Legal Query Reformulation using Deep Learning

Arunprasath Shankar
LexisNexis
Raleigh, USA
arunprasath.shankar@lexisnexis.com

Venkata Nagaraju Buddarapu
LexisNexis
Raleigh, USA
venkatanagaraju.buddarapu@lexisnexis.com

ABSTRACT

Query reformulation is the process of iteratively modifying a query to improve the quality of search engine results. In recent years, the task of reformulating natural language (NL) queries has received considerable diligence from both industry and academic communities. Traditionally, query reformulation has been mostly approached by using the *noisy channel* model. Since legal queries are diverse and multi-faceted, these traditional approaches cannot effectively handle low frequency and out-of-vocabulary (OOV) words. Motivated by these issues, we rethink the task of legal query reformulation as a type of monolingual *neural machine translation* (NMT) problem, where the input (source) query is potentially erroneous and the output (target) query is its corrected form. We propose a unified and principled framework with multiple levels of granularity. Specifically, (i) an encoder with character attention which augments the subword representation; (ii) a decoder with attentions that enable the representations from different levels of granularity to control the translation cooperatively and (iii) a semi-supervised methodology to extract and augment a large-scale dataset of NL query pairs combining syntactic and semantic operations. We establish the effectiveness of our methodology using an internal dataset, where the training data is automatically obtained from user query logs. We further demonstrate that training deep neural networks on additional data with synthesized errors can improve performance for translation.

1. INTRODUCTION

Technology and innovation are transforming the legal profession in manifold ways. The legal industry is undergoing significant disruption and arguably machine intelligence is most advancing in the area of discovery and search. Technologies are moving from basic keyword searches to *predictive coding*, which involves algorithms that predict whether a document is relevant or not. In [1], McGinnis et al. envisage two phases of technological changes in this area. The first phase, expected to come in the next 10 years, involves perfecting semantic search that will allow lawyers to input NL queries to interfaces and systems responding to those queries directly with relevant information. The second phase involves technology that is able to identify issues, given a set of facts and then suggest relevant authorities that might apply to those issues.

Query reformulation or correction is a crucial component for any service that requires users to type in NL, such as a search engine. It is an iterative process where a user reformulates (rewrites) queries to improve search results or gain new information. He achieves this by using either his own prior knowledge or through assisted tools. Legal query reformulation is not a trivial task, as legal search queries are often short, complex and lack context. Automatic query correction systems are one of the most widely used tools within NL applications. Search engines support users in this task in two ways, (a) explicitly by suggesting related queries or query completions, or (b) implicitly by expanding the query to improve quality and recall of organic results. Successful reformulations must closely align with the original query both syntactically, as sequences of characters or words, and semantically, often involving transparent taxonomic relations.

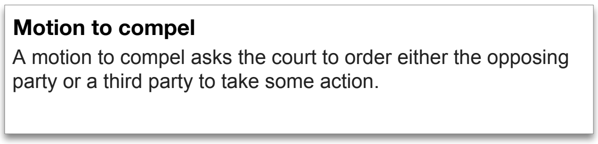
From a probabilistic perspective, given an incorrect query q^- that we wish to reformulate to a correct query q^+, we seek to model $q^+ = \arg\max_q P(q|q^-)$ where q is the original query or ground truth. In a traditional noisy channel model [2], the model consists of two parts: (i) a language model i.e., $P(q)$ that represent the prior probability of the intended correct input query; and (ii) an error model i.e., $P(q|q^-)$ that represent the process in which the correct input query gets corrupted to its incorrect form. There are several drawbacks with this approach: (i) we need two separate models and the error in estimating one model would affect the performance of the final output, (ii) it is not easy to model the channel since there is a lot of sources for these mistakes e.g., typing too fast, unintentional key stroke, phonetic ambiguity, etc. and (iii) it is not easy to obtain clean training data for language model as the input does not follow what is typical in NL. Since the end goal is to get a query that maximizes $P(q|q^-)$, can we directly model this conditional distribution instead.

In this work, we explore this route, which by passes the need to have multiple models and avoid suffering errors from multiple sources. We achieve this by applying the encoder-decoder framework combined with attention learning using recurrent neural networks [3] and rethink the query correction problem as a NMT problem, where the incorrect input is treated as a foreign language. We also propose a semi-supervised methodology to perform error detection and construct a large-scale dataset for training and experimentation. To the best of our knowledge, there is no prior research work on the idea of using attention learning in a legal setting. Also, our work is the first that uses neural machine translation based methodologies on user generated legal data for query reformulation. We demonstrate that NMT models can successfully be applied to the task of query reformulation in a legal background and validate that the trained models are naturally capable of handling orthographic errors and rare words, and can flexibly correct a variety of error types. We further find that augmenting the network training data with queries containing synthesized errors can result in significant gains in performance.

In: Proceedings of the Third Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2019), June 21, 2019, Montreal, QC, Canada.

2. APPLICATION

An "answer card" is a search feature, usually displayed in a box or panel, that occurs above organic results and tries to directly answer a question. Most answer boxes are primarily text, containing a relatively short answer and may provide limited information based on user's entered query. In legal context, the source can be a legal question, profile search or can take many forms. Figure 1 shows a card that is an answer to a legal question "**define motion to compel**" and Figure 2 shows a profile card for a judge search using the query "**who is judge antonin scalia ?**".

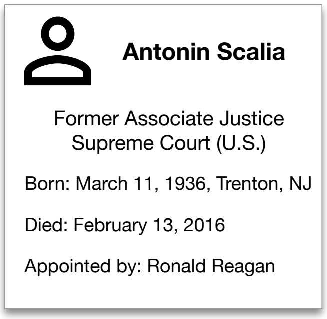


Motion to compel

A motion to compel asks the court to order either the opposing party or a third party to take some action.

Figure 1: Answer Card

Vertical engines are search engines that specialize in different types of search. Answer cards are usually backed by vertical search engines that are tightly coupled with AI/NLP systems. These NLP systems may be a machine/deep learning model(s) recognizing and classifying entities that trigger and render respective cards based on user's query intent[4]. NLP models rely heavily on vocabulary and any OOV words contained in the query can have an adverse effect on the coverage or functioning of answer cards.



Antonin Scalia

Former Associate Justice
Supreme Court (U.S.)

