FASPell: A Fast, Adaptable, Simple, Powerful Chinese Spell Checker Based On DAE-Decoder Paradigm

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, Junhui Liu
Intelligent Platform Division, iQIYI, Inc.
{hongyuzhong, yuxianguo, heneng, liunan, liujunhui}@qiyi.com

Abstract

We propose a Chinese spell checker – **FASP**ell based on a new paradigm which consists of a denoising autoencoder (DAE) and a decoder. In comparison with previous state-of-the-art models, the new paradigm allows our spell checker to be **F**aster in computation, readily **A**daptable to both simplified and traditional Chinese texts produced by either humans or machines, and to require much **S**impler structure to be as much **P**owerful in both error detection and correction. These four achievements are made possible because the new paradigm circumvents two bottlenecks. First, the DAE **curtails the amount of Chinese spell checking data needed for supervised learning (to <10k sentences)** by leveraging the power of unsupervisedly pre-trained masked language model as in BERT, XLNet, MASS etc. Second, the decoder helps to **eliminate the use of confusion set** that is deficient in flexibility and sufficiency of utilizing the salient feature of Chinese character similarity.

1 Introduction

There has been a long line of research on detecting and correcting spelling errors in Chinese texts since some trailblazing work in the early 1990s (Shih et al., 1992; Chang, 1995). However, despite the spelling errors being reduced to substitution errors in most researches[1] and efforts of multiple recent shared tasks (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015; Fung et al., 2017), Chinese spell checking remains a difficult task. Moreover, the methods for languages like English can hardly be directly used for the Chinese language because there are no delimiters between words, whose lack of morphological variations makes the syntactic and semantic interpretations of any Chinese character highly dependent on its context.

[1] Likewise, this paper only covers substitution errors.

1.1 Related work and bottlenecks

Almost all previous Chinese spell checking models deploy a common paradigm where a fixed set of similar characters of each Chinese character (called *confusion set*) is used as candidates, and a filter selects the best candidates as substitutions for a given sentence. This naive design is subjected to two major bottlenecks, whose negative impact has been unsuccessfully mitigated:

- **overfitting to under-resourced Chinese spell checking data**. Since Chinese spell checking data require tedious professional manual work, they have always been under-resourced. To prevent the filter from overfitting, Wang et al. (2018) propose an automatic method to generate pseudo spell checking data. However, the precision of their spell checking model ceases to improve when the generated data reaches 40k sentences. Zhao et al. (2017) use an extensive amount of ad hoc linguistic rules to filter candidates, only to achieve worse performance than ours even though our model does not leverage any linguistic knowledge.

- **inflexibility and insufficiency of confusion set in utilizing character similarity**. The feature of Chinese character similarity is very salient as it is related to the main cause of spelling errors (see subsection 2.2). However, the idea of confusion set is troublesome in utilizing it:

    1. *inflexibility* to address the issue that confusing characters in one scenario may not be confusing in another. The difference between simplified and traditional Chinese shown in Table 1 is an example. Wang et al. (2018) also suggest that confusing characters for ma-

Proceedings of the 2019 EMNLP Workshop W-NUT: The 5th Workshop on Noisy User-generated Text, pages 160–169
Hong Kong, Nov 4, 2019.

chines are different from those for humans. Therefore, in practice, it is very likely that the correct candidates for substitution do not exist in a given confusion set, which harms recall. Also, considering more similar characters to preserve recall will risk lowering precision.

2. *insufficiency* in utilizing character similarity. Since a cut-off threshold of quantified character similarity (Liu et al., 2010; Wang et al., 2018) is used to produce the confusion set, similar characters are actually treated indiscriminately in terms of their similarity. This means the information of character similarity is not sufficiently utilized. To compensate this, Zhang et al. (2015) propose a spell checker that has to consider many less salient features such as word segmentation, which add more unnecessary noises to their model.

1.2 Motivation and contributions

The motivation of this paper is to circumvent the two bottlenecks in subsection 1.1 by changing the paradigm for Chinese spell checking.

As a major contribution and as exemplified by our proposed Chinese spell checking model in Figure 1, the most general form of the new paradigm consists of a denoising autoencoder[2] (DAE) and a decoder. To prove that it is indeed a novel contribution, we compare it with two similar paradigms and show their differences as follows:

1. Similar to the old paradigm used in previous Chinese spell checking models, a model under the DAE-decoder paradigm also produces candidates (by DAE) and then filters the candidates (by the decoder). However, candidates are produced on the fly based on contexts. If the DAE is powerful enough, we should expect that all contextually suitable candidates are recalled, which prevent the inflexibility issue caused by using confusion set. The DAE will also prevent the overfitting issue because it can be trained unsupervisedly using a large number of natural texts. Moreover, character similarity can be used by the decoder without losing any information.

