Spelling Correction of User Search Queries through Statistical Machine Translation

Saša Hasan[*] sasa.hasan@gmail.com

Carmen Heger[†] heger.carmen@gmail.com

Saab Mansour saamansour@ebay.com

eBay Inc.
2065 Hamilton Ave
San Jose, CA 95125, USA

Abstract

We use character-based statistical machine translation in order to correct user search queries in the e-commerce domain. The training data is automatically extracted from event logs where users re-issue their search queries with potentially corrected spelling within the same session. We show results on a test set which was annotated by humans and compare against online autocorrection capabilities of three additional web sites. Overall, the methods presented in this paper outperform fully productized spellchecking and autocorrection services in terms of accuracy and F1 score. We also propose novel evaluation steps based on retrieved search results of the corrected queries in terms of quantity and relevance.

1 Introduction

Spelling correction is an important feature for any interactive service that takes input generated by users, e.g. an e-commerce web site that allows searching for goods and products. Misspellings are very common with user-generated input and the reason why many web sites offer spelling correction in the form of "Did you mean?" suggestions or automatic corrections. Autocorrection increases user satisfaction by correcting obvious errors, whereas suggestions make it convenient for users to accept a proposed correction without retyping or correcting the query manually. Spelling correction is not a trivial task, as search queries are often short and lack context. Misspelled queries might be considered correct by a statistical spelling correction system as there is evidence in the data through frequent occurrences.

While common successful methods (cf. Section 1.1) rely on either human-annotated data or the entire web, we wanted to use easily accessible in-domain data and on top of that technology that is already available. In this work, we use user event logs from an e-commerce web site to fetch similar search query pairs within an active session. The main idea is that users issue a search query but alter it into something similar within a given time window which might be the correction of a potential typo. To the best of our knowledge, the idea of collecting query corrections using user session and time information is novel. Previous work suggested collecting queries using information that a user clicked on a proposed correction. Our proposed method for collecting training data has several advantages. First, we do not rely on a previous spelling correction system, but on user formulations. Second, for many search queries, especially from the tail where search recall is generally low, these misspellings yield few results, and thus, users looking for certain products are inclined to correct the query themselves in order to find what they are looking for. We use Damerau-Levenshtein distance (Damerau, 1964) on character level as similarity criterion, i.e. queries within a specific edit distance are considered to be related.

The steps proposed in this work are:

1. Extraction of similar user queries from search logs for bootstrapping training data (Section 2),
2. classification and filtering of data to remove noisy entries (Section 3), and

[*] The author is now affiliated with Lilt Inc., Stanford, CA, USA.

[†] The author is now affiliated with Stylight GmbH, Munich, Germany.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 451–460, Lisbon, Portugal, 17-21 September 2015.

3. spelling correction cast into a statistical machine translation framework based on character bigram sequences (Section 4).

We also evaluate the work thoroughly in Section 5 where we compare our method to three other online sites, two of them from the e-commerce domain, and present a novel approach that determines quality based on retrieved search results. We show examples indicating that our method can handle both corrections of misspelled queries and queries with segmentation issues (i.e. missing whitespace delimiters). A summary can be found in Section 6.

To our knowledge, this is the first work that uses character-based machine translation technology on user-generated data for spelling correction. Moreover, it is the first to evaluate the performance in an e-commerce setting with there relevant measures.

1.1 Related Work

One of the more prominent papers on autocorrection of misspelled input is (Whitelaw et al., 2009). The three-step approach incorporates a classification step that determines whether a word is misspelled, computes the most likely correction candidate and then, again, classifies whether this candidate is likely to be correct. An error model based after (Brill and Moore, 2000) is trained on similar word pairs extracted from large amounts of web sites, and a language model is used to disambiguate correction candidates based on left and right context around the current position. Our method differs in several ways. First, we consider full query pairs as training data, and do not use single words as primary mode of operation. Second, we do not train explicit error models $P(w|s)$ for words w and observed corrections s, but use standard phrase-based machine translation modeling to derive phrase and lexical translation models. Although our level of context is shorter, especially for long words, the system automatically uses cross-word level context.