Born: March 11, 1936, Trenton, NJ

Died: February 13, 2016

Appointed by: Ronald Reagan

Figure 2: Profile Card

The scope of this research is to develop a framework to detect these erroneous queries and autocorrect them. The framework acts as an extra layer between the answer cards and NLP recognition systems to facilitate a coherent mechanism for coping up with missing information (vocabulary); which in turn enable us to return more appropriate cards. For e.g., for the above mentioned answer card queries, lets say a user typed in "**defin e motiom to compell**" or "**whi is jufge antonin scakia?**" instead, our deployed framework would fix (reformulate) the queries to its correct form and display the cards successfully.

3. BACKGROUND AND RELATED WORK

There are two major areas related to our research and the proposed models. First is the work on the task of query reformulation in information retrieval involving traditional, statistical and neural approaches and, second is the process of error detection (selecting candidate queries that are incorrect) for training the reformulation models. We will introduce related studies specific to these areas in the this section.

3.1 Query Reformulation

3.1.1 ***Traditional Approaches:*** In [5], Xu et al. used top results retrieved by the original query to perform reformulation by expansion. This method is popular and influenced by the initial ranking of results, however it cannot utilize user-generated data. Other approaches focused on using user query logs to expand a query by means of clickthrough rate [6], co-occurrence in search sessions, or query similarity based on click-through graphs [7]. The advantage of these approaches is that user feedback is readily available in user query logs and can efficiently be precomputed. However, these approaches all face the problem of requiring some resource dependency on a specific search engine.

3.1.2 ***Statistical Machine Translation:*** The task of machine correction is defined as correcting a N-character or word source query $S = s_1, \ldots, s_N = s_1^N$ into a M-character or word target query $T = t_1, \ldots, t_M = t_1^M$. Thus, the correction system can be defined as a function F:

$$\widehat{T} = F(S) \tag{1}$$

which returns a correction hypothesis $\widehat{T}$ given an input word S. Recently, several studies have been adopting data from user query logs as input to Statistical Machine Translation (SMT) for query reformulation and expansion [8][9]. These methods treat user queries as one language and the reformulated queries as another language. However, their major drawback is the difficulty of modeling corrections in different granularities, e.g., characters or words, which is necessary to reduce the rate of unknown words that are detrimental to their proper functioning.

Our approach differs in several ways. First, we consider full query pairs as training data, and do not use single word tokens as a 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 word/character based NMT models to derive more meaningful and accurate lexical translation models.

3.1.3 ***Neural Machine Translation:*** Recently, the use of neural networks has delivered significant gains for mapping tasks between pairs of sequences due to to their ability to learn a better representation of the data and context. Recurrent Neural Networks (RNNs) are a set of neural networks for processing sequential data and modeling long-distance dependencies which is a common phenomenon in human language. NMT models [10] use RNNs and learn to map from source language input to target language input via continuous-space intermediate representations effectively. We use an encoder-decoder bound uni/bi-directional RNNs with an attention mechanism as the core component of our proposed system. [11]. The encoder maps the input query to a higher-level representation with a uni or bi-directional RNN architecture similar to that of [12]. The decoder also employs a RNN that uses content-based attention mechanism [13] to attend to the encoded representation and generate the output query one character at a time.

3.1.4 ***Character Level Reasoning:*** Word-level NMT models are poorly suited to handle OOV words due fixed vocabulary [14]. Recent works have proposed workarounds for this problem. Since a word is usually thought of as the basic unit of language communication [15], early NMT systems built these representations starting from the word level [16]. Machine learning/NLP models typically limit vocabulary size due to the complexity of training [17]. Therefore, they are unable to translate rare words, and it is a standard practice to replace OOV words with unknown (UNK) symbol. By doing so, useful information is discarded, resulting in systems that are not able to correct erroneous words. In our framework, we strive to circumvent this issue by using smaller units such as subwords to address the problem of OOV words.

3.2 Error Detection

In most translation systems, before any reformulation is carried out, a detection process is conducted on the input query to extract any potentially incorrect words. In [18], He et al. proposed a learning to rewrite framework that focuses on candidate ranking for error detection. A non-word error refers to a potentially incorrect word that does not exist in a given dictionary. Dictionary lookup is one of the basic techniques employed to compare input strings with the entries of a language resource, e.g., lexicon or corpus [19]. Such a language resource must contain all inflected forms of the words and it should be updated regularly. If a given word does not exist in the language resource, it will be marked as a potentially incorrect word which is a huge disadvantage of this approach.

Minimum edit distance is one of the most studied techniques for error detection. It is based on counting edit operations, which are defined in most systems as insertion, deletion, substitution and transposition. Jaro-Winkler [20], Wagner-Fischer [21], and Levenshtein [22] are among the most famous edit distance algorithms. This approach has a main drawback, in that it penalizes all change operations in the same way, without taking into account the character that is used in the change operation. In contrast to above mentioned approaches, our error detection methodology combines both syntactic and semantic attributes of a user query to extract incorrect non-word occurrences. Here, we establish a unified framework (see Figure 3) that encompasses aspects like frequency distribution, query rank and subword representations for error detection alongside NMT and attention learning.

4. PROPOSED FRAMEWORK

4.1 Query Preparation

For our experiments, we collected a total of $\sim 62M$ distinct queries grouped by frequency, collected over a period of 5 years. The queries under study included a considerable proportion of boolean queries $\sim 32M$ of them. We used a heuristic approach to filter search queries. Using a combination of regular expressions and strict constraints, we excluded any query with boolean or special character connectors; this left us with $\sim 29M$ NL queries.

We used the *punkt* sentence tokenizer [23] to tokenize the queries into words. The tokens were further filtered down to remove noise and short forms. Any token that is a digit and of length $\geqslant 5$ was excluded from our study. The created vocabulary contained $\sim 1M$ words (tokens). Table 1 shows top 10 ranked queries (left) and tokens (right) at the end of query preparation process.

Rank	Query
1	breach of contract
2	unjust enrichment
3	summary judgement
4	res judicata
5	fraud
6	motion to dismiss
7	conversion
8	statute of limitations
9	defamation
10	civil procedure

Rank	Token
1	motion
2	contract
3	court
4	judgment
5	insurance
6	evidence
7	property
8	attorney
9	liability
10	statute

Table 1: Top 10 Queries and Tokens

4.2 Query Representation

The first step in creating a model, is to choose a representation for the input and output. For instance, a query can be represented in word-level, which is transferring the data with indices that refer to the words or in character-level, which is transferring the data with indices that refer to a closed vocabulary of language specific characters and symbols. However, in this work, we anticipate the need to handle an exponentially large input space of tokens by choosing a character-level query representation. Thus, model inputs are individual characters which are a sequence of indices corresponding characters.