2. The DAE-decoder paradigm is *sequence-to-sequence*, which makes it resemble the encoder-decoder paradigm in tasks like machine translation, grammar checking, etc. However, in the encoder-decoder paradigm, the encoder extracts semantic information, and the decoder generates texts that embody the information. In contrast, in the DAE-decoder paradigm, the DAE provides candidates to reconstruct texts from the corrupted ones based on contextual feature, and the decoder[3] selects the best candidates by incorporating other features.

Besides the new paradigm per se, there are two additional contributions in our proposed Chinese spell checking model:

- we propose a more precise quantification method of character similarity than the ones proposed by Liu et al. (2010) and Wang et al. (2018) (see subsection 2.2);

- we propose an empirically effective decoder to filter candidates under the principle of getting the highest possible precision with minimal harm to recall (see subsection 2.3).

1.3 Achievements

Thanks to our contributions mentioned in subsection 1.2, our model can be characterized by the following achievements relative to previous state-of-the-art models, and thus is named **FASP**ell.

- Our model is **F**ast. It is shown (subsection 3.3) to be faster in filtering than previous state-of-the-art models either in terms of absolute time consumption or time complexity.

- Our model is **A**daptable. To demonstrate this, we test it on texts from different scenarios – texts by humans, such as learners of Chinese as a Foreign Language (CFL), and by machines, such as Optical Character Recognition (OCR). It can also be applied to both simplified Chinese and traditional Chinese, despite the challenging issue that some erroneous usages of characters in traditional texts are considered valid usages in simplified texts (see Table 1). To the best of our knowledge, all previous state-of-the-art models only focus on human errors in traditional Chinese texts.

[2] the term *denoising autoencoder* follows the same sense used by Yang et al. (2019), which is arguably more general than the one used by Vincent et al. (2008).

[3] The term *decoder* here is analogous as in *Viterbi decoder* in the sense of finding the best path along candidates.

Table 1: Examples on the left are considered valid usages in simplified Chinese (SC). Notes on the right are about how they are erroneous in traditional Chinese (TC) and suggested corrections. This inconsistency is because multiple traditional characters were merged into identical characters in the simplification process. Our model makes corrections for this type of errors only in traditional texts. In simplified texts, they are not detected as errors.

SC Examples	Notes on TC usage
周末 (weekend)	周 → 週　周 only in 周到, etc.
旅游 (trip)	游 → 遊　游 only in 游泳, etc.
制造 (make)	制 → 製　制 only in 制度, etc.

- Our model is **S**imple. As shown in Figure 1, it has only a masked language model and a filter as opposed to multiple feature-producing models and filters being used in previous state-of-the-art proposals. Moreover, only a small training set and a set of visual and phonological features of characters are required in our model. No extra data are necessary, including confusion set. This makes our model even simpler.

- Our model is **P**owerful. On benchmark data sets, it achieves similar F1 performances (subsection 3.2) to those of previous state-of-the-art models on both detection and correction level. It also achieves arguably high precision (78.5% in detection and 73.4% in correction) on our OCR data set.

2 FASPell

As shown in Figure 1, our model uses masked language model (see subsection 2.1) as the DAE to produce candidates and confidence-similarity decoder (see subsection 2.2 and 2.3) to filter candidates. In practice, doing several rounds of the whole process is also proven to be helpful (subsection 3.4).

2.1 Masked language model

Masked language model (MLM) guesses the tokens that are masked in a tokenized sentence. It is intuitive to use MLM as the DAE to detect and correct Chinese spelling errors because it is in line with the task of Chinese spell checking. In the original training process of MLM in BERT (Devlin et al., 2018), the errors are the random masks, which are the special token `[MASK]` 80% of the time, a random token in the vocabulary 10% of the time and the original token 10% of the time. In cases where a random token is used as the mask, the model actually learns how to correct an erroneous character; in cases where the original tokens are kept, the model actually learns how to detect if a character is erroneous or not. For simplicity purposes, FASPell adopts the architecture of MLM as in BERT (Devlin et al., 2018). Recent variants – XLNet (Yang et al., 2019), MASS (Song et al., 2019) have more complex architectures of MLM, but they are also suitable.