The idea of using consecutive user queries from a stream of events to improve ranking of web search results was described in (Radlinski and Joachims, 2005). The authors introduce the notion of *query chains* that take advantage of users reformulating their queries as a means to learn better ranking functions. The classification of query chains is performed by support vector machines, and its training data is generated in a supervised fashion by manual inspection and annotation. In contrast, we do not manually annotate any of our training data. Since our initial sets are quite noisy, we apply a couple of heuristics that try to produce a cleaner subset of the data that contains mostly misspelled queries and their potential correction candidates.

Another focus of researchers was specifically to tackle misspelled web search queries and use search engine logs for training and evaluation data (Gao et al., 2010), which differs from our work by collecting data using "click-through" enforcement. The user is presented with a spelling correction, and if she clicks on it, they learn that the correction is valid. Our method does not need a previous spelling correction to work. In addition, our proposed method has the potential to learn corrections of new and rare terms that will not be produced by an automatic spelling correction.

In (Zhang et al., 2006), the authors use a conventional spellchecker to correct web queries through additional reranking of its output by a ranking SVM. The training data is in part automatically extracted, but also contains manually annotated pairs and, thus, is a semi-supervised approach. In this paper, we use an unsupervised approach to generate training data. Query spelling correction that is based on click-through data and uses a phrase-based error model is reported in (Sun et al., 2010). Our models operate on character sequences instead of words, and we do not observe issues with identity transformations (i.e. non-corrections for correctly spelled input).

In (Cucerzan and Brill, 2004), the authors investigate a transformation method that corrects unlikely queries into more likely variants based on web query logs. The iterative approach transforms a search query based on word uni- and bigram decompositions, and the authors evaluate on both a large set that contains around 17% misspelled queries and a smaller set that is based on successive user-reformulated similar queries, a similar setup that we use to extract our training data. They stress the importance of a good language model, as performance drops drastically going from a bigram to a unigram LM.

The use of character-based models in combination with statistical machine translation is not novel and was proposed for spelling correction, e.g., in (Formiga and Fonollosa, 2012), (Liu et al.,

2013) or (Chiu et al., 2013). The authors compare a distance-based approach including a language model, a confusion network-based approach, a translation approach through a heuristically defined phrase table coding all character transformations, and a character-based machine translation approach using standard procedures (automatic word alignment and phrase extraction). The training data is manually created in contrast to our work where we automatically bootstrap training data from query logs. Research has also been done for translation of closely related languages (e.g. (Vilar et al., 2007) and (Nakov and Tiedemann, 2012)) and transliteration (e.g. (Deselaers et al., 2009)). An early summary paper with various spelling-related problems (non-word error detection, isolated-word error correction, and context-dependent word correction) can be found in (Kukich, 1992).

2 Extraction of training data

We use event logs that track user interactions on an e-commerce web site. Each action of a user visiting the site is stored in a data warehouse and HDFS (Hadoop Distributed File System). We store several billion of these user records on a daily basis. The setup allows us to efficiently process large amounts of data points using a Map-Reduce framework via Hadoop[1]. As part of user event tracking, all interactions on the site are stored in the database, e.g. which search terms were entered, which links were clicked, and what actions resulted in this (i.e. buying an item, adding it to a like or watch list, or simply advancing to the next page of search results, and so on). We focus on search terms that users enter within a given period of time.

Our hypothesis is that users that enter consecutive search terms are not satisfied with the results and try to modify the query until the results are acceptable. We then analyze the sequence of search queries as part of each user session. We only extract consecutive search queries that are similar in terms of character-level Damerau-Levenshtein edit distance which is the minimum number of character insertions, deletions, substitutions and transpositions (i.e. the swapping of two characters). Note that spelling correction in this kind of environment also needs to address segmentation of queries in case of missing whitespace. It is quite common to find query terms being concatenated without the use of a space character, e.g. *calvinklein*, *ipadair* or *xboxone*. Also, search queries are short in nature and often lack context, and particularly for the e-commerce domain largely consist of brand and product names and associated attributes (e.g. size of clothing).