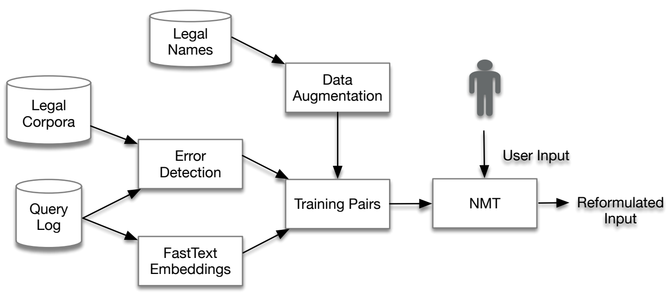


Figure 3: Proposed Framework

We have defined specific characters like <START> and <END> in order to specify start and end of each query. This character will be later used in the training process as a criteria. Note that in the character-level representation, we have also considered spaces, certain diacritics and punctuations as unique characters, i.e., 40 unique characters in all. Once a query is mapped into indices, the characters are embedded, i.e. each character is represented by a one-hot vector and is multiplied by a trainable matrix with the size of the input **X** embeddings size. The embedding allows the model to group together characters that are similar for the task.

4.3 Candidate Selection

Our scoring heuristic calculates a sequence of feature scores for each token from the prepared vocabulary. The key motivation is to separate these tokens into $+ve$ (correct) and $-ve$ (incorrect) buckets. We achieve this by combing three distributional attributes.

First, since our vocabulary is derived from user log queries, we can associate the individual tokens to the rank of its constituent queries. Second, we can also affiliate a token to its individual frequency within the vocabulary. And third, by the membership of the tokens to selected legal lexicons. Our selected lexicons comprise of extracted vocabulary from headnotes ($\sim 217K$) derived from US legal caselaw documents, legal names consisting of judge ($\sim 67K$) and expert witness ($\sim 388K$) names derived legal master databases. We also use a secondary lexicon of common US english names ($\sim 234K$).

Let Q be an ordered set of distinct legal queries derived from user logs and ranked by frequency. $Q \Rightarrow \{q_1, q_2, \ldots, q_r\}$ where r is the rank of the query; $r \in R$. Let R denote an ordered set of ranks; $R \Rightarrow \{1, 2, \ldots, L\}$, where L is the length of Q i.e., total number of distinct queries. Let $\triangledown \Rightarrow \{x_1, x_2, \ldots, x_l\}$ denote the vocabulary of distinct tokens derived from Q where l is length of the individual query. For any given token x in vocabulary $\triangledown$, it can belong to a subset of queries $\hat{Q}$; $\hat{Q} \subseteq Q$. This implies that the token can mapped to a closed set of ranks $\hat{R}$; $\hat{R} \subseteq R$. E.g., the token *discriminatory* can belong to multiple queries within Q, hence $\hat{R}$ could be a closed set e.g., $\{6, 37, 763, 23445\}$.

4.3.1 ***Mean Rank Score:*** Given $\hat{R} \subseteq R$, for a token $x \in Q$, $\hat{R}$ can be defined as $\{r_1, r_2, \ldots, r_l\}$. We compute the mean rank score (ξ) as follows:

$$\xi = \frac{\sum_{i=1}^{l} 1 - \frac{r_i}{L}}{|\hat{R}|} \tag{2}$$

where $|\hat{R}|$ is size of $\hat{R}$.

4.3.2 ***Neighbor Rank Score:*** Given a token x_0, it can co-occur alongside other tokens from the vocabulary contained to the queries under investigation. Let $\hat{\triangledown} \Rightarrow \{x_1, x_2, \ldots, x_l\}$ denote the subset of tokens co-occurring alongside x. For every candidate token inside $\hat{\triangledown}$, we have a corresponding mean rank score ξ derived from equation 2. We restrict the window of co-occurrence to 2 neighbors, one before and after the token. Let $\hat{\xi}$ be the set of corresponding mean rank scores; $\hat{\xi} \Rightarrow \{x_1, x_2, \ldots, x_l\}$ The neighbor rank score (χ) is computed as:

$$\chi = \sum_{i=1}^{l} \frac{\xi_i}{|\hat{\xi}|} \tag{3}$$

4.3.3 ***Frequency Score:*** Let ϵ denote the frequency of a token; $x \in \triangledown$. Frequency score (ζ) is a normalized value representing the importance of frequency of occurrence of a token in the corpus. It is defined as follows:

$$\zeta = \left\| \frac{\epsilon}{|\triangledown|} \right\| \tag{4}$$

4.3.4 ***Lexicon Membership Score:*** Let $\triangledown_H$ denote the vocabulary of tokens derived from headnotes. Let $\triangledown_J$ denote the vocabulary of tokens derived from judge master (judge name database) and let $\triangledown_E$ denote tokens from expert witness database. Let $\triangledown_C$ indicate all the tokens from common valid US English names vocabulary. For any given token x, the lexicon membership score (φ) is computed as follows:

$$\varphi = \begin{cases} 1, & \text{if } (x \in \triangledown_H) \vee (x \in \triangledown_J) \vee (x \in \triangledown_E) \vee (x \in \triangledown_C) \\ \frac{1}{2}, & \text{otherwise} \end{cases} \tag{5}$$

4.3.5 ***Relevance Score:*** Finally, a relevance score (Ω) is computed by combining all the scores derived from equations 2, 3, 4 and 5 as follows:

$$\Omega = \left\| \varphi \cdot \left(1 - \frac{1}{e^{2\xi} + 1}\right) \cdot \left(1 - e^{-\chi}\right) \cdot \left(1 - \frac{1}{1 + \zeta}\right) \right\| \tag{6}$$

The relevance score expresses the relative importance of a token and can be used to classify a word into $+ve$ or $-ve$ bucket. We establish this separation by using a cut-off threshold (~0.83) obtained via trial and error experiments.

4.4 Ranking Strategy using FastText

Traditional approaches to representing rare and OOV words follow assigning a random vector [24] or binning all rare words into a new "UNK" word type. Another popular technique is to encode word definitions with an external resource (Long et al., 2016) and train at the morpheme level [25]. Creating semantic representations for OOV words is a difficult NLP task. And word-based (Mikolov et al., 2013; aka word2vec)[26] approaches do not contribute well to resolve this problem.