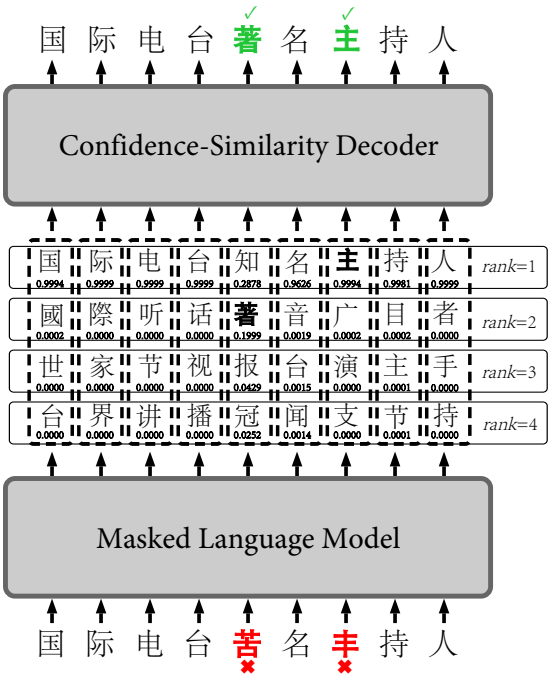

Figure 1: A real example of how an erroneous sentence which is supposed to have the meaning of "*A famous international radio broadcaster*" is successfully spell-checked with two erroneous characters 苦 and 丰 being detected and corrected using FASPell. Note that with our proposed confidence-similarity decoder, the final choice for substitution is not necessarily the candidate ranked the first.

However, just using a pre-trained MLM raises the issue that the errors introduced by random masks may be very different from the actual errors in spell checking data. Therefore, we propose the following method to fine-tune the MLM on spell checking training sets:

- For texts that have no errors, we follow the original training process as in BERT;

- For texts that have errors, we create two types of training examples by:

1. given a sentence, we mask the erroneous tokens with themselves and set their target labels as their corresponding correct characters;
2. to prevent overfitting, we also mask tokens that are not erroneous with themselves and set their target labels as themselves, too.

The two types of training examples are balanced to have roughly similar quantity.

Fine-tuning a pre-trained MLM is proven to be very effective in many downstream tasks (Devlin et al., 2018; Yang et al., 2019; Song et al., 2019), so one would argue that this is where the power of FASPell mainly comes from. However, we would like to emphasize that the power of FASPell should not be biasedly attributed to MLM. In fact, we show in our ablation studies (subsection 3.2) that MLM itself can only serve as a very *weak* Chinese spell checker (its performance can be as poor as F1 being only 28.9%), and the decoder that utilizes character similarity (see subsection 2.2 and 2.3) is also significantly indispensable to producing a *strong* Chinese spell checker.

2.2 Character similarity

Erroneous characters in Chinese texts by humans are usually either visually (subsection 2.2.1) or phonologically similar (subsection 2.2.2) to corresponding correct characters, or both (Chang, 1995; Liu et al., 2010; Yu and Li, 2014). It is also true that erroneous characters produced by OCR possess visual similarity (Tong and Evans, 1996).

We base our similarity computation on two open databases: *Kanji Database Project*[4] and *Unihan Database*[5] because they provide shape and pronunciation representations for all CJK Unified Ideographs in all CJK languages.

2.2.1 Visual similarity

The *Kanji Database Project* uses the Unicode standard – Ideographic Description Sequence (IDS) to represent the shape of a character.

As illustrated by examples in Figure 2, the IDS of a character is formally a string, but it is essentially the preorder traversal path of an ordered tree.

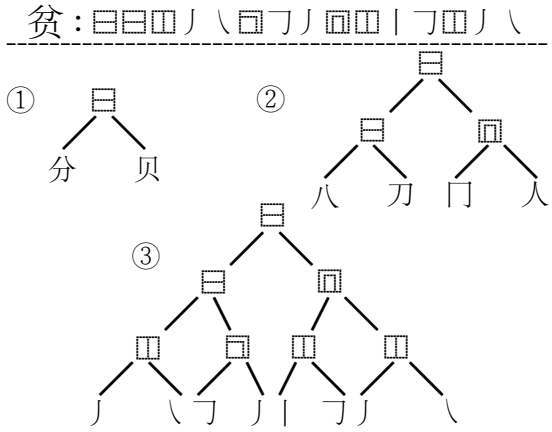

Figure 2: The IDS of a character can be given in different granularity levels as shown in the tree forms in ①-③ for the simplified character 贫 (meaning *poor*). In FASPell, we only use stroke-level IDS in the form of a string, like the one above the dashed ruling line. Unlike using only actual strokes (Wang et al., 2018), the Unicode standard Ideographic Description Characters (e.g., the non-leaf nodes in the trees) describe the layout of a character. They help us to model the subtle nuances in different characters that are composed of identical strokes (see examples in Table 2). Therefore, IDS gives us a more precise shape representation of a character.