Our method extracts similar search query pairs that will be used in our statistical machine translation setup as training data. Table 1 shows examples that we extract from the event logs. We use edit distance thresholds of 3 and 5 characters, where the latter is generally noisier. Noise in this context is everything that is not related to misspelled queries. After a closer look, we observe that many queries are simply rewrites, i.e., users either make refinements to their original query by adding more words to narrow down results, e.g., *leather wallet* → *leather wallet men*, deleting words to decrease specificity, e.g., *gucci belt* → *gucci*, or simply going through various related products or attributes, e.g., *iphone 5* → *iphone 5s* → *iphone 6* where they progress through different products or *nike air 9* → *nike air 9.5* → *nike air 10* where they iterate through different sizes.

The logs on HDFS are organized as sessions where each session contains a stream of user events up to a specific time of inactivity. We use event timestamps to determine how long the users need between consecutive queries, and discard similar query pairs if they are above a threshold of 20 seconds. We use Hadoop-based mapper and reducer steps for the data extraction procedure. In the mappers, pairs of similar user search queries get emitted as per above edit distance criterion on character level, whereas the reducer simply accumulates all counts for identical search query pairs. Due to the size of the data, we run the Hadoop extraction jobs on 24-hour log portions, thus obtaining separate data sets for each day. Overall, we can extract several hundred thousand to several million similar query pairs on a daily basis for edit distance thresholds of 3 and 5, respectively.

We pull several months of data from the Hadoop logs and accumulate each daily pull with unique entries for training our spelling correction system. As mentioned above, search queries that are similar but where the original query is not misspelled make up a big portion of the extracted data. The following section focuses on how to filter the data to result in containing mostly query pairs that fit

[1] http://hadoop.apache.org

Search query	Similar consecutive query	Edit distance	User correction?
nike air hurache	nike air huarache	1	Yes
jordan size 9	jordan size 9.5	2	No
galaxy s4	galaxy s5	1	No
pawer cord forplaystation 3	power cord for playstation 3	2	Yes
iphine 6	iphone 6	1	Yes
iphone 6 plus	iphone 6 plus case	5	No
micheal korrs wstches	michael kors watches	3	Yes
calvin klien men boit	calvin klein men boot	2	Yes
boots	boots men	4	No
sueter	sweater	2	Yes

Table 1: Extracted query pairs found in user event logs. Labels in column 4 indicate whether user-initiated spelling correction (Yes) has taken place vs. a search reformulation that entails the source query not being misspelled (No).

our use case, i.e. mappings from misspelled to corrected ones.

3 Filtering non-misspelled input

We tested a heuristic approach to filtering search query pairs: a combination of regular expressions and thresholded feature scores that detect search progressions and refinements which should be removed from the training data as they do not represent valid user corrections.

The manual filtering heuristic calculates a sequence of features for each search query pair, and as soon as a feature fires, the entry is removed. In the following, we will use the notation (x, y) for a search query pair that is extracted from the logs explained by our method in the previous section. Query x is a user search query, and query y is a similar query issued by the user in the same session within a specific time window. Example query pairs are (*babydoll*, *baby dolls*), (*bike wherl*, *bike wheel*) or (*size 12 yellow dress*, *size 14 yellow dress*). We use the following features as part of this process:

Regular expressions. We remove search query pairs (x, y) if y is a rewrite of x using search-related operators, e.g. $y =$"x" which adds quotation marks around the query (and, thus, has an edit distance of 2). Example: (*bluetooth speakers*, *"bluetooth speakers"*).

LM ratio. We use an English language model trained on publicly available corpora (e.g. Europarl), frequent queries and web content from the e-commerce domain to calculate log-likelihoods for each query and filter entries if the ratio is above zero, i.e. $\log p(x/y) = \log p(x) - \log p(y) > 0$. This step essentially removes query pairs (x, y) if the log-likelihood of query y is smaller than x which usually indicates that the correction is more perplexing than the original query. Example: (*bluetooth ear phones*, *bluetooth ear hpones*) with a log-likelihood ratio of -7.31 + 16.43 > 0 is removed as a typo actually appears on the "corrected" side.

