd
Non Misspellings	FD&C, intro-, duced, ATION, HSOOMS, #1, alaudoid, yeqioos, recommended, forwhiter, JavaPresse., (260ml), HT1S1A10d, Magnesiumstearate, No.5Al, MICHAEL, 7.10104E+11, Fabrique, d'origine, politique

Table 1: Sample Product Image Instances

Proceedings of the 28th International Conference on Computational Linguistics: Industry Track, pages 124–135
Barcelona, Spain (Online), December 12, 2020

non-dictionary words including quantities, identification numbers, ingredients and other proper nouns, which should not be detected as misspellings. Further, text extraction from these noisy images is challenging since they often suffer from lighting, orientation, text occlusion and font dissimilarities issues. As a result, the class distribution is skewed with a very small number of misspellings amidst a large set of non misspellings. For instance, in Figure 1, "knder" and "Wnite" (present in the top row images) are the only two misspellings that are to be detected among several other words.

Leveraging existing approaches for misspelling detection from product images is beset with a number of challenges. First, although spelling research has intrigued the NLP community for long (Damerau, 1964; Kukich, 1992), misspelling detection research (Zamora et al., 1981; Dalkiliç and Çebi, 2009; Attia et al., 2012; Yu and Li, 2014) is very sparse, language specific and the primary approach has remained a dictionary lookup. This approach does not scale or generalize to billions of product images leading to a large number of false positive detections. Second, unlike most language modeling tasks, the use of contextual cues is ineffective. As a misspelling might be deliberate, its occurrence is unlikely to be influenced by its neighbouring words. Further, due to varied image quality and orientation, the extracted words may be scattered and not in the order they appear on the image. And third, to the best of our knowledge, there is no publicly available large scale corpus of product images annotated for misspellings. To address these deficiencies we make the following contributions.

- We curate Image-MisSpell [1], the first large scale corpus of product images and novel language agnostic misspelling features, explicitly annotated for misspellings and augmented with publicly available misspelling data. (Section 4.1, 4.2)
- We propose a weakly supervised cost sensitive weighted SVM model that leverages importance weighting of instances to handle within class distributional variance. The model can scale and generalize well to unseen examples. The small model size and low inference latency make it ideal to be used in a production environment. (Section 4.3)
- Finally, we present extensive experiments, results and discussions of the model and observe an improvement in F1-score of up to 20% over the dictionary lookup baseline. This result is promising because curating a custom set of non-dictionary words to filter for billions of products is prohibitively expensive and our model achieves better results with significantly lower cost. (Section 5, 6).

The remaining sections contain an overview of spelling related literature (Section 2), problem formulation (Section 3) and finally our conclusions and plans for further research (Section 7).

2 Related Work

Misspelling detection research is very limited, small scale and often on domain specific private data (Zamora et al., 1981). Approaches for misspelling detection primarily involve use of a predefined dictionary of n-grams (Zamora et al., 1981) and words (Dalkiliç and Çebi, 2009; Yu and Li, 2014; Attia et al., 2012). Additionally, dictionaries used are limited to specific languages like Chinese (Yu and Li, 2014), Turkish (Dalkiliç and Çebi, 2009) and Arabic (Attia et al., 2012).

Most existing work on spelling has focused on spelling error correction (Kukich, 1992; Ng et al., 2014; Chollampatt and Ng, 2018; Nagata et al., 2017; Hagen et al., 2017; Flor et al., 2019; Flor, 2012; Toutanova and Moore, 2002; Brill and Moore, 2000; Hasan et al., 2015). Initial approaches for correcting misspelling include those that compute edit distance and phonetic similarity between misspelling and candidate correction. Consequently, Flor (2012) introduced an approach to combine ranking candidate corrections using contextual cues with edit distance and phonetic similarity. Although several datasets exist for spelling correction research, many of these are small and proprietary corpora. Flor et al. (2019) proposed TOEFL-Spell as the first large scale corpus to benchmark spelling correction performance.

Spelling research is transitioning to leverage word embeddings with the advent of deep learning. Since pretrained word embeddings like Word2Vec (Mikolov et al., 2013b; Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) do not support Out-Of-Vocabulary (OOV) words such as a misspelling, Bojanowski et al. (2017) and Piktus et al. (2019) introduced approaches for obtaining word embeddings resilient to misspellings. However, misspelling embeddings leverage context during training and have

[1] https://github.com/amzn/image-misspell-coling2020

been designed to be close to their correct variants. This is counter intuitive to our task where we would like to disentangle misspellings from non misspellings and the occurrence of a misspelling is unlikely to be influenced by its neighbouring words present on product images.