In our model, we only adopt the string-form IDS. We define the visual similarity between two characters as one minus normalized[6] Levenshtein edit distance between their IDS representations. The reason for normalization is twofold. Firstly, we want the similarity to range from 0 to 1 for the convenience of later filtering. Secondly, if a pair of more complex characters have the same edit distance as a pair of less complex characters, we want the similarity of the more complex characters to be slightly higher than that of the less complex characters (see examples in Table 2).

We do not use the tree-form IDS for two reasons even as it seems to make more sense intuitively. Firstly, even with the most efficient algorithm (Pawlik and Augsten, 2015, 2016) so far, tree edit distance (TED) has far greater time complexity than edit distance of strings ($O(mn(m+n))$ vs. $O(mn)$). Secondly, we did try TED in preliminary experiments, but there was no significant difference from using edit distance of strings in terms of spell checking performance.

[4] http://kanji-database.sourceforge.net/

[5] https://unicode.org/charts/unihan.html

[6] Since the maximal value of Levenshtein edit distance is the maximum of the lengths of the two strings in question, we normalize it simply by dividing it by the maximum length.

Table 2: Examples of the computation of character similarities. IDS is used to compute visual similarity (V-sim) and pronunciation representations in Mandarin Chinese (MC), Cantonese Chinese (CC), Japanese On'yomi (JO), Korean (K) and Vietnamese (V) are used to compute phonological similarity (P-sim). Note that the normalization of edit distance gives us the desired fact that less complex character pair (午, 牛) has smaller visual similarity than more complex character pair (田, 由) even though both of their IDS edit distances are 1. Also, note that 午 and 牛 have more similar pronunciations in some languages than in others; the combination of the pronunciations in multiple languages gives us a more continuous phonological similarity.

	IDS	MC	CC	JO	K	V	V-sim	P-sim
午 (noon)	⿰⿱𠂉一⿱一丨	wu3	ng5	go	o	ngọ	0.857	0.280
牛 (cow)	⿱⿰丿一⿱一丨	niu2	ngau4	gyuu	wu	ngưu		
田 (field)	⿴⿻丨𠃌⿰⿱一丨一	tian2	tin4	den	cen	điền	0.889	0.090
由 (from)	⿱⿻丨𠃌⿰⿱一丨一	you2	jau4	yuu	yu	do		

2.2.2 Phonological similarity

Different Chinese characters sharing identical pronunciation is very common (Yang et al., 2012), which is the case for any CJK language. Thus, If we were to use character pronunciations in only one CJK language, the phonological similarity of character pairs would be limited to a few *discrete* values. However, a more *continuous* phonological similarity is preferred because it can make the curve used for filtering candidates smoother (see subsection 2.3).

Therefore, we utilize character pronunciations of all CJK languages (see examples in Table 2), which are provided by the *Unihan Database*. To compute the phonological similarity of two characters, we first calculate one minus normalized Levenshtein edit distance between their pronunciation representations in all CJK languages (if applicable). Then, we take the mean of the results. Hence, the similarity should range from 0 to 1.

2.3 Confidence-Similarity Decoder

Candidate filters in many previous models are based on setting various thresholds and weights for multiple features of candidate characters. Instead of this naive approach, we propose a method that is empirically effective under the principle of getting the highest possible precision with minimal harm to recall. Since the decoder utilizes contextual confidence and character similarity, we refer to it as the confidence-similarity decoder (CSD). The mechanism of CSD is explained, and its effectiveness is justified as follows:

First, consider the simplest case where only one candidate character is provided for each original character. For those candidates that are the same as their original characters, we do not substitute the original characters. For those that are different, we can draw a confidence-similarity scatter graph. If we compare the candidates with the ground truths, the graph will resemble the plot ① of Figure 3. We can observe that the true-detection-and-correction candidates are denser toward the upper-right corner; false-detection candidates toward the lower-left corner; true-detection-and-false-correction candidates in the middle area. If we draw a curve to filter out false-detection candidates (plot ② of Figure 3) and use the rest as substitutions, we can optimize character-level precision with minimal harm to character-level recall for detection; if true-detection-and-false-correction candidates are also filtered out (plot ③ of Figure 3), we can get the same effect for correction. In FASPell, we optimize correction performance and manually find the filtering curve using a training set, assuming its consistency with its corresponding testing set. But in practice, we have to find two curves – one for each type of similarity, and then take the union of the filtering results.