Edit operations. We use a simple heuristic that detects search refinements in terms of word insertions and deletions: a word-level edit distance criterion is used to remove entries where edit operations indicate insertions or deletions of full words. We also detect substitutions on number tokens which are also excluded from training data. Examples: (*polo shirt*, *polo shirt xl*), (*nikon d700*, *nikon d7100*).

Frequent terms. We look at queries (x, y) and use a vocabulary with relative frequencies based on the query data y to determine whether substitutions on word level change a frequent token into another frequent token. Examples: (*snake bat wooden*, *snake bat wood*), (*hd dvds*, *hd dvd*) where *wood*/*wooden* and *dvds*/*dvd* are both frequent tokens and, thus, most likely rewrites and not corrections.

Language ID. The primary search language on US sites is English, but there is also a non-negligible mix of other languages, Spanish in particular. We do not remove all queries that are not identified as English because language identification on (often short) search queries is a non-trivial

Method	#queries	#tokens	MT Acc
all data	80.5M	235.6M	62.0%
heuristic filter	12.6M	40.1M	65.5%

Table 2: Filtering training data with a heuristic set of features. Accuracies are given on DEV for MT baselines with differences only in the data setup.

task with a high error rate and filtering for only English would remove a lot of valid entries. We simply remove query pairs where x is identified as either Spanish or Unknown based on Google's Compact Language Detector[2]. Example: (*accesorios cuarto*, *accesorios de cuarto*).

These heuristics help us to reduce the training data size from 80 million noisy search query pairs with around 235 million tokens to 12.5 million query pairs with roughly 40 million tokens that are of higher quality and most likely spelling corrections.

Table 2 shows results of this filtering step in terms of data sizes and the accuracy on the dev set (cf. Section 4). We observe that the filtering scheme reduces overall training size by almost 85% and increases accuracy on the development set by 3.5% absolute. The removal of query pairs is very aggressive at this point, and overall quality of the spelling correction framework might benefit from a more careful selection. We will look into improved variants of filtering through a maximum entropy classifier trained on actual search results in the future.

4 Autocorrection framework

We cast the autocorrection task into character-based statistical machine translation. For this, we prepare the data by splitting words into sequences of lowercased characters and use a special character to mark whitespace that indicates word boundaries. Once these sequences of characters are corrected, i.e. translated, they are merged back to the full word forms. Table 3 shows an example search query being preprocessed, translated and postprocessed. We use bigram characters instead of single characters, as suggested in (Tiedemann, 2012), in order to improve the statistical alignment models and make them more expressive.

For training the autocorrection system we use basic methods and open-source tools for statistical machine translation. The character alignment is obtained by using GIZA++ (Och and Ney, 2003) for 4, 3 and 2 iterations of IBM Model 1, HMM, and IBM Model 3, respectively. As opposed to the standard machine translation task, we did not observe improvements from IBM Model 4 and do not use it as part of the alignment process.

We use Moses (Koehn et al., 2007) for standard phrase extraction, building KenLM language models (Heafield, 2011) and tuning. The standard set of features is used, including a phrase model, word lexicon model, length penalty, jump penalty and a language model. The model weights are optimized using MERT (Och, 2003). The Moses framework allows us to easily conduct experiments with several settings and find the optimal one for our task at hand. We experiment with varying context sizes for phrase table and language model, additional features and different scoring methods, e.g. BLEU (Papineni et al., 2002) in comparison to directly maximizing accuracy on the dev set. A more detailed description of those experiments including results will follow in Section 5.

Evaluation data. In order to evaluate our proposed framework, we extracted 10,000 query pairs from a one week period not part of the training data. The initial size of 3.5M query pairs was reduced to 10k by exponential reservoir sampling (Osborne et al., 2014) after sorting by frequency. The result is a set of representative query pairs that focuses more on frequent misspellings, but also contains rare queries from the tail of the distribution.