3 Problem Formulation

Given an image *I* consisting of *n* words, and an OCR Engine *O* which extracts words w_i from the image not necessarily in the order they appear:

$$O(I) = \{w_1, w_2, ..., w_n\} \tag{1}$$

we learn a model f such that for each word w_i and feature extractor g

$$(f \circ g)(w_i) = \begin{cases} 1 & w_i \text{ is a misspelling} \\ 0 & w_i \text{ is a non misspelling} \end{cases} \tag{2}$$

A word *w* is characterized as a *misspelling* if it loosely satisfies one or more of the following conditions:

- Is *not* present in a dictionary of *any* language
- Differs from its correctly spelled word form in only *very few characters*
- In most cases, does *not* contain punctuation, digits and special characters
- In most cases, contains only alphabets

An important distinction to our problem formulation is that a word *w* is a non misspelling, not only if it is present in the dictionary, but also if it doesn't satisfy any of the other conditions for a misspelling outlined above. This implies there could be words which are not present in the dictionary but are non misspellings. Further, our focus is on modelling words to detect misspellings *after* they have been extracted from images. We do not make any changes to either the existing OCR pipeline *before* word extraction or any potential misspelling correction approaches *after* misspelling detection.

4 Proposed Approach

We now describe the process of data curation, feature extraction and model learning.

4.1 Data Curation

To detect misspellings from product images, we create the Image-MisSpell Corpus curated from two distinct sources (1) Bin Check Images and (2) Public Corpora of Misspellings (Mitton, 2007).

4.1.1 Bin Check Images

Products are stored in large containers or *bins*. Based on requests investigators perform a bin check and capture product images. These images may include external packaging, internal product accessories, and documentation. Investigators may augment the captured images with product and brand images found online. As seen in Figure 1, these product images are very noisy. They contain diverse font styles and suffer from illumination issues, blurriness, rotation and occlusions making the process of extracting text using an OCR system, annotating the misspellings and detecting misspellings by avoiding false positives, a very challenging task. Once the product images were collected, words were extracted using AWS Textract (Amazon-AWS, 2019) and annotated for misspellings. Samples are present in Table 1.

As is evident from both Figure 1 and Table 1, the images contain a lot of non-dictionary instances such as quantities, punctuation, special characters, identification numbers, dates, zipcodes, ingredient lists and proper nouns that should be flagged as a non misspelling. Further adding to the challenge is that although the images predominantly contain English text, they occasionally do also contain multilingual non-English text. The Bin Check Images dataset is summarized by category in Table 2. The dataset is very skewed and contains only 0.5% misspelled words in relation to 99.5% of non misspelled words. As the dataset grows over time, the skew is only expected to get worse. This could lead to a large number of false positive detections by a classifier. To ensure our corpus is adequately balanced and representative, rather than waiting on the time-consuming and expensive process of using human investigators, we

# Misspelled Words	57 (0.5%)
# Non Misspelled Words	10617 (99.5%)

Table 2: Bin Check Images Dataset Summary

leverage a large public corpus of misspellings (Mitton, 2007) in a *weakly supervised* manner. Although the misspellings present in (Mitton, 2007) were not obtained from product images, their features are similar to those extracted from the Bin Check Images dataset (see Figure 2a) and can thus be used to correct class imbalance.

4.1.2 Public Corpora of Misspellings

The corpus has been curated by Mitton (2007), as an amalgamation of four datasets, to promote spellchecking research. It has a single correctly spelled English word which is present in a dictionary and is mapped to one or more spelling errors which may or may not be present in a dictionary. The constituent datasets are described below.
(1) birkbeck : It contains 36,133 misspellings of 6,136 words. It includes results of handwritten spelling tests run on students. Correct spellings are given in Oxford English form and misspellings due to American forms have been explicitly excluded.
(2) holbrook : It contains 1791 misspellings of 1200 target words (including 20 of unknown targets represented as '?'). The passages are taken from (Holbrook, 1964). They are extracts from the writings of secondary-school children, in their next-to-last year of schooling.
(3) aspell : It contains 531 misspellings of 450 words for testing the GNU Aspell Spellchecker.
(4) wikipedia : It contains 2,455 misspellings of 1,922 words made by Wikipedia editors.
To curate a single corpus we performed the following steps:

- Removed redundancy by mapping unique correct words to unique misspellings across the datasets.
- Removed words with unknown targets.
- Removed misspellings that are present in a dictionary and corresponding correct words if they do not have any misspellings associated with it after deletion. For instance, we noticed that "here" was a misspelling of "there" whereas by our problem formulation, "here" is not a misspelling as it is present in the dictionary.
- Replace the underscore by a space. Spaces are represented by an underscore in these datasets.