Now, consider the case where there are $c > 1$ candidates. To reduce it into the previously described simplest case, we rank the candidates for each original character according to their contextual confidence and put candidates that have the same rank into the same group (i.e., c groups in total). Thus, we can find a filter as previously described for each group of candidates. All c filters combined further alleviate the harm to recall because more candidates are taken into account.

In the example of Figure 1, there are $c = 4$ groups of candidates. We get a correct substitution 丰 → 主 from the group whose rank = 1, another one 苦 → 著 from the group whose rank = 2, and no more from the other two groups.

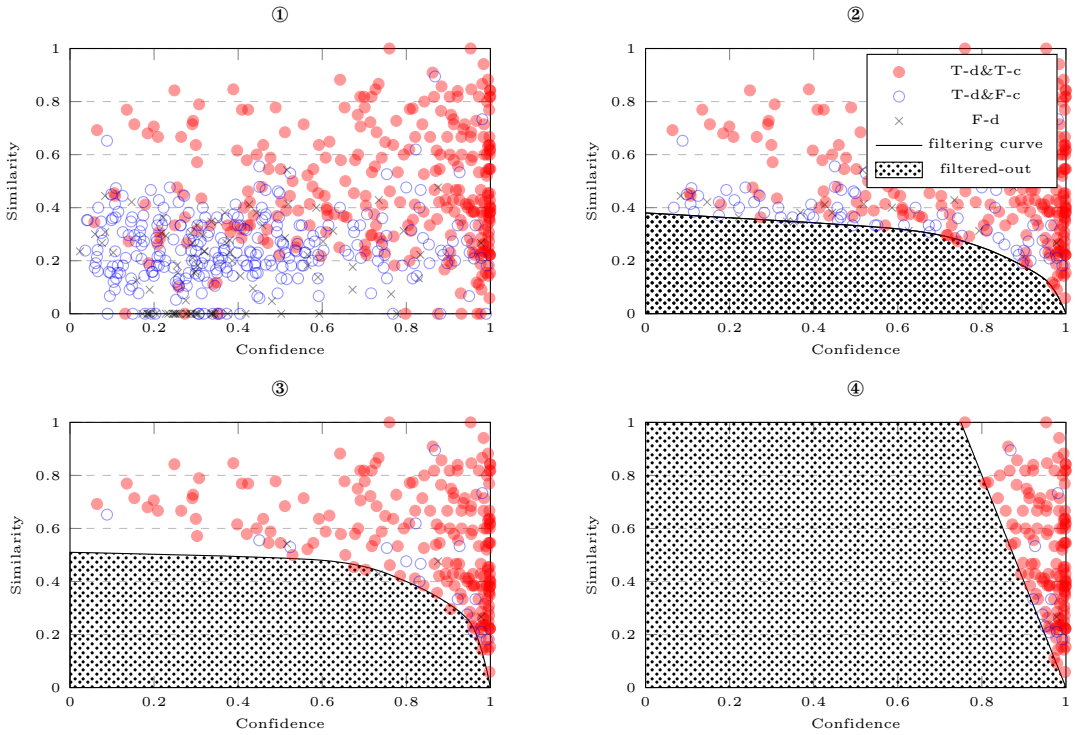

Figure 3: All four plots show the same confidence-similarity graph of candidates categorized by being true-detection-and-true-correction (T-d&T-c), true-detection-and-false-correction (T-d&F-c) and false-detection (F-d). But, each plot shows a different way of filtering candidates: in plot ①, no candidates are filtered; in plot ②, the filtering optimizes detection performance; in plot ③, as adopted in FASPell, the filtering optimizes correction performance; in plot ④, as adopted by previous models, candidates are filtered out by setting a threshold for weighted confidence and similarity ($0.8 \times confidence + 0.2 \times similarity < 0.8$ as an example in the plot). Note that the four plots use the actual first-rank candidates (using visual similarity) for our **OCR data** (Trn_{ocr}) except that we randomly sampled only 30% of the candidates to make the plots more viewable on paper.

