Uncovering Code-Mixed Challenges: A Framework for Linguistically Driven Question Generation and Neural based Question Answering

Deepak Gupta[‡], **Pabitra Lenka**[†*], **Asif Ekbal**[‡], **Pushpak Bhattacharyya**[‡]
[‡]Indian Institute of Technology Patna, India
[†]International Institute of Information Technology Bhubaneswar, India
[‡]`{deepak.pcs16, asif, pb}@iitp.ac.in`
[†]`pabitra.lenka18@gmail.com`

Abstract

Existing research on question answering (QA) and comprehension reading (RC) are mainly focused on the resource-rich language like English. In recent times, there has been a rapid growth of multi-lingual contents on the web, and this has posed several challenges to the existing QA systems. Code-mixing is one such challenge that makes the task even more complex. In this paper, we propose a linguistically motivated technique for code-mixed question generation (CMQG) and a neural network based architecture for code-mixed question answering (CMQA). For evaluation, we manually create the code-mixed questions for Hindi-English language pair. In order to show the effectiveness of our neural network based CMQA technique, we utilize two benchmark datasets, *viz.* SQuAD and MMQA. Experiments show that our proposed model achieves encouraging performance on CMQG and CMQA.

1 Introduction

The people who are multilingual in nature often switch back and forth between their native languages and the foreign (popular) languages to express themselves on the web. This is very common nowadays, particularly when people express their opinions (or making any communication) through various social media platforms. This phenomenon of embedding the morphemes, words, phrases, etc. of one language into another is popularly termed as code-mixing (CM) (Myers-Scotton, 1997, 2002). The recent study (Safran, 2015) has uncovered that users frequently use question patterns, namely ‘how’ (38%), ‘why’ (24%), ‘where’ (15%), ‘what’ (11%), and ‘which’ (12%) in their queries as opposed to a ‘statement query’.

Presently, the search engines have become intelligent and are capable enough to provide precise answer to a natural language query/question[1].Several virtual assistants such as Siri, Cortana, Alexa, Google Assistant, etc are also equipped with these facilities. However, these search engines and virtual assistants are efficient only in handling the queries written in English. Let us consider the following two representations (English and code-mixed) of the same question.
(i) **Q:** *“Who is the foreign secretary of USA?”*
(ii) **Q:** *“USA **ke** foreign secretary **koun hai**?”*
***(Trans:**“Who is the foreign secretary of USA?”)*
Search engines are able to provide the exact answer to the first question. It is to be noted that although both the questions are same, the search engine is unable to return the exact answer for the second question, which is code-mixed in nature. It rather returns the top-most relevant web pages.

In this paper, we propose a framework for code-mixed question generation (CMQG) as well as code-mixed question answering (CMQA) involving English and Hindi. Firstly, we propose a linguistically motivated technique for generating the code-mixed questions. We followed this approach as we did not have access to any labeled data for code-mixed question generation. Thereafter, we propose an effective framework based on deep neural network for Code-mixed Question Answering (CMQA). In our proposed CMQA technique, we use multiple attention based recurrent units to represent the code-mixed questions and the English passages. Finally, our answer-type focused network (attentive towards the answer-type of the question being asked) extracts the answer for a given code-mixed question. We summarize our contributions as follows:
(i). We propose a linguistically motivated unsuper-

[*] Work carried out during the internship at IIT Patna

[1] The capability of handling factoid questions is higher than the complex questions or descriptive questions.

vised algorithm for Code-mixed Question Generation (CMQG). **(ii).** We propose a bilinear attention and answer-type focused neural framework to deal with CMQA. **(iii).** We create two CMQA datasets to further explore the research on CMQA. In addition to this, we manually create a code-mixed question dataset, and subsequently a code-mixed question classification dataset. **(iv)** We provide a state-of-the-art setup to extract answers from the English passages for the corresponding code-mixed questions. The source code of our proposed systems and the datasets can be found here[2].