Sample examples from the dataset categorized by class is present in Table 3.

Misspellings	acomedation,acruied, expaine, exterordenary, heven, inconviencence, leaft, misstake
Non Misspellings	accommodation, accrued, explained, extraordinary, heaven, inconvenience, left, mistake

Table 3: Public Corpus Sample Instances

In contrast to the samples from the Bin Check Images in Table 1, these words are very clean, predominantly in English, contain very few proper nouns and have almost no punctuation and special characters other than a space. After performing the above mentioned steps, we ended up with a single dataset which we henceforth term as the "public corpus". The public corpus' summary statistics are summarized in Table 4. Since the distribution of samples is skewed towards more misspellings than non misspellings, it serves to complement the Bin Check Images dataset well.

# Misspelled Words	32051 (80%)
# Non Misspelled Words	8179 (20%)

Table 4: Public Corpus Dataset Summary

4.1.3 Image-MisSpell Corpus

We created the Image-MisSpell Corpus from both the Bin Check Images Dataset and the Public Corpus. We used all the samples of the Bin Check Images and randomly sampled instances from the Public Corpus

so that the class distribution is balanced. Consequently, we used a stratified sampling strategy to create a 90:10 train/test split that preserved the class balance in both the splits and ensured the distributions are similar. These splits can be used to build models for automatic detection of misspellings. The Image-MisSpell Corpus is summarized in Table 5.

Class	#Train Words	#Test Words
Misspelling	15027	1600
Non Misspelling	15064	1600

Table 5: Image-MisSpell Corpus Dataset Summary

4.2 Feature Extraction

The performance and success of machine learning models is heavily dependent on the features chosen. Different representations can hide or reveal the diverse factors that explain the variation in the data. Although automatic feature representation and transfer learning has shown promising results on several NLP benchmark tasks, it is not directly applicable for the task of misspelling detection since (1) context does not affect misspellings on product images and (2) pretrained misspelling embeddings (Piktus et al., 2019) are designed to be close to their correct variants. Thus, we chose to perform traditional feature engineering relying on domain knowledge and generic priors. A blend of language specific and language agnostic features are chosen. We define the features below.

4.2.1 Standalone Features

These features are extracted from a single word and do not require any additional information.
(1) `textract_score`: The AWS Textract Word Recognizer produces a confidence score associated with each recognized word from the image (Litman et al., 2020). Textract can detect Latin-script characters from the standard English alphabet and ASCII symbols. The prediction confidence score is calculated as the average of the CTC decoder probabilities until the end-of-sequence token. The score is bounded in the range [0,1] with higher scores for more confident predictions. Since the Public Corpus does not have an associated confidence score, we simulate the score using a uniform sampling from the 95% confidence interval (in Table 6) of the textract scores of the Bin Check Images Corpus. This ensures a non zero variance across all samples and preserves the features characteristics. Ignoring the values completely or filling in with order statistics or mean/median/mode is not appropriate when number of missing values is very high. This feature is included as we are interested in words that have mostly been correctly recognized by the OCR system.

Stats \ Class	Misspelling	Non Misspelling
Mean	0.97	0.90
Max	1.00	1.00
Min	0.39	0.01
95% Conf.Int.	[0.95, 0.99]	[0.89, 0.90]

Table 6: Bin Check Images `textract_score` distribution

(2) `ner_score`: We obtain the probability of each word being a named entity from AWS Comprehend (Amazon-AWS, 2017). The decoder probability predicted for each token is averaged. This basically tells us given confidence score of each token, how likely the whole span will be an entity (Shen et al., 2017). We default to the English language model of Comprehend. Using this feature, the model would thus have the ability to exclude named entities as misspellings while still being able to classify their variants as misspellings.
(3) `pos_score`: We obtain the probability of each word being a proper noun (PROPN) from AWS Comprehend's Part-Of-Speech (POS) Tagger (Amazon-AWS, 2017). We default to the English language model of Comprehend. Using this feature, the model would have the ability to exclude proper nouns as misspellings while being able to classify their variants as misspellings.