We asked humans to create a gold reference annotation that we can use for tuning and evaluation purposes. The guidelines were to check whether for a search query pair (x, y), the left query x is misspelled, and if so, whether the similar candidate y is a correct correction or else provide the most likely correction. If x was not misspelled, the guidelines instructed to propagate query x to the gold reference y, i.e. for those cases, we have identity or a true negative. If query x is not English, the annotators had to mark those entries and we removed them from the set. This step affected 798 queries (i.e. around 8%, mostly Spanish), and the final set contains 9202 query pairs. We split those into half to produce 4,600 queries for dev and 4,602 for test. The true negatives in those sets (i.e. entries that do not need to be corrected) are ~15%. We are aware that this approach focuses on

[2]https://code.google.com/p/cld2

	characters	character bigrams
Original:	hollowin custome	
Preprocessing:	h o l l o w i n S c u s t o m e	ho ol ll lo ow wi in nS Sc cu st to om me
Translation:	h a l l o w e e n S c o s t u m e	ha al ll lo ow we ee en nS Sc co st tu um me
Postprocessing:	halloween costume	

Table 3: Preprocessing of input data as single character or character bigram sequences. "S" denotes whitespace. After translation, postprocessing transforms back to word-level surface forms.

Eval data	DEV	TEST
#queries	4,600	4,602
#tokens	15,593	15,557
CER[%]	6.1	6.0
SER[%]	84.2	84.3

Table 4: Statistics on dev and test portions of the post-edited evaluation data. CER is the character error rate of the misspelled queries against the gold reference, SER is the sentence (i.e. here query-level) error rate of the sets.

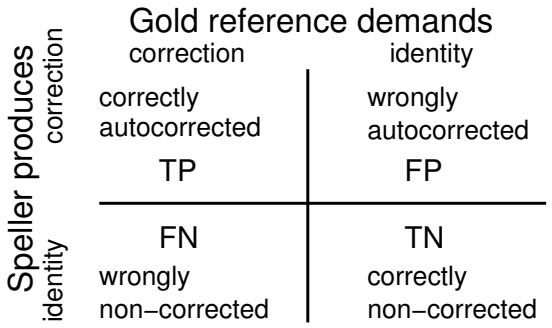

Figure 1: Scoring schema.

recall over precision, as the majority of queries is usually not misspelled. Nevertheless, exponential reservoir sampling helps us to focus on the head of that distribution, and our main goal is to evaluate the capabilities of the spelling correction framework, not the overall system integration. A final investigation in combination with query expansion that will be evaluated in the context of the live site is left for future work. Detailed statistics on the two sets can be found in Table 4.

Additional training data. In addition to the parallel data pull from the query logs as described in Section 2, we also take the top queries from those logs where we are sure they are most likely correct and can be used as gold reference (e.g. search queries like *handbags* or *wedding dress* appear thousands of times), and generate artificially noisified variants based on an English keyboard layout and statistics derived from our development set. On the dev set, we calculated a character error rate of roughly 6%, and this rate is used as a mutation rate for the artificially introduced noise. We also determine the following edit operation rates based on the dev set: 6% character transpositions, 18% for deletions, 33% for insertions, and 43% for substitutions. The target character for substitutions and insertions is based on a random Gaussian distribution with a distance mean 1 and standard deviation of 1 around the selected character, i.e. we target neighboring keys on the keyboard. For the query *wedding dress*, e.g., this method introduces misspelled variants such as *weddig dress*, *weedimg dress*, or *weddinb dreess*. We run this method 10 times on a set of 688k queries and remove all duplicates, resulting in additional 4.6M search query pairs with around 13M tokens for the training data.

As a final step, we add frequent queries from user event logs but also product titles of the live inventory to the language model. In total, the monolingual portion of the data contains roughly 31M entries with 319M tokens which are added to the target side of the bilingual training portion.