2 Related Work

Code-mixing refers to the mixing of more than one language in the same sentence. Creating resources and tools capable of handling code-mixed languages is more challenging in comparison to the traditional language processing activities that are concerned with only one language. In recent times, researchers have started investigating methods for creating tools and resources for various Natural Language Processing (NLP) applications involving code-mixed languages. Some of the applications include language identification (Chittaranjan et al., 2014; Barman et al., 2014), part-of-speech (PoS) tagging (Vyas et al., 2014; Jamatia et al., 2015; Gupta et al., 2017), question classification (Raghavi et al., 2015), entity extraction (Gupta et al., 2018a, 2016b), sentiment analysis (Rudra et al., 2016; Gupta et al., 2016a) etc. Developing QA system in a code-mixed scenario is, itself, very novel in the sense that there have not been very significant attempts towards this direction, except the few such as (Chandu et al., 2017). Our literature survey shows that the existing methods of question generation (general) include both rules Heilman and Smith (2010); Ali et al. (2010) and machine learning (Serban et al., 2016; Wang et al., 2017a) techniques. A joint model of question generation and answering based on sequence-to-sequence neural network model is proposed in (Wang et al., 2017a).

In recent times, there have been several studies on deep learning based reading comprehension/ QA (Hermann et al., 2015; Cui et al., 2017; Shen et al., 2017; Wang et al., 2017b; Gupta et al., 2018c; Wang and Jiang, 2016; Berant et al., 2014; Maitra et al., 2018; Cheng et al., 2016; Trischler et al., 2016). To the best of our knowledge, this is the very first attempt to automatically generate the code-mixed questions (i.e. question generation), as well as provide a robust solution by developing an end-to-end neural network model for CMQA.

[2]http://www.iitp.ac.in/~ai-nlp-ml/resources.html

3 Code-Mixed Question Generation

We focus on a code-mixed scenario involving two languages, *viz.* English and Hindi. Due to the scarcity of labeled data, we could not employ any sophisticated machine learning technique for question generation. Rather, we propose an unsupervised algorithm that automatically formulates the code-mixed questions. The algorithm makes use of several NLP components such as PoS tagger, transliteration and lexical translation. We construct Hindi-English code-mixed question from a given Hindi question. Let us consider the following three questions:

- **Q_1**: *What is the name of the baseball team in Seattle?*
- **Q_2**: सिएटल में बेसबॉल दल का नाम क्या है?
  (**Trans:** *What is the name of the baseball team in Seattle?*)
  (**Transliteration**: Seattle mai baseball dal ka naam kya hai?)
- **Q_3**:*Seattle mein baseball team ka naam kya hai?*
  (**Trans:** *What is the name of the baseball team in Seattle?*)

All the three questions are same but are asked in English, Hindi and the code-mixed English-Hindi languages. It can be seen that **Q_2** and **Q_3** are similar and share many *false cognates (Moss, 1992)* [(Seattle, सिएटल), (naam, नाम), (kya, क्या), (mai, में), (baseball, बेसबॉल)]. The question **Q_3** has the direct transliteration of the Hindi words (सिएटल → Seattle), (नाम → naam), (क्या → kya), (में → mai) and (बेसबॉल → baseball). There are some words (e.g. '*team*') in **Q_3** which are the English lexical translations from Hindi. We perform a thorough study of Hindi sentences and their corresponding code-mixed Hindi-English sentences, and observe the following:

(1) Named entities (NEs) of type person (PER) remain same in both Hindi as well as the code-mixed English-Hindi (EN-HI) sentence. These NEs are only transliterated. E.g.
Hindi: <u>महात्मा गांधी</u> का जन्म कब हुआ था?
(**Trans:** When was <u>Mahatma Gandhi</u> born?)

Code-Mixed (EN-HI): *Mahatma Gandhi ka birth kab hua tha?*
The NE *Mahatma Gandhi* of type 'PER' is transliterated in code-mixed sentence.