(4) `gibb_score`: We calculate the probability of a word being gibberish using a first order markov chain. The model should exclude gibberish from the predicted misspellings. We pretrain a 27 x 27 matrix of bigram probabilities (26 lowercase characters and a space) using a text file [2] of a million English words. It is a concatenation of public domain book excerpts from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus. For each word encountered, we use the matrix to calculate the joint probability of occurrence of all possible bigrams formed from the chosen character set. OOV words are expected to have a low probability of occurrence. Further, long gibberish words are also expected to have a low joint probability due to the product of several low probability bigrams. In the practical application, the gibberish score can also help with OCR errors. For instance, a gated structure on an image may be incorrectly recognized as a series of alphabets (l's or x's) even though it does not actually contain any letters and is only a graphic resembling text. The joint probability in these cases is also expected to be low. Given a matrix *M* of bigram probabilities, and ordered sequence of *n* characters $[c_1, ..., c_n] \in M$, the gibberish score of word *w* is calculated as follows:

$$\text{gibb\_score(w)} = \text{gibb\_score}([c_1, ..., c_n]) = \prod_{i=1}^{n-1} (M[c_i, c_{i+1}]) \tag{3}$$

(5) `word_length`: Integer length of a word.
(6) `contains_nonalpha`: Boolean value that indicates whether word *w* contains non alphabetical characters. Misspellings usually do not contain punctuation, special characters and digits.

$$\text{contains\_nonalpha} = \begin{cases} 1, & w \text{ contains non alpha} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

(7) `dict_presence`: Boolean value for whether the word *w* is present in a dictionary. We make use of the Hunspell dictionary (Németh, 2010) with its Python bindings (Latinier, 2014). We leverage dictionaries of prevalent languages [en_us, en_gb, en_au, en_ca, en_za, es, de, fr, pt, it, ru] from (Wormer, 2018) since product images occasionally contain multilingual text. In addition to checking for dictionary presence of the word in its recognized form, we also check for presence of its stemmed and lowercase forms. PyHunspell provides flexibility to add in new words to the dictionary at runtime. We insert a set of about 100 product specific words. Words present in a dictionary of any language should not be misspellings. This feature ensures the model is less sensitive to a specific language.

$$\text{dict\_presence} = \begin{cases} 1, & w \text{ present in a dict} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

4.2.2 String Similarity Features

In order to leverage a weak context, we obtain string similarity metrics using the autocorrected form of the word with least edit distance. For the Bin Check Images Dataset we use the PyHunspell module to suggest the closest autocorrect word. The autocorrected form of the word may be from any of the language dictionaries. Ties are broken based on the order of the listed languages in the dict_presence feature. The Public Corpus Dataset already has a word mapped to its correct form.
(1) `editdistance`: We calculate the Levenshtein distance (Hyyrö, 2001) with equal weight for insertion, deletion and substitution. This is further normalized by dividing by the maximum of the lengths of the two words ensuring the value obtained is bounded in the range [0,1].
(2) `jaccard_similarity`: Jaccard Similarity is another well-known measure of the similarity of a pair of strings. It captures information orthogonal to edit distance. Given two *sets* of unique lowercase characters w_1, w_2 the jaccard similarity is defined in Equation 6.

$$\text{jaccard\_similarity} = |w_1 \cap w_2| / |w_1 \cup w_2| \tag{6}$$

[2] `https://norvig.com/big.txt`

(3) `string_containment`: Sometimes an indicator of a match is that the letters in one string are entirely or partially contained in the other string. String containment captures this notion. Given two *sets* of unique lowercase characters w_1, w_2 the string containment is defined in Equation 7.

$$\text{string\_containment} = |w_1 \cap w_2|/min(|w_1|, |w_2|) \tag{7}$$

(4) `char_intersection_count`: This is simply the number of unique lowercase intersecting characters given two sets of characters w_1, w_2.

$$\text{char\_intersection\_count} = |w_1 \cap w_2| \tag{8}$$

4.3 Importance Weighting and Learning Using Privileged Information:

Once we have represented the Image-MisSpell corpus using the features described, we are able to build models for automatic detection of misspellings. However, we must keep in mind the data and class distributions before modeling. Most machine learning models perform poorly when there is a skew in the class distribution. With a greater class imbalance ratio, the decision function usually favours the majority class. Two strategies for handling the class imbalance include (1) over sampling the minority class using approaches like SMOTE (Chawla et al., 2002) and ADASYN (He et al., 2008) and (2) under sampling the majority class. Given the abundance of data due to the augmentation with the Public Corpus, we were able to generate a class balanced corpus by under sampling the majority class.