3 Experiments and results

We first describe the data, metrics and model configurations adopted in our experiments in subsection 3.1. Then, in subsection 3.2, we show the performance on spell checking texts written by humans to compare FASPell with previous state-of-the-art models; we also show the performance on data that are harvested from OCR results to prove the adaptability of the model. In subsection 3.3, we compare the speed of FASPell and three state-of-the-art models. In subsection 3.4, we investigate how hyper-parameters affect the performance of FASPell.

3.1 Data, metrics and configurations

We adopt the benchmark datasets (all in traditional Chinese) and sentence-level[7] accuracy, precision, recall and F1 given by SIGHAN13 - 15 shared tasks on Chinese spell checking (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015). We also harvested 4575 sentences (4516 are simplified Chinese) from OCR results of Chinese subtitles in videos. We used the OCR method by Shi et al. (2017). Detailed data statistics are in Table 3.

Table 3: Statistics of datasets.

Dataset	# erroneous sent	# sent	Avg. length
Trn_{13}	350	700	41.8
Trn_{14}	3432	3435	49.6
Trn_{15}	2339	2339	31.3
Tst_{13}	996	1000	74.3
Tst_{14}	529	1062	50.0
Tst_{15}	550	1100	30.6
Trn_{ocr}	3575	3575	10.1
Tst_{ocr}	1000	1000	10.2

We use the pre-trained masked language

[7]Note that although we do not use character-level metrics (Fung et al., 2017) in evaluation, they are actually important in the justification of the effectiveness of the CSD as in subsection 2.3

Table 4: Configurations of FASPell. FT means the training set for fine-tuning; CSD means the training set for CSD; r means the number of rounds and c means the number of candidates for each character. U is the union of all the spell checking data from SIGHAN13 - 15.

FT	CSD	Test set	r	c	FT steps
$U - Tst_{13}$	Trn_{13}	Tst_{13}	1	4	10k
$U - Tst_{14}$	Trn_{14}	Tst_{14}	3	4	10k
$U - Tst_{15}$	Trn_{15}	Tst_{15}	3	4	10k
(-)	Trn_{ocr}	Tst_{ocr}	2	4	(-)

model[8] provided by Devlin et al. (2018). Settings of its hyper-parameters and pre-training are available at https://github.com/google-research/bert. Other configurations of FASPell used in our major experiments (subsection 3.2 - 3.3) are given in Table 4. For ablation experiments, the same configurations are used except when CSD is removed, we take the candidates ranked the first as default outputs. Note that we do not fine-tune the mask language model for OCR data because we learned in preliminary experiments that fine-tuning worsens performance for this type of data[9].

3.2 Performance

As shown in Table 6, FASPell achieves state-of-the-art F1 performance on both detection level and correction level. It is better in precision than the model by Wang et al. (2018) and better in recall than the model by Zhang et al. (2015). In comparison with Zhao et al. (2017), It is better by any metric. It also reaches comparable precision on OCR data. The lower recall on OCR data is partially because many OCR errors are harder to correct even for humans (Wang et al., 2018).

Table 6 also shows that all the components of FASPell contribute effectively to its good performance. FASPell without both fine-tuning and CSD is essentially the pre-trained mask language model. Fine-tuning it improves recall because FASPell can learn about common errors and how they are corrected. CSD improves its precision with minimal harm to recall because this is the underlying principle of the design of CSD.

Table 5: Speed comparison (ms/sent). Note that the speed of FASPell is the average in several rounds.

Test set	FASPell	Wang et al. (2018)
Tst_{13}	**446**	680
Tst_{14}	**284**	745
Tst_{15}	**177**	566

3.3 Filtering Speed[10]

First, we measure the filtering speed of Chinese spell checking in terms of absolute time consumption per sentence (see Table 5). We compare the speed of FASPell with the model by Wang et al. (2018) in this manner because they have reported their absolute time consumption[11]. Table 5 clearly shows that FASPell is much faster.