5 Experiments

In this section, we report on a series of hillclimbing experiments used to optimize performance. In the following, we use standard information retrieval criteria for evaluation, namely accuracy, precision, recall, and F1 score. Figure 1 depicts a natural way of how the scoring is performed. TP denotes true positives, FN false negatives, FP false positives, and TN true negatives, respectively. Note that for the case where the gold reference demands a correction, and the speller produces a wrong correction different from the source query, we have to increase both false positives FP and false negatives FN. With this, we can do stan-

TEST [%]	Acc	Prec	Rec	F1
online A	68.8	**84.0**	64.2	72.8
online B	63.0	77.4	60.2	67.7
online C	56.3	58.6	60.7	59.7
MT (this work)	**70.6**	74.0	**72.3**	**73.1**
MT (alt. setup)	70.2	77.1	69.3	73.0

Table 5: Comparison of accuracy, precision, recall and F1 score against other online sites on the test set.

dard calculation of accuracy as $(TP + TN)/(P + N)$ with $P = TP + FN$ and $N = TN + FP$, precision as $TP/(TP + FP)$, recall as TP/P, and F1 score as $2TP/(2TP + FP + FN)$.

5.1 Hillclimbing and performance

We compare the performance of our autocorrection system with that of three other online search engines, both from the e-commerce as well as web search domain. For this, we automated entering our search terms on the corresponding sites and extracted candidates from "Did you mean..." suggestions or "Showing results for..." autocorrections. Table 5 shows that our method outperforms all other systems in accuracy, recall and F1 score. We argue that recall in our setup is more important than precision because the goal is to correct as much as possible as autocorrection is usually invoked by the search backend as a result of low or *null* recall size.

Gradual improvements were made to the system setup and we track those on the development set. Table 6 gives a couple of major waypoints in the hillclimbing process. The baseline incorporates all extracted training data and uses phrases up to length 3 (i.e. up to three character bigrams for both source and target side). The baseline language model uses 6-grams. This setup is trained on very noisy data that contains a lot of search refinements that are not actual misspellings. The filtered data improves results. In general, we observe a 3-4% relative improvement across all scores when using bigrams instead of single characters. We experiment with additional phrase features as part of the phrase table and add 5 features based on phrase pair-specific edit operations, i.e. the number of insertions, deletions, substitutions, transpositions and final overall edit distance, which helps to increase precision. The artificially noisified data gives additional small gains, as well as direct optimization of accuracy instead of BLEU or WER. We did not observe significant differences when tuning on BLEU versus WER. Most of the improvement though comes from increasing context size, i.e., 10-gram language models and lengths up to 5 bigram character spans for both source and target phrases. We also observe that iterative correction, i.e. running the speller a second time over already corrected data, further improves performance slightly which is in-line with findings in (Cucerzan and Brill, 2004) and (Gubanov et al., 2014).

Increasing precision. We also investigated a system setup that focuses on precision over recall in order to be more in sync with the online systems that have been most likely optimized to a more cautious correction mode. Our previous experiments prefer recall which is due to the 85:15 split of misspelled vs. correct queries in the dev set. For a more conservative mode that focuses on precision, we updated the dev set by automatically adding "mostly-correct" queries with identity as correction candidate. For this, we extracted the most frequent queries with a high number of search results which can be deemed to be "almost" correct. The dev size increased to roughly 22k queries with an approximate split of 85:15 for "correct" vs. misspelled. This is a more realistic setting if the correction is applied to *all* incoming queries, irrespective of the number of corresponding search results (note also that this mode is different from our initial one where we apply corrections only to queries with low or *null* results). We also found that tuning on Matthews Correlation Coefficient (Matthews, 1975) balances better precision versus recall, especially for unbalanced classes which is the case here (i.e. 85:15 split).

In the last line of Table 6 we added more data extracted over several additional months. The additional data amounts to 60M queries, therefore increasing the total training size to 72M queries. This final setup, as described, improves precision but hurts recall slightly. Overall, the F1 and accuracy measures are still improved which is most likely due to the additional training data.