Now, although the classes are balanced, we noticed that the distributions within each class varied. The Public Corpus non misspellings are very clean and contain words present in a dictionary whereas the Bin Check Images non misspellings class is very noisy and contains words not present in the dictionary. Further, the actual number of misspellings from the Bin Check Images is very small. An approach to correct the sampling bias, used in active learning (Beygelzimer et al., 2009) and weighted SVMs (Lapin et al., 2014), is called *importance weighting*. Lapin et al. (2014) state that "prior knowledge expressible with privileged features can be encoded by weights associated with every training example". Higher the weight, more emphasis the classifier puts on that instance to classify correctly. The Instance Weighted or Cost Sensitive SVM (He and Ma, 2013; Yang et al., 2007) is able to implicitly incorporate importance weighting into its objective. We now discuss SVM optimization formulation incorporating the instance weighting.

Given *n*, *d* dimensional training instances $x_i \in \mathbb{R}^d, \forall i \in [1, n]$, binary labels $y \in \{1, -1\}^n$, a kernel function ϕ, the SVM algorithm obtains $w \in \mathbb{R}^d, b \in \mathbb{R}$ such that the prediction $sign(w^T\phi(x_i) + b)$ is correct in most cases. The primal objective is:

$$\min_{w,b,\zeta} \frac{1}{2}w^T w + C\sum_{i=1}^{n}\zeta_i, \text{ subject to } y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i, \text{ where } \zeta_i \geq 0, i = 1, ..., n \tag{9}$$

Since data is not always perfectly separable by a hyperplane, few samples are allowed to be at a distance of ζ_i from their correct margin boundary. The C controls the strength of this penalty. As each example in the training dataset has its own weight, the instance C_i can be calculated as a weighting of the global C such that $C \in \mathbb{R}^n$. A larger weighting is used for lesser representative instances allowing the margin to be softer and preventing misclassified examples. In other words, the modified SVM algorithm would *not* skew the hyperplane towards the distribution with larger samples since the minority samples are assigned a higher misclassification cost. The weights are inversely proportional to the number of samples that satisfy the condition, i.e, smaller the number of samples, higher is the weight.

$$C_i = \text{instance\_weight}_i * C \tag{10}$$

5 Experimental Setup

In this section, we describe the experimental set up used for training, the model used and feature ablations.
Datasets. We use the Image-MisSpell and an out-of-domain publicly available dataset TOEFL-Spell

Rule	Weight
Initial Weight for All samples	= 1
'filename' == 'bin_check'	= 2
'contains_nonalpha' == 1 && 'is_error' == 0	= 5
'filename' == 'bin_check' && 'is_error' == 1	= 10
'filename' == 'bin_check' && 'is_error' == 0	= 15
'filename' == 'bin_check' && 'is_error' == 0 && 'dict_presence' == 0	= 20
'textract_score' $\geq$ 0.95 && 'is_error' == 1	+2
'textract_score' $<$ 0.1	-1

Table 7: Instance Weight values. Each instance initially has a weight of 1. The weights are then modified sequentially in the order of the rules listed.

(Flor et al., 2019) for our experiments. The Image-MisSpell dataset includes about 30K train samples and 3.2K test samples. Both the train and test splits are class balanced. More details have been described in Section 4.1. The TOEFL-Spell is a corpus of learner essays, annotated for spelling errors. We preprocess and extract out features from the TOEFL-Spell dataset similar to the Public Corpus and obtain 3699 misspellings and 2343 non misspellings. All models have been trained using the Image-MisSpell trainset. For evaluation, we use both the Image-MisSpell and the TOEFL-Spell testsets.

Models. We present two baselines that involve table lookups and thresholds that *do not* include a learning component (1) Hunspell dictionary presence lookup and (2) heuristics model. The machine learning model we used is a Cost Sensitive / Weighted SVM (WSVM) with linear and RBF kernels chosen due to the implicit ability to allow importance weighting and interpretability using linear feature weights. We also present feature ablations using the standalone and string similarity features in isolation.

Implementation details. For the heuristics model, the features and conditions chosen empirically by human annotators include $[\text{textract\_score} < 0.93, \text{ner\_score} \geq 0.1, \text{pos\_score} \geq 0.99, \text{gibb\_score} \leq 0.01, \text{word\_length} \leq 2, \text{contains\_nonalpha} == 1, \text{dict\_presence} == 1]$. If any of the conditions are satisfied, the word is classified as a non misspelling, else it is a misspelling. The importance weight heuristic values are present in Table 7. Both absolute and relative values are used. The SVM parameters are RBF kernel coefficient $\gamma = 1/(num\_features * \sigma^2(train\_set))$ and a vector C as described in Table 7 and Equation 10.