Token	Score
roberds	0.9967
robers	0.9962
robergs	0.9913
robertus	0.9851
robertson	0.9797
robertt	0.9777
rowert	0.9766
rochfort	0.9758
rogerts	0.9756
robtert	0.9731

Token	Score
cbreach	0.9839
fbreach	0.9771
onbreach	0.9684
breachh	0.9683
bebreach	0.9659
breachj	0.9657
breachn	0.9643
ofbreach	0.9641
mcbreach	0.9627
orbreach	0.9612

Table 2: Top-10 Similar Words by fastText

Character-based embedding approaches are more suited to handle the OOV problem (Bojanowski et al., 2017; aka fastText)[27]. For mapping the $+ve$ tokens to their corresponding variants (incorrect $-ve$ counterparts), we used fastText [28]. **fastText** is a character-based embedding approach that it is well-suited to produce embeddings for OOV words. It's able to do this by learning vectors for character n-grams within the word and summing those vectors to produce the final vector or embedding for the word itself. For training, we set the embedding size to 100 and the training window to 10. We also set alpha value to 0.025, min and max values of n to 2 and 6 respectively. The embedding matrix was created using a batch size of $10K$ words in 100 epochs. We trained fastText embeddings on the $29M$ NL queries using a $p3.2x$large EC2 instance with 1 Tesla $V100$ GPU. Table 2 shows top 10 most similar words for the OOV $+ve$ word "*roberts*" (left) and OOV $-ve$ word "*breach*".

4.5 Query Augmentation

Legal names like judge, expert witness, attorney names etc. are often misspelled by users during search. Since we were not able to capture a considerable amount of these errors from query logs (real world data), we decided to synthetically augment these type of errors and generate queries for legal names algorithmically. For this purpose, we took all the names ($+ve$ instances) from our legal corpora and applied a variant of **Needleman-Wunsch** algorithm to create its $-ve$ counterparts.

The distance computation is identical to Levenshtein except that character mistakes are given different weights depending on how far two characters are on a standard keyboard layout (QWERTY). The weights are assigned by projecting the keyboard characters onto a cartesian system. For e.g., A to S is given a mistake weight of 0.4, while A to D is a 0.6. We generate query pairs using 4 types of transformation - addition, deletion, replacement and transposition. Table 3 shows a few of the generated examples for augmentation.

Misspelled	Corrected	Operation
mchael lowy	michael lowy	addition
david yamhamoto	david yamamoto	deletion
christibe ballard	christine ballard	replacement
warren glciker	warren glicker	transposition

Table 3: Augmented Queries

4.6 Training Pairs

In previous sections, we discussed the need to segregate the vocabulary into $+ve$ and $-ve$ tokens and the use of fastText character n-grams or subwords to capture word mappings ($+ve$ to $-ve$). At the end of the candidate selection process, the entire vocabulary is broken into two groups of $\sim 180K$ $+ve$ and $\sim 830K$ $-ve$ tokens. At the end of ranking using fastText, we end up with around $\sim 2.4M$ legal term mapping and $\sim 50K$ legal name mapping. The legal name mapping is expanded via the augmentation process elaborated in the previous section to around $\sim 1.5M$ pairs. The legal term and name mappings are then matched against the indexed $60M$ user queries to create **query pairs** that will be used for training our translation models.

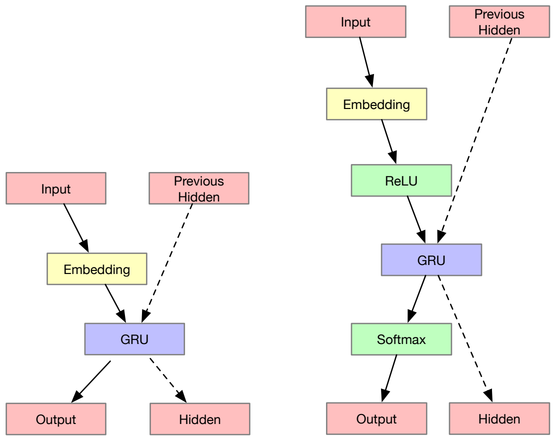
4.7 Neural Machine Translation

Most NMT systems follow the encoder-decoder framework with attention mechanism proposed by Bahdanau et al. [29]. Given a source query $q^- = q_1^- \cdots q_i^- \cdots q_I^-$ and a target query $q^+ = q_1^+ \cdots q_j^+ \cdots q_J^+$, we aim to directly model the translation probability as:

$$P(q^+|q^-;\Theta) = \prod_1^J P\left(q_j|q_{<j}^+, q^-;\Theta\right) \tag{7}$$

where Θ is a set of parameters and $q_{<j}^+$ is a sequence of previously generated target characters.

4.7.1 ***Encoder:*** We use an RNN to architect an encoder. The Encoder outputs a value for every character from the input query. The task of the encoder is to provide a representation of the input query. The input is a sequence of characters, for which the embedding matrix is constructed using fasttext[28]. Also to get the right context, we explore both uni and bi-directional RNNs here. For every input character the encoder outputs a vector and a hidden state, and uses the hidden state for the next input character. Figure 4(a) portrays a simple overview of the encoder.



(a) Encoder (b) Decoder

Figure 4: Encoder and Decoder

Following [29], we use a bi-directional RNN with Gated Recurrent Units (GRUs) to encode the source query:

$$\begin{aligned}\overrightarrow{h_i} &= \textbf{GRU}\left(\overrightarrow{h_{i-1}}, s_i; \overrightarrow{\Theta}\right)\\ \overleftarrow{h_i} &= \textbf{GRU}\left(\overleftarrow{h_{i-1}}, s_i; \overleftarrow{\Theta}\right)\end{aligned} \tag{8}$$

where s_i is the i^{th} source embedding, **GRU** is a gated recurrent unit, $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are the parameters of forward and backward GRU, respectively. The annotation of each source character q_i^- is obtained by concatenating the forward and backward hidden states:

$$\overleftrightarrow{h_i} = \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h_i} \end{bmatrix} \tag{9}$$