Second, to compare FASPell with models (Zhang et al., 2015; Zhao et al., 2017) whose absolute time consumption has not been reported, we analyze the time complexity. The time complexity of FASPell is $O(scmn + sc \log c)$, where s is the sentence length, c is the number of candidates, mn accounts for computing edit distance and $c \log c$ for ranking candidates. Zhang et al. (2015) use more features than just edit distance, so the time complexity of their model has additional factors. Moreover, since we do not use confusion set, the number of candidates for each character of their model is practically larger than ours: $x \times 10$ vs. 4. Thus, FASPell is faster than their model. Zhao et al. (2017) filter candidates by finding the single-source shortest path (SSSP) in a directed graph consisting of all candidates for every token in a sentence. The algorithm they used has a time complexity of $O(|V| + |E|)$ where $|V|$ is the number of vertices and $|E|$ is the number of edges in the graph (Eppstein, 1998). Translating it in terms of s and c, the time complexity of their model is $O(sc + c^s)$. This implies that their model is exponentially slower than FASPell for long sentences.

[8] https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip

[9] It is probably because OCR errors are subject to random noise in source pictures rather than learnable patterns as in human errors. However, since the paper is not about OCR, we do not elaborate on this here.

[10] Considering only the filtering speed is because the Transformer, the Bi-LSTM and language models used by previous state-of-the-art models or us before filtering are already well studied in the literature.

[11] We have no access to the 4-core Intel Core i5-7500 CPU used by Wang et al. (2018). To minimize the difference of speed caused by hardware, we only use 4 cores of a 12-core Intel(R) Xeon(R) CPU E5-2650 in the experiments.

Table 6: This table shows spell checking performances on both detection and correction level. Our model – FASPell achieves similar performance to that of previous state-of-the-art models. Note that fine-tuning and CSD both contribute effectively to its performance according to the results of ablation experiments. (− FT means removing fine-tuning; − CSD means removing CSD.)

Test set	Models	Detection Level				Correction Level			
		Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)
Tst_{13}	Wang et al. (2018)	(-)	54.0	**69.3**	60.7	(-)	(-)	(-)	52.1
	Yeh et al. (2013)	(-)	(-)	(-)	(-)	62.5	70.3	62.5	66.2
	FASPell	63.1	**76.2**	63.2	**69.1**	60.5	73.1	60.5	**66.2**
	FASPell − FT	40.9	75.5	40.9	53.0	39.6	**73.2**	39.6	51.4
	FASPell − CSD	41.0	42.3	41.1	41.6	31.3	32.2	31.3	31.8
	FASPell − FT − CSD	47.9	65.2	47.8	55.2	35.6	48.4	35.4	40.9
Tst_{14}	Zhao et al. (2017)	(-)	(-)	(-)	(-)	(-)	55.5	39.1	45.9
	Wang et al. (2018)	(-)	51.9	**66.2**	**58.2**	(-)	(-)	(-)	**56.1**
	FASPell	70.0	**61.0**	53.5	57.0	69.3	**59.4**	52.0	55.4
	FASPell − FT	57.8	54.5	18.1	27.2	57.7	53.7	17.8	26.7
	FASPell − CSD	49.0	31.0	42.3	35.8	44.9	25.0	34.2	28.9
	FASPell − FT − CSD	56.3	38.4	26.8	31.6	52.1	26.0	18.0	21.3
Tst_{15}	Zhang et al. (2015)	70.1	**80.3**	53.3	**64.0**	69.2	**79.7**	51.5	62.5
	Wang et al. (2018)	(-)	56.6	**69.4**	62.3	(-)	(-)	(-)	57.1
	FASPell	74.2	67.6	60.0	63.5	73.7	66.6	59.1	**62.6**
	FASPell − FT	61.5	74.1	25.5	37.9	61.3	72.5	24.9	37.1
	FASPell − CSD	65.5	49.3	59.1	53.8	60.0	40.2	48.2	43.8
	FASPell − FT − CSD	63.7	59.1	35.3	44.2	57.6	38.3	22.7	28.5
Tst_{ocr}	FASPell	18.6	**78.5**	18.6	30.1	17.4	**73.4**	17.4	**28.1**
	FASPell − CSD	**34.5**	65.8	**34.5**	**45.3**	**18.9**	36.1	**18.9**	24.8

3.4 Exploring hyper-parameters

First, we only change the number of candidates in Table 4 to see its effect on spell checking performance. As illustrated in Figure 4, when more candidates are taken into account, additional detections and corrections are recalled while maximizing precision. Thus, increase in the number of candidates always results in the improvement of F1. The reason we set the number of candidates $c = 4$ in Table 4 and no larger is because there is a trade-off with time consumption.