6 Results and Discussion

Experiment		Image-MisSpell			TOEFL-Spell		
		P	R	F1	P	R	F1
Baseline	Hunspell Dictionary Lookup	0.61	0.99	0.76	0.94	1.00	0.97
	Heuristics with Thresholds	0.89	0.70	0.79	0.99	0.73	0.84
Model	SVM + Linear Kernel	0.74	1.00	0.85	0.98	1.00	0.99
	SVM + RBF Kernel	0.78	1.00	0.87	0.99	1.00	1.00
	WSVM + Linear Kernel	0.87	0.96	0.91	0.98	0.96	0.97
	WSVM + RBF Kernel	0.86	0.97	0.91	0.99	0.96	0.98
Ablation	WSVM + Linear (only standalone features)	0.85	0.97	0.90	0.98	0.96	0.97
	WSVM + Linear (only string similarity features.)	0.68	0.75	0.71	0.98	0.60	0.74

Table 8: Experimental Results on the Image-MisSpell and TOEFL-Spell datasets.

We report results using standard classification metrics of precision, recall and F1-score in Table 8. On the Image-MisSpell corpus, which contains many non-dictionary words as non misspellings, the WSVM models indicate up to 20% relative improvement in F1-scores. As expected, this behaviour is not observed on the TOEFL-Spell dataset which does not have any non-dictionary words as non misspellings. However, results are comparable to a dictionary lookup. Without using instance weighting, which is tailored to the Image-MisSpell distribution, we get near perfect detection on the TOEFL-Spell dataset. In most cases and across the two datasets, the SVM and WSVM models perform better than the baselines. The choice of kernel (linear or RBF) does not alter performance significantly. The performance of the WSVM is better than the SVM on the Image-MisSpell, but not so on TOEFL-Spell. This indicates that the importance weighting utilized by the WSVM helps improve performance when the underlying distributions vary,

unlike the TOEFL-Spell dataset which is very clean and without a within class distributional variance. Feature ablations indicate that the exclusion of either standalone or string similarity features causes a reduction in performance.

Benefits of Hand Crafted Features. There are finer distinctions to the type of misspellings and non misspellings that may occur. In Table 9 we detail the form of word encountered and the most prominent corresponding features which help disentangle them.

Form of Word	Classification	Prominent Feature(s)
Not present in any dictionary and differs from the correct form in few edit distances	Misspelling	Standalone and String Similarity Features
Not in any dictionary but is a correctly spelled domain word (entities, terminology)	Non Misspelling	dict_presence, ner_score, pos_score
Misspelled domain word (entities, terminology)	Misspelling	String Similarity Features, dict_presence, ner_score, pos_score
Random collection of alphabets or alpha-numeric characters	Non Misspelling	gibb_score, contains_nonalpha

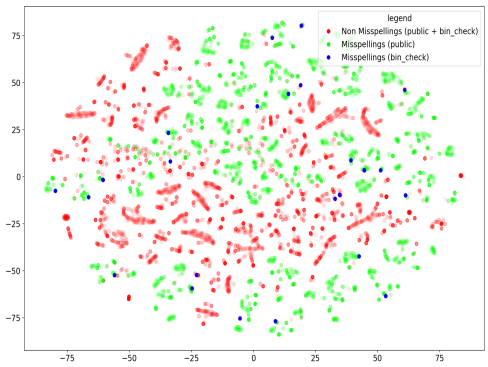
Table 9: Details about the type of misspellings that may occur and corresponding features that help accurately detect them.

Weak Supervision. Given a small set of labeled data, we can express functional invariances as weak label distributions. In this way we view techniques such as data augmentation as a form of weak supervision (Ratner et al., 2017). Data augmentation was necessary due to the skewed class distribution on the Bin Check Images Dataset as shown in Table 2. As the Bin Check Images dataset matures, the skewness is expected to get worse and a means to correct for this imbalance is required. Without augmentation, the classifer could choose to ignore the misspelling class entirely and still lead to a 99.5% accuracy. Although data augmentation helped correct the class imbalance, the Public Corpus misspelling class needed to have a similar distribution to that of the Bin Check Images misspellings. This distributional similarity of the misspellings is evident in the 2-dimensional TSNE plot of the Image-MisSpell trainset is depicted in Figure 2a. Since the distribution of misspellings of the public corpus was similar to the Bin Check Images the data augmentation was a reasonable choice.