4.7.2 ***Decoder:*** The decoder is also built using a RNN. It takes the same representation of the input context along with the previous hidden state and output character prediction, and generates a new hidden decoder state and a new output character prediction. The decoder is a forward RNN (uni-directional) with GRUs predicting the translation y character by character. This prediction takes the form of a probability distribution over the entire output vocabulary (100 characters). At every step of decoding, the decoder is given an input character and hidden state. The initial input character is the start of query character <START>, and the first hidden state is the context vector (the encoder's last hidden state). Figure 4(b) illustrates a decoder and its associated states. The probability of generating the j^{th} word y_j is:

$$P\left(q_j^+|q_{<j}^+, q^-;\theta\right) = \text{softmax}\left(\begin{bmatrix} \psi_{j-1} \\ \varphi_j \\ \kappa_j \end{bmatrix}\right) \tag{10}$$

where ψ_{j-1} is the character embedding of the $j-1^{\text{th}}$ target word, φ_j is the decoder's hidden state of time j, and κ_j is the context vector at time j. The state φ_j is computed as:

$$\varphi_j = \textbf{GRU}\left(\varphi_{j-1}, \begin{bmatrix} \psi_{j-1} \\ \kappa_j \end{bmatrix}; \Theta_\varphi\right) \tag{11}$$

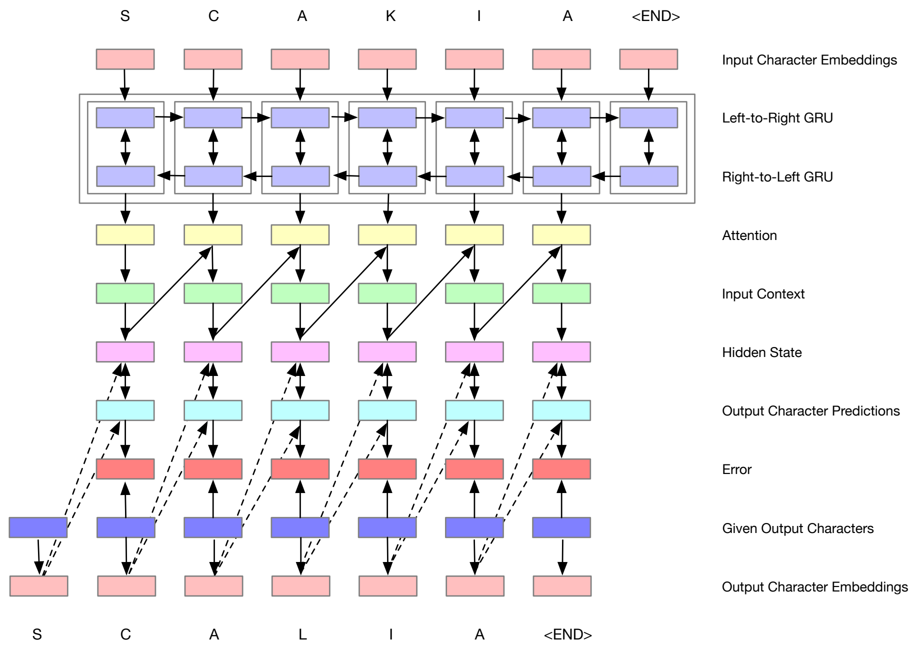


Figure 5: Encoder Decoder with Attention

4.8 Attention Mechanism

The attention mechanism was introduced to address the limitation of modeling long dependencies and the efficient usage of memory for computation. It intervenes as an intermediate layer between the encoder and the decoder, having the objective of capturing the information from the sequence of tokens that are relevant to the contents of the query [13]. The mechanism is shown in Figure 5. The basic problem that the mechanism solves is that instead of forcing the network to encode all parameters into one fixed-length vector, it allows the network to make use of the input sequence.

Key to our approach is the use of a character-based model with an attention mechanism, which allows for orthographic errors to be captured and avoids the OOV problem suffered by word-based NMT methods. Unlike the encoder-decoder model that uses the same context vector for every hidden state of the decoder, attention computes the context vector c_j as a weighted sum of the source annotations:

$$\kappa_j = \sum_{i=1}^{I} \Delta_{ji} \cdot \overleftrightarrow{h_i} \tag{12}$$

where the attention weight Δ_{ji} is computed as:

$$\Delta_{ji} = \frac{\exp\left(\Xi_{ji}\right)}{\sum_{i=1}^{I} \exp\left(\Xi_{ji}\right)} \tag{13}$$

$$\Xi_{ji} = \alpha_a^{\mathrm{T}} \tanh\left(\beta_a \varphi_{j-1} + \gamma_a \overrightarrow{h_i}\right) \tag{14}$$

where α_a, β_a and γ_a are the weight matrices, and Ξ_{ji} is the model that scores how well φ_{j-1} and $\overleftrightarrow{h_i}$ match.

4.9 Gated Recurrent Unit (GRU)

In practice, a simple RNN is difficult to train properly due to the problems of the vanishing/exploding gradient as described in [30]. Therefore, in this work, we utilize GRU (Cho et al., 2014 [11]) as an improved version of simple RNN which can alleviate the gradient problem. Along the lines of Long Short Term Memory (LSTM), in GRUs the forget and input gates are coupled into an *update gate* $\mathbf{z}_t$. The advantage of GRUs over LSTMs is the smaller number of gates that makes them less memory as well as computationally intense, which is often a critical aspect for NMT.

Given an input sequence $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N})$, GRU can be adopted as an encoder to compute the corresponding sequence of hidden state $\mathbf{h}_t = (\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_N})$ as:

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{U}_{hz}\mathbf{h}_{t-1}\right) \tag{15}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{U}_{hr}\mathbf{h}_{t-1}\right) \tag{16}$$

$$\widetilde{\mathbf{h}}_t = \mathbf{tanh}\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{U}_{rh}\left(\mathbf{r} \otimes \mathbf{h}_{t-1}\right)\right) \tag{17}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \otimes \mathbf{h}_{t-1} + \mathbf{z}_t \odot \widetilde{\mathbf{h}}_t \tag{18}$$

where σ is the sigmoid function and $\otimes$ is an element-wise multiplication operator. $\mathbf{z}_t$, $\mathbf{r}t$ and $\widetilde{\mathbf{h}}_t$ are the update gate, reset gate and candidate activation, respectively. $\mathbf{W}_{xz}$, $\mathbf{W}_{xr}$, $\mathbf{W}_{xh}$, $\mathbf{U}_{hz}$, $\mathbf{U}_{hr}$ and $\mathbf{U}_{rh}$ are related weight matrices.

5. EXPERIMENTS

5.1 Sequence Representation

An input or output sequence for an error correction system can be represented in different levels. At the character-level, a sequence is processed character by character. When searching for errors, humans often consider a bigger sequence of characters at word-level.