Second, we do the same thing to the number of rounds of spell checking in Table 4. We can observe in Figure 4 that the correction performance on Tst_{14} and Tst_{15} reaches its peak when the number of rounds is 3. For Tst_{13} and Tst_{ocr}, that number is 1 and 2, respectively. A larger number of rounds sometimes helps because FASPell can achieve high precision in detection in each round, so undiscovered errors in last round may be detected and corrected in the next round without falsely detecting too many non-errors.

4 Conclusion

We propose a Chinese spell checker – FASPell that reaches state-of-the-art performance. It is based on DAE-decoder paradigm that requires only a small amount of spell checking data and gives up the troublesome notion of confusion set. With FASPell as an example, each component of the paradigm is shown to be effective. We make our code and data publically available at https://github.com/iqiyi/FASPell.

Future work may include studying if the DAE-decoder paradigm can be used to detect and correct grammatical errors or other less frequently studied types of Chinese spelling errors such as dialectical colloquialism (Fung et al., 2017) and insertion/deletion errors.

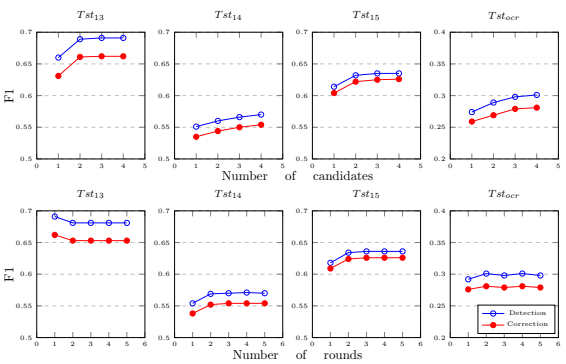

Figure 4: The four plots in the first row show how the number of candidates for each character affects F1 performances. The four in the second row show the impact of the number of rounds of spell checking.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments. We also thank our colleagues from the IT Infrastructure team of iQIYI, Inc. for the hardware support. Special thanks go to Prof. Yves Lepage from Graduate School of IPS, Waseda University for his insightful advice about the paper.

References

Chao-Huang Chang. 1995. A new approach for automatic chinese spelling correction. In *Proceedings of Natural Language Processing Pacific Rim Symposium*, volume 95, pages 278–283. Citeseer.

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

David Eppstein. 1998. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673.

Gabriel Fung, Maxime Debosschere, Dingmin Wang, Bo Li, Jia Zhu, and Kam-Fai Wong. 2017. Nlptea 2017 shared task – Chinese spelling check. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 29–34, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. 2010. Visually and phonologically similar characters in incorrect simplified chinese words. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 739–747, Beijing, China. Association for Computational Linguistics.

Mateusz Pawlik and Nikolaus Augsten. 2015. Efficient computation of the tree edit distance. *ACM Trans. Database Syst.*, 40(1):3:1–3:40.

Mateusz Pawlik and Nikolaus Augsten. 2016. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157 – 173.

Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.

DS Shih et al. 1992. A statistical method for locating typo in Chinese sentences. *CCL Research Journal*, pages 19–26.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Xiang Tong and David A. Evans. 1996. A statistical approach to automatic OCR error correction in context. In *Fourth Workshop on Very Large Corpora*.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to SIGHAN 2015 bake-off for Chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for Chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527, Brussels, Belgium. Association for Computational Linguistics.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42, Nagoya, Japan. Asian Federation of Natural Language Processing.

Shaohua Yang, Hai Zhao, Xiaolin Wang, and Bao liang Lu. 2012. Spell checking for Chinese. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Jui-Feng Yeh, Sheng-Feng Li, Mei-Rong Wu, Wen-Yi Chen, and Mao-Chuan Su. 2013. Chinese word spelling correction based on n-gram ranked inverted index list. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 43–48, Nagoya, Japan. Asian Federation of Natural Language Processing.

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.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of SIGHAN 2014 bake-off for Chinese spelling check. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132, Wuhan, China. Association for Computational Linguistics.

Shuiyuan Zhang, Jinhua Xiong, Jianpeng Hou, Qiao Zhang, and Xueqi Cheng. 2015. HANSpeller++: A unified framework for Chinese spelling correction. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 38–45, Beijing, China. Association for Computational Linguistics.

Hai Zhao, Deng Cai, Yang Xin, Yuzhu Wang, and Zhongye Jia. 2017. A hybrid model for Chinese spelling check. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 16(3):21:1–21:22.