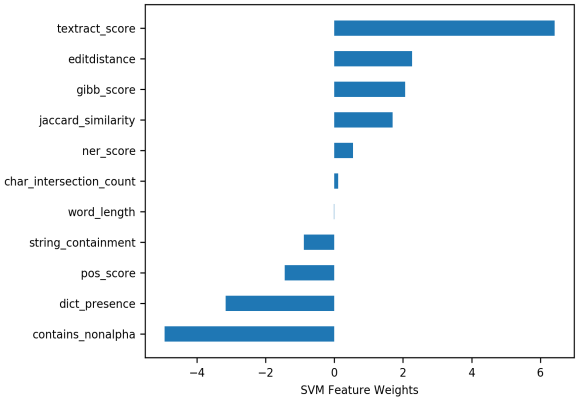
Interpretability and Explainability. Machine learning systems are increasingly making or informing several critical decisions. However, they are often opaque black boxes that are not explainable. A key requirement in this study was to have a model whose decision can be *explained* by a human when required. The WSVM + Linear model decision can be easily interpreted by examining the learned feature weights (Figure 2b). Positive weighted features contribute to the misspelling labels and negative weighted features contribute to non misspelling labels. We notice that the weights learned are inline with human interpretations and the models decision can be held accountable.

Model Deployment. The WSVM + Linear model takes up only a few kilobytes of spaces and has very low inference latency. In addition, the instance weights provided a useful toggle to tweak the feature importance based on runtime performance, thus making this model suitable for a production environment.

Use of Word Embeddings. Although techniques such as Word2Vec (Mikolov et al., 2013b; Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) have been extensively used, they cannot provide embeddings for OOV words such as misspellings. FastText by Bojanowski et al. (2017) and Misspelling Oblivious Embeddings (MOE) by Piktus et al. (2019) introduced approaches for obtaining word embeddings for OOV words. Although FastText can capture morphological aspects of text, it may not be particularly resistant to misspellings which can occur also within the dominant morphemes. MOE on the other hand combines the FastText loss with a spell correction loss giving explicit importance to misspelled words. However, these are supervised tasks which embed misspellings close to their correct variants. Since our task involves disentangling misspellings from their correct variants, the direct use of these embeddings without the use of hand crafted features may not be ideal. Further, pretrained MOE have not been released publicly at the time of writing this paper. An interesting experiment, which we leave as future work, is to augment the string similarity features with MOE since both are expected to capture



(a) TSNE visualization of instances.



(b) SVM Classifier Feature Weights.

Figure 2: The misspelling features of the bin_check images are *mostly* distributed amongst the public misspellings features as seen in (a). Feature weights are in line with human interpretation in (b).

similarity between a misspelling and its closely related correctly spelled word form, complimenting the feature space.

Error Analysis. We perform an error analysis of the WSVM+Linear model used in production. Qualitative examples are present in Table 10. Words present in the dictionary of popular languages ("environment", "bonjour"), proper nouns ("Microsoft", "Amazon"), gibberish ("jncjnvuhebvioheosv env") and words with special characters/digits ("(njcwjncp3e9r") are rightly classified as non misspellings. Words which mostly satisfy the conditions in Section 3 such as "mroning", "Micorsoft", and "precison", are rightly classified as misspellings. However, the model does occasionally make mistakes. Even though "jncjnvuhebvioheosv" is gibberish, and "Amazn", "environmen" differs in only a few characters from its correct form the model incorrectly classifies them as non misspellings. A character level model may be able to accurately disentangle correct words like "environment" from many more of its incorrect forms like "environmnt" and "environemnt" by better learning about similar character patterns. We must also note that in certain cases this behaviour is due to the insufficient representative samples in the trainset that the importance weighting is unable to correct for. For instance, we noticed that long words with no meaning are extremely rare ($<0.5\%$) and hence the gibberish detector mostly does not fire on them. Additional samples, representative of the system's use case in practice can correct for these errors.

Misspellings	mroning, environmnt, Micorsoft, Facebdok, precison, bonjuor, jncjnvuhebvioheosv, helli, helo
Non Misspellings	jncjnvuhebvioheosv env, environment, environmen, Amazon, Amazn, Microsoft, (njcwjncp3e9r, bonjour, #4tg%ju, lake union, marketplace, he$$o, 10/23/2019, 50oz, MagnesiumStearate

Table 10: Qualitative Example Results Produced by the WSVM+Linear Model

7 Conclusion and Future Directions

This paper addressed the problem of automatic detection of misspellings from product images. We presented Image-MisSpell, a large dataset of product images augmented with public corpora, exhaustively annotated for misspelling detection. We also defined a rich set of features that can be extracted from words without context. Further, we proposed a cost sensitive weighted SVM model that leverages importance weighting to account for the within class distributional variation. Finally through experiments we demonstrated the enhanced performance of this model in relation to baseline approaches. The approach is promising since it achieves better results with a lower time, monetary and human capital investment. Future directions include leveraging deep learning where character level language modeling could be used to learn embeddings that disentangle misspellings from their correct form. Visual features like Pyramidal Histogram of Characters (PHOC) (Almazán et al., 2014) can also provide important cues for OCR errors and deliberate misspellings.