Finally, we ran a large-scale experiment and extracted the 100k most frequent unique queries (which account for 7.8M queries based on their actual frequencies). Since they represent the most common queries, some of them typed thousands of times by the users, they are deemed to be

DEV [%]	Acc	Prec	Rec	F1
baseline	62.0	66.2	61.0	63.5
+filtered data	65.5	68.8	67.6	68.2
+phrase length 5	67.2	70.0	69.6	69.8
+phrase features	67.3	71.2	68.3	69.8
+artificial noise	67.8	71.4	68.3	69.8
+10-gram LM	68.7	71.9	70.5	71.2
+tune on Acc	68.9	72.1	70.3	71.2
+iterative 2nd run	69.0	72.2	70.5	71.4
+alt. setup (Prec)	69.5	75.4	68.5	71.8

Table 6: Hillclimbing on the dev set.

Query	Category distribution
moto x	50% cell phones, 30% automotive, 5% clothing, ...
tory burch fitbit	75% jewelry, 20% sports, ...
madden 15	65% games, 18% toys, 7% collectibles, ...

Table 7: Item category distributions for queries.

"mostly" correct. We autocorrect the set through our best system which results in changed queries in <0.5% of the cases after manual filtering of non-errors (e.g. segmentation differences that are handled separately by our query expansion system, e.g. *rayban* → *ray ban* which are equivalent in the search backend) and when compared against the original input set. This means that most of the queries are left unchanged and that the decoding process of the SMT system does a good job in classifying whether a correction is needed or not. Manual inspection of the most frequent differences shows that changes actually happen on frequently misspelled words which are part of those top 100k queries, e.g. *micheal* → *michael*, *infared* → *infrared*, or *accesories* → *accessories*.

5.2 Search evaluation

An interesting way to evaluate the performance of the autocorrection system is to look at it from a search perspective. If the goal is to improve user experience in search and retrieve more relevant results, we need to compare the search result sets in terms of quantity and quality. For quantity we are mainly interested in how many queries lead to 0 search results before and after autocorrection. Table 8 shows these numbers for our test set. We improve the rate of *null* results by 51.8% absolute. In a deeper analysis we see that out of the 81.6% in the source set we improve the query behavior for 63.8% queries after autocorrection while we only decrease from non-*null* to *null* for 1.6%. A manual inspection of the 1.6% shows that those queries already led to very low search results (<10) even for the source.

TEST [%]	*null* results	KL div. < 1
misspelled	81.6	40.9
gold corrections	27.1	100.0
online A	36.5	84.4
online B	36.0	82.1
online C	33.0	76.4
MT (this work)	**29.8**	**86.0**

Table 8: Search results evaluation on the test set. For *null* results, lower rates are better. For KL div. evaluation, higher rates are better, as they indicate a closer match with the category distribution from the gold corrections.

For quality, we compare the search result sets for the autocorrected queries with those of the reference and get an estimate of how similar each autocorrected query behaves to the expected behavior given by its gold correction. In particular, we extract the categories from all items which are returned as part of the search API calls (see Table 7). We use Kullback-Leibler divergence (Kullback and Leibler, 1951) to compute the difference between the category distributions. The KL divergence defines the information lost when using one distribution to approximate another:

$$D(P||Q) = \sum_i P(i) \cdot \ln\left(\frac{P(i)}{Q(i)}\right)$$

In our case, we use the category distribution of the results of the reference query as the observed data P which we want to model with the category distribution Q from the results of the autocorrected query. This gives us a more realistic evaluation of the method, as it is more relevant to the retrieval problem that autocorrection is trying to address. The queries *perfumes for women* and *perfume for women*, e.g., retrieve similar amounts and types of items although their strings are different (note the plural *s* in the first one) which is penalized when computing accuracy on string level.

In order to deal with zero-probability events, we smooth both distributions such that they contain exactly the same categories. For this we add categories that are present in one distribution but not

Misspelled query	Correction
adidda whatch blooto	adidas watch bluetooth
awrppstale buttin shirt	aeropostale button shirt
bangen olafsn headset	bang olufsen headset
camra exesers	camera accessory
crystal and righston cuf ring	crystal and rhinestone cuff ring
fauxfurmidcalfwesternboots	faux fur mid calf western boots
fotbool chus	football shoes
otherbooxiphon5	otterbox iphone 5
womens realhairsaltandpeper	womens real hair salt and pepper

Table 9: Examples of misspelled user search queries that the presented system is able to correct.

the other with a very small probability which we subtract from the highest probability in the distribution so that all probabilities still sum up to 1. In the case of getting 0 search results for either the autocorrected or reference query we cannot compute the KL divergence and we simply mark those cases with a high number. If both queries return 0 search results we return a distance of 0. Note that with this we do not penalize queries that are misspelled and should have been corrected even if they still retrieve 0 items in search after correction. However, in this test we are only interested in the search results we get and not in the correction itself.