Clause-level, phrase-level, sentence-level and text-level are other common representations for modeling NMT sequences. In our research, we worked at character-level where each character in an input sequence is mapped to a real-valued number and its corresponding embedding. In order to model linguistic dependencies in each sequence, we took a selected list of characters into account including space. This enabled us to deal with different kinds of errors and a larger range of characters in each sequence. On the other hand, the output of our models were also represented at the character level.

5.2 Selection Strategy

In our experiments, for tuning and evaluation purposes we used a gold reference annotation following a simple selection strategy. The strategy is used to check whether for a search query pair (x, y), the left query x is erroneous, and if so, whether the similar candidate y is a correct correction or else provide the most likely correction. If x was not incorrect, we propagate query x to the gold reference y, i.e. for those cases, we have it identity as a true negative. For training our models, we split the $\sim 4M$ query pairs into three splits - train, dev and test in the ratio $99.995 : 0.005 : 0.005$ respectively. This produces $20K$ queries for dev and test set each. The true negatives in these sets (i.e. entries that do not need to be corrected) account to about $\sim 2\%$. The validation of annotation was mostly done via subject matter experts.

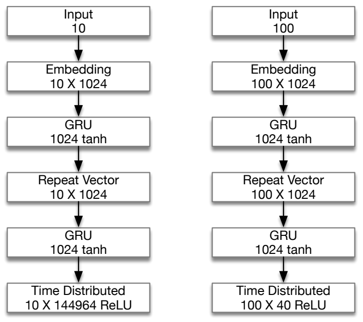
5.3 Model Architecture

In this section, we discuss about the various architectures experimented for neural machine translation.

5.3.1 ***NMT I*:.** The first architecture we experimented with, for the task of NMT is a word-based model. The input layer uses a dense vector representation of the vocabulary (144,964 words) derived from query pairs followed by an embedding layer of dimension 10X1024. The length of the sequence of the query is constrained to a maximum fixed length of 10 words. This architecture uses a repeat vector along with 1024 GRU units. This is the only word-based architecture we experimented with for NMT.

5.3.2 ***NMT II*:.** Architecture II follows a similar architecture as NMT I, except the word sequence is replaced with a character sequence. The sequence length here is constrained by two factors: (i) the total number of characters that constitute the vocabulary (40 characters) which includes special characters and, (ii) the length of the sequence which is set to fixed maximum length of 100 characters. Both architectures NMT I and II do not use attention and use a single embedding at the character level. Figure 6 shows the architectural layers for NMT models I and II.

5.3.3 ***NMT III*.** In the third architecture setup, we use two character embedding layers which is different from NMT I and II architectures which uses only one embedding. NMT III also does not use a repeat vector layer. This is a character-based only architecture and does not use attention. Figure 7 illustrates the architecture of NMT model III. The dimensions of embedding and time-distributed layers are similar to NMT II.



(a) Word-based **(b) Char-based**

Figure 6: Architecture - NMT I and II

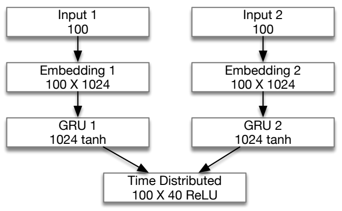


Figure 7: Architecture - NMT III

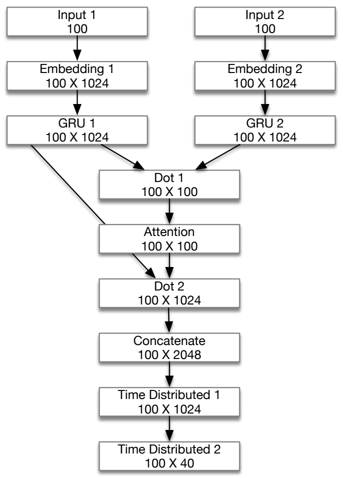


Figure 8: Architecture - NMT IV

5.3.4 ***NMT IV*.** Model IV is the first setup to use an attention layer. It is similar to NMT III in using two character embeddings, one at the encoder and other at the decoder level. See Figure 8.

5.3.5 ***Bi-directional Architectures*.** We also experimented the previously discussed neural architectures (NMT I to IV) by replacing uni-directional GRUs with bi-directional GRU units. This is only applied to the encoder component of the architecture.

5.4 Training

For our experiments, we used $p3.16x$large EC2 instance with 8 Tesla $V100$ GPUs. Table 4 shows the training time (in minutes) for the various architecture setups (100 epochs with a batch size of 512). The NMT models were implemented using TensorFlow.

Model	Avg Time
NMT I (Uni)	87.41
NMT I (Bi)	126.32
NMT II (Uni)	95.76
NMT II (Bi)	163.73
NMT III (Uni)	114.05
NMT III (Bi)	239.81
NMT IV (Uni)	168.25
NMT IV (Bi)	320.94

Table 4: NMT - Training Time

6. EVALUATION METRICS

This section presents evaluation methods used to evaluate the NMT models. Although various methods can be used to evaluate a machine translation system, for our use case of query reformulation, evaluation is limited to a binary classification of predictions where the matching elements are considered as true predictions and others as false. However, a good evaluation must include more details about this comparison. Over the years, a number of metrics have been proposed for evaluation of query correction, each motivated by weaknesses of previous metrics. There is no single best evaluation metric and the performance of a metric depends on the research goals and application. Thus, we have evaluated our system based on the most popular metrics to date.

In classification tasks, accuracy is one of the most widely used performance measure. Accuracy corresponds to the ratio of correctly classified inputs to the total number of inputs. One drawback of this metric is that correction actions are completely ignored. In order to take the correction actions into scope, we use additional performance metrics like *precision*, *recall* and *F-score*. For our experiments, we use $F_{0.5}$, since it places twice as much emphasis on precision as on recall. This metric has also been used in CoNLL-2014 shared task [31]. We also use more modified metrics such as $BLEU$, $GLEU$ and Character n-gram F-score ($CHRF$) thus gaining better insight into translation system's performance. These metrics are explained in the following subsections.

6.1 BLEU

The Bilingual Evaluation Understudy Score (*BLEU*) is a widely popular metric for machine translation [32]. For our evaluations, in order to apply $BLEU$ to individual queries, we used smoothed $BLEU$, whereby we add 1 to each of the n-gram counts before we calculate the n-gram precisions. This prevents any of the n-gram precisions from being zero, and thus will result in non-zero values even when there are not any 4-gram matches.