Acknowledgments

We thank Srikar Appalaraju for scoping out this problem. We also thank Karen Hovsepian, Shuping Ji and Thi Nhat Anh Nguyen for providing constructive feedback about this work. We also extend our gratitude to the reviewers who provided us with very constructive and detailed comments. Finally, we'd like to thank Qingzhou Yu, Ananth Kasturirangan, Kirsty Bolden, Will Lewis and others from tech and business for their support.

References

Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. 2014. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2552–2566.

Amazon-AWS. 2017. Comprehend. `https://aws.amazon.com/comprehend/`.

Amazon-AWS. 2019. Textract. `https://aws.amazon.com/textract/`.

Mohammed Attia, Pavel Pecina, Younes Samih, Khaled Shaalan, and Josef Van Genabith. 2012. Improved spelling error detection and correction for arabic. In *Proceedings of COLING 2012: Posters*, pages 103–112.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance weighted active learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 49–56.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting on association for computational linguistics*, pages 286–293. Association for Computational Linguistics.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sean Coughlan. 2011. Spelling mistakes 'cost millions' in lost online sales, `https://www.bbc.com/news/education-14130854`. *BBC News*.

Gökhan Dalkiliç and Yalçin Çebi. 2009. Turkish spelling error detection and correction by using word n-grams. In *2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control*, pages 1–4. IEEE.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Michael Flor, Michael Fried, and Alla Rozovskaya. 2019. A benchmark corpus of english misspellings and a minimally-supervised model for spelling correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 76–86.

Michael Flor. 2012. Four types of context for automatic spelling correction. *TAL*, 53(3):61–99.

Hongyu Gong, Yuchen Li, Suma Bhat, and Pramod Viswanath. 2019. Context-sensitive malicious spelling error correction. In *The World Wide Web Conference*, pages 2771–2777.

Matthias Hagen, Martin Potthast, Marcel Gohsen, Anja Rathgeber, and Benno Stein. 2017. A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1261–1264.

Saša Hasan, Carmen Heger, and Saab Mansour. 2015. Spelling correction of user search queries through statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 451–460.

Haibo He and Yunqian Ma. 2013. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE.

David Holbrook. 1964. English for the rejected: Training literacy in the lower streams of the secondary school.

Heikki Hyyrö. 2001. Explaining and extending the bit-parallel approximate string matching algorithm of myers. Technical report, Citeseer.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *Acm Computing Surveys (CSUR)*, 24(4):377–439.

Maksim Lapin, Matthias Hein, and Bernt Schiele. 2014. Learning using privileged information: Svm+ and weighted svm. *Neural Networks*, 53:95–108.

Benoit Latinier. 2014. PyHunspell. `https://github.com/blatinier/pyhunspell`.

Ron Litman, Oron Anschel, Shahar Tsiper, Roee Litman, Shai Mazor, and R Manmatha. 2020. Scatter: selective context attentional scene text recognizer. *arXiv preprint arXiv:2003.11288*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Roger Mitton. 2007. Corpora of misspellings for download. `https://www.dcs.bbk.ac.uk/~ROGER/corpora.html`.

Ryo Nagata, Hiroya Takamura, and Graham Neubig. 2017. Adaptive spelling error correction models for learner english. *Procedia Computer Science*, 112(C):474–483.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

László Németh. 2010. Hunspell. `https://github.com/hunspell/hunspell`.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Édouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3226–3234.

Alex Ratner, Stephen Bach, Paroma Varma, and Chris Ré. 2017. Weak Supervision: The New Programming Paradigm for Machine Learning. `https://dawn.cs.stanford.edu/2017/07/16/weak-supervision`.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction.

Titus Wormer. 2018. Hunspell Dictionaries. `https://github.com/wooorm/dictionaries`.

Xulei Yang, Qing Song, and Yue Wang. 2007. A weighted support vector machine for data classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(05):961–976.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

EM Zamora, Joseph J Pollock, and Antonio Zamora. 1981. The use of trigram analysis for spelling error detection. *Information Processing & Management*, 17(6):305–316.