Table 8 shows that the presented method is closest in terms of KL divergence to category distributions of search results based on the gold references. Even though this is a nice and easy way to indicate quality of search results, we do not claim this method to be an in-depth analysis of relevance of search results and leave this for future work.

5.3 Examples

The examples in Table 9 demonstrate the abilities of the presented spelling correction framework. We are able to handle extremely misspelled queries, e.g. due to phonetic spelling by possibly non-native speakers (*camra* → *camera*, *chus* → *shoes*), words being glued together due to missing whitespace (*fauxfurmid...* → *faux fur mid...*), or brand name confusions (*righston* → *rhinestone*).

6 Conclusion

In this paper, we presented a powerful spelling correction system for the e-commerce domain using statistical phrase-based machine translation based on character bigram sequences. We extracted training data from event logs where users issue similar search queries within a certain period of time. Filtering through various heuristics is used to clean the initially noisy data set and remove entries not related to spelling errors. We evaluated our system against established online search sites, both in the general and e-commerce domain, and showed favorable results in terms of recall and retrieval rates.

We plan to further invest in improving precision. For this, we feel that adding full word-level models will help to overcome the somewhat limited context present in our character sequence-based models. We also plan to investigate the prototype directly in query expansion as part of the search backend.

Acknowledgments

The authors would like to thank the Localization team at eBay for creating gold corrections for the evaluation sets, and members of the HLT and Search teams for fruitful discussions as well as reading early drafts of this work and giving valuable feedback.

References

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong.

Hsun-wen Chiu, Jian-cheng Wu, and Jason S. Chang. 2013. Chinese spelling checker based on statistical machine translation. In *Proc. of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 49–53, Nagoya, Japan.

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective

knowledge of web users. In *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 293–300, Barcelona, Spain.

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

Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. 2009. A deep learning approach to machine transliteration. In *Fourth EACL Workshop on Statistical Machine Translation*, pages 233–241, Athens, Greece.

Lluís Formiga and José A. R. Fonollosa. 2012. Dealing with input noise in statistical machine translation. In *Proc. of the 24th International Conference on Computational Linguistics (Coling 2012): Posters*, pages 319–328, Mumbai, India.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proc. of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366, Beijing, China.

Sergey Gubanov, Irina Galinskaya, and Alexey Baytin. 2014. Improved iterative correction for distant spelling errors. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 168–173, Baltimore, MD.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proc. of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, UK.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics: Demos and Posters*, pages 177–180, Prague, Czech Republic.

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

Solomon Kullback and Richard Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid Chinese spelling correction using language model and statistical machine translation with reranking. In *Proc. of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan.

Brian W. Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 301–305, Jeju Island, Korea.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Miles Osborne, Ashwin Lall, and Benjamin Van Durme. 2014. Exponential reservoir sampling for streaming language models. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 687–692, Baltimore, Maryland.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Filip Radlinski and Thorsten Joachims. 2005. Query chains: Learning to rank from implicit feedback. In *Proc. of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 239–248, Chicago, IL.

Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 266–274, Uppsala, Sweden.

Jörg Tiedemann. 2012. Character-based pivot translation for under-resourced languages and domains. In *Proc. of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 141–151, Avignon, France.

David Vilar, Jan-Thorsten Peter, and Hermann Ney. 2007. Can we translate letters? In *Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Ged Ellis. 2009. Using the Web for language independent spellchecking and autocorrection. In *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899, Singapore.

Yang Zhang, Pilian He, Wei Xiang, and Mu Li. 2006. Discriminative reranking for spelling correction. In *Proc. of the 20th Pacific Asia Conference on Language, Information and Computation*, pages 64–71, Wuhan, China.