6.2 GLEU

Recently, Napoles et al. ameliorated *BLEU* metric for evaluation of grammatical error correction systems and proposed the Generalized Language Evaluation Understanding ($GLEU$) [33][34]. For $GLEU$, the precision is modified to assign extra weight to the n-grams that are present in the reference and the hypothesis, but not those of the input. We have used the last update of the original implementation of the $GLEU$ introduced in [34].

6.3 Character n-gram F-score ($CHRF$)

CHRF calculates the sentence level character n-gram F-score as described in Maja Popovic, 2015 [35]. It is shown to correlate very well with human rankings of different machine translation outputs, especially for morphologically rich targets. For our study, we use $CHRF1$ (standard F-*score*, $\beta = 1$) with uniform n-gram weights.

7. RESULTS

In the previous section, we presented the translation models for the task of query reformulation and the details of the models were explained. In this section, the results obtained from the models are discussed. Table 6 and 7 shows results of our correction models using the evaluation metrics discussed in the previous section.

7.1 Baseline

We define pairs of source queries and their ground-truth annotations as the baseline of our NMT models. In this baseline, we assume that none of our implemented models intervene in the task of correction and only references are considered as correction. Simply saying, baseline is a model that makes no corrections on the input query. It enables us to interpret the performance of each model in comparison to the default results.

7.2 Winners

As we expect, since the baseline system contains the ground-truth correction, the BLEU and the GLEU scores for the baseline system have a maximum value 1.00. Using these metrics, the bi-directional NMT model (model IV) with the attention layer shows higher scores in comparison to other models, i.e., F-*score* = 94.11%, $BLEU$ = 0.9255 and $GLEU$ = 0.9256. Interestingly, $CHRF$ scores of models III and IV (bi-directional) are almost identical. The choice of metric also portrays few other insights, for example $CHRF$ performs poorly on the addition operation. Similarly, $GLEU$ metric is a bad choice for operations replacement and transposition while performs very well on other type of operations.

	Uni			Bi		
Model	P	R	$F_{0.5}$	P	R	$F_{0.5}$
NMT I	83.76	85.37	84.08	83.91	81.01	83.31
NMT II	86.24	84.61	85.90	88.64	87.22	88.35
NMT III	91.17	80.04	90.74	92.77	93.86	92.98
NMT IV	93.08	90.79	92.61	93.61	96.19	94.11

Table 6: NMT Results - Standard Metrics (%)

Incorrect	Correct
contact void due to barratry or champterty	contract void due to barratry or champerty
nondisclsoure unenforceable lackof consideration summary judgment for breach o fcontract	nondisclosure unenforceable lack of consideration summary judgment for breach of contract
declaratry judgmentnd discovery	declaratory judgment and discovery
business judgment rulegross negligencce	business judgment rule and gross negligence
tennesee power of attorneyey	tennessee power of attorney
collaterale stopp el amount of damages	collateral stoppel amount of damages
agreement to remove fence statue of drauds	agreement to remove fence statute of frauds
purpose of motion inlimine are evidentary	purpose of motion in limine are evidentiary
officerwilliam alexsander karabelas	officer william alexander karabelas
associate justise ruth bader ginsberg	associate justice ruth bader ginsburg
officerwilliam alexsander karabelas	officer william alexander karabelas

Table 5: Sample Results - Query Reformulation

	Uni			Bi		
Model	*BLEU*	*GLEU*	*CHRF*	*BLEU*	*GLEU*	*CHRF*
NMT I	0.8253	0.8341	0.8601	0.8352	0.8476	0.8632
NMT II	0.8305	0.8484	0.8519	0.8524	0.8742	0.8849
NMT III	0.8645	0.8924	0.8734	0.8752	0.8994	0.9056
NMT IV	0.9024	0.9255	0.9145	0.9255	0.9256	0.9055

Table 7: NMT Results - Modified Metrics

7.3 Limitations

A few examples of source queries and its output corrections for some of our trained NMT models are illustrated in Table 5. The red tags refer to the incorrect tokens in the input and the green tags are the correctly predicted tokens by our trained models. Looking carefully at the distribution of incorrect prediction of correct input words, we can deduce that the models perform less sensibly when the size of sequence become gradually bigger. To prove this, we evaluated the models by limiting the sequences to a fixed size. Figure 9 shows the attention heatmap for a sample query.

8. CONCLUSION

In this paper, we have investigated the potential of using character-level information and subword-based NMT models for the problem of query reformulation. First, we extended the encoder with a character attention mechanism for learning better source side representations. Then, we incorporated information about source side characters into the decoder with attention, so that the character-level information can cooperate with the word-level information to better control the translation. Our experiments demonstrate the effectiveness of our models and proves that both OOV and frequent words benefit from the character-level information.

9. FUTURE WORK

For future research, we plan to explore translation models with action level, i.e., prevent over learning of models by not training them over correct input tokens (action =“OK”). Recently, reinforcement learning techniques [36] and End-to-End Memory Networks [37] have been used for the task of error correction. We plan to explore these networks to read the input sequence multiple times in order to make an output and also update memory contents at each step. We also plan to extend our work to question answering for grammar error correction and apply reformulation to boolean queries.

REFERENCES

[1] R. Pearce and J. Mcginnis, “The great disruption: How machine intelligence will transform the role of lawyers in the delivery of legal services,” *Fordham Law Review*, vol. 82, p. 3041, 05 2014.

[2] M. D. Kernighan, K. W. Church, and W. A. Gale, “A spelling correction program based on a noisy channel model,” in *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING ’90, (Stroudsburg, PA, USA), pp. 205–210, Association for Computational Linguistics, 1990.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[4] S. Arunprasath and B. Venkata Nagaraju, “Deep ensemble learning for legal query understanding,” in *Proceedings of CIKM 2018 Workshop on Legal Data Analytics and Mining (LeDAM 2018)*, CEUR-WS.org, October 2018. To appear.

[5] J. Xu and W. B. Croft, “Query expansion using local and global document analysis,” in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’96, (New York, NY, USA), pp. 4–11, ACM, 1996.

[6] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, “Probabilistic query expansion using query logs,” in *Proceedings of the 11th International Conference on World Wide Web*, WWW ’02, (New York, NY, USA), pp. 325–332, ACM, 2002.

[7] B. M. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani, “Concept-based interactive query expansion,” in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM ’05, (New York, NY, USA), pp. 696–703, ACM, 2005.

[8] J. Gao and J.-Y. Nie, “Towards concept-based translation models using search logs for query expansion,” in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM ’12, (New York, NY, USA), pp. 1:1–1:10, ACM, 2012.

[9] S. Riezler, Y. Liu, and A. Vasserman, “Translating queries into snippets for improved query expansion,” in *COLING*, 2008.

[10] D. Britz, Q. V. Le, and R. Pryzant, “Effective domain mixing for neural machine translation.,” in *WMT* (O. Bojar, C. Buck, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, and J. Kreutzer, eds.), pp. 118–126, Association for Computational Linguistics, 2017.

[11] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

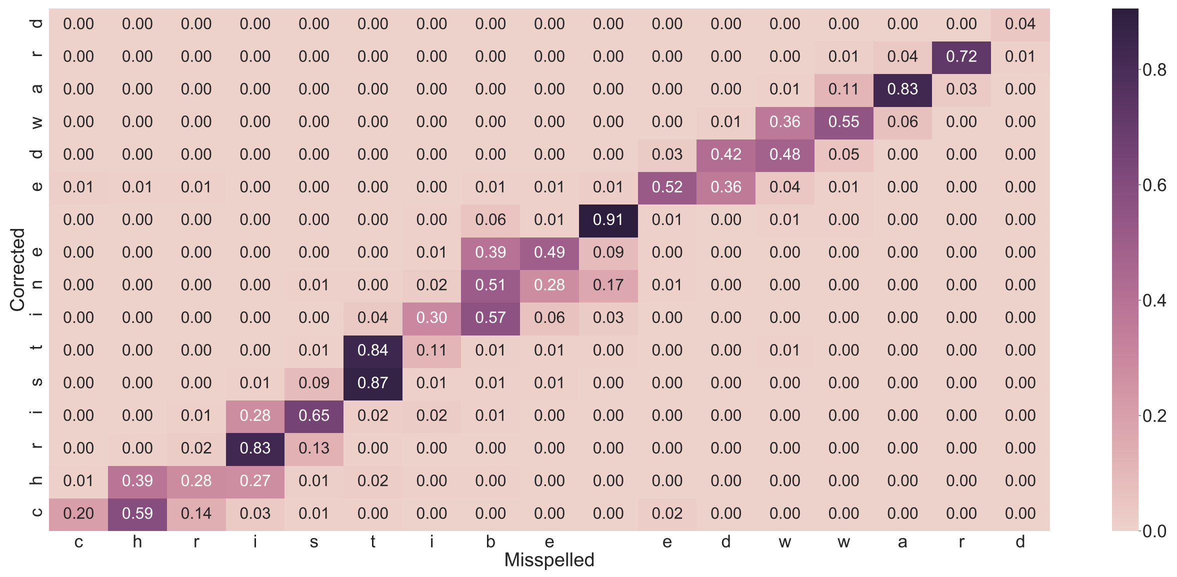


Figure 9: Attention Mechanism Heatmap

[12] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, “Listen, attend and spell,” *CoRR*, vol. abs/1508.01211, 2015.

[13] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *CoRR*, vol. abs/1409.0473, 2014.

[14] A. Graves, “Generating sequences with recurrent neural networks.,” *CoRR*, vol. abs/1308.0850, 2013.

[15] R. Jackendoff, *Semantic Structures*. Cambridge, MA: MIT Press, 1990.

[16] R. Weng, S. Huang, Z. Zheng, X.-Y. Dai, and J. Chen, “Neural machine translation with word predictions.,” in *EMNLP* (M. Palmer, R. Hwa, and S. Riedel, eds.), pp. 136–145, Association for Computational Linguistics, 2017.

[17] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, “On using very large target vocabulary for neural machine translation,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1–10, Association for Computational Linguistics, 2015.

[18] Y. He, J. Tang, H. Ouyang, C. Kang, D. Yin, and Y. Chang, “Learning to rewrite queries,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM ’16, (New York, NY, USA), pp. 1443–1452, ACM, 2016.

[19] K. Kukich, “Techniques for automatically correcting words in text,” *ACM Comput. Surv.*, vol. 24, pp. 377–439, Dec. 1992.

[20] W. E. Winkler, “String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage,” in *Proceedings of the Section on Survey Research*, pp. 354–359, 1990.

[21] R. A. Wagner and M. J. Fischer, “The string-to-string correction problem,” *J. ACM*, vol. 21, pp. 168–173, Jan. 1974.

[22] V. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[23] E. Loper and S. Bird, “Nltk: The natural language toolkit,” in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP ’02, (Stroudsburg, PA, USA), pp. 63–70, Association for Computational Linguistics, 2002.

[24] B. Dhingra, H. Liu, R. Salakhutdinov, and W. W. Cohen, “A comparative study of word embeddings for reading comprehension,” *CoRR*, vol. abs/1703.00993, 2017.

[25] T. Luong, R. Socher, and C. Manning, “Better word representations with recursive neural networks for morphology,” in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 104–113, Association for Computational Linguistics, 2013.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *CoRR*, vol. abs/1301.3781, 2013.

[27] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *TACL*, vol. 5, pp. 135–146, 2017.

[28] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[29] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *CoRR*, vol. abs/1409.0473, 2014.

[30] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[31] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, “The conll-2014 shared task on grammatical error correction,” in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pp. 1–14, Association for Computational Linguistics, 2014.

[32] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[33] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, “Ground truth for grammatical error correction metrics,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 588–593, Association for Computational Linguistics, 2015.

[34] C. Napoles, K. Sakaguchi, M. Post, and J. R. Tetreault, “GLEU without tuning,” *CoRR*, vol. abs/1605.02592, 2016.

[35] M. Popović, “chrF: character n-gram f-score for automatic MT evaluation,” in *Proceedings of the Tenth Workshop on Statistical Machine Translation*, (Lisbon, Portugal), pp. 392–395, Association for Computational Linguistics, Sept. 2015.

[36] K. Sakaguchi, M. Post, and B. Van Durme, “Grammatical error correction with neural reinforcement learning,” in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 366–372, Asian Federation of Natural Language Processing, 2017.

[37] S. Sukhbaatar, a. Szlam, J. Weston, and R. Fergus, “End-to-end memory networks,” in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2440–2448, Curran Associates, Inc., 2015.