(2) NEs of type location (LOC) and organization (ORG) present in a Hindi sentence are replaced with their best lexical translations in English. For example:
Hindi: लखनऊ दिल्ली से कितनी दूर है?
(**Trans:** How far Lucknow is from Delhi?)
Code-Mixed (EN-HI): *Lucknow New Delhi se kitni dur hai?*
The 'PER' type NEs are only transliterated as the names do not have any variation in Hindi or English. For example, सचिन तेंदुलकर (**Trans:** Sachin Tendulkar) $\rightarrow$ 'Sachin Tendulkar'. It is just needed to be written in Roman script. However, same does not hold for 'LOC' and 'ORG'. For example, transliterating भारतीय अंतरिक्ष अनुसंधान संगठन (**Trans:** Indian Space Research Organisation) $\rightarrow$ 'Bharatiya Antriksh Anushandhan Sangathan' is incorrect.

(3) PoS tags such as singular or plural noun (NN), proper noun (NNP), spatio-temporal noun (NST) and adjective (JJ) present in a Hindi sentence are often replaced with their context aware lexical translation in English. For example:
Hindi: व्यक्तियों को उनकी रचनात्मकता के लिए कौन से अधिकार दिए जाते हैं?
(**Trans:** Which rights are given to individuals for their creativity?)
Code-Mixed (EN-HI): *Individuals ko unki creativity ke liye koun se rights diya jate hain?*

The underlined words in the Hindi sentence have noun (NN) PoS tags, therefore the corresponding words are replaced with their best lexical translations in their respective code-mixed sentence.

(4) The remaining words in the Hindi sentence are transliterated (in English) and a code-mixed EN-HI sentence is formed. For example, the remaining words of the previous Hindi sentence are transliterated (in underlines) and the code-mixed EN-HI sentence is formed.
Code-Mixed (EN-HI): *Individuals ko unki creativity ke liye koun se rights diye jate hain?*

The main challenge of automatic CMQG is to find the best lexical translation which suits the most in the given context of the particular question. Let us consider the various lexical translation choices $tr_i = \{tr_{(i,\,1)}, tr_{(i,\,2)}, \dots, tr_{(i,\,l_i)}\}$ for the token (t_i), where l_i is the number of lexical translations available for the token t_i. The lexical translation disambiguation algorithm selects the most probable lexical translation of token t_i from a set of l_i possible translations. We generate a query by adding the previous token t_{i-1} and the next token t_{i+1} with the token of interest designated by t_i. The context within a query provides important clues for choosing the right transliteration of a given query word. For example, for a query S ={शहर, पूर्व, स्कॉटलैंड} (**Trans:** {city, east, Scotland}), where the word 'पूर्व' is the word in interest for which the most probable lexical translation needs to be identified from the list {*BC, East*}. Here, based on the context, we can see that the choice of translation for the word 'पूर्व' is *'east'* since the combinations {*city, east*} and {*east, Scotland*} are more likely to co-occur in the corpus than {*city, BC*} and {*BC, Scotland*}. We follow the iterative disambiguation algorithm (Monz and Dorr, 2005) which judges a pair of items to gather partial evidences for the likelihood of a translation in a given context. An occurrence graph is constructed using the query term S and the translation set TR, such that the translation candidates of different query terms are connected with the associated Dice Co-efficient weight between them. At the same time, it is also ensured that there should not be any edge between the different translation candidates of the same query term. We initialize each translation candidate with equal likelihood of a translation. After initialization, the weight of each translation candidate is iteratively updated using the weights of the translation candidate linked to it and the weight of the link connecting them. At the end of the iteration the weight of each translation candidate is normalized to ensure that these all sum up to 1.

4 Proposed Approach for CMQA

Given a code-mixed question Q with tokens $\{q_1, q_2 \dots, q_m\}$ and an English passage P having tokens $\{p_1, p_2 \dots, p_n\}$, where m and n are the number of tokens in the question and the passage, respectively. The task is to identify answer A with tokens $\{p_i, p_{i+1} \dots, p_j\}$ of length $j - i + 1$, where $1 \leq i \leq n$ and $i \leq j \leq n$.
Each model component is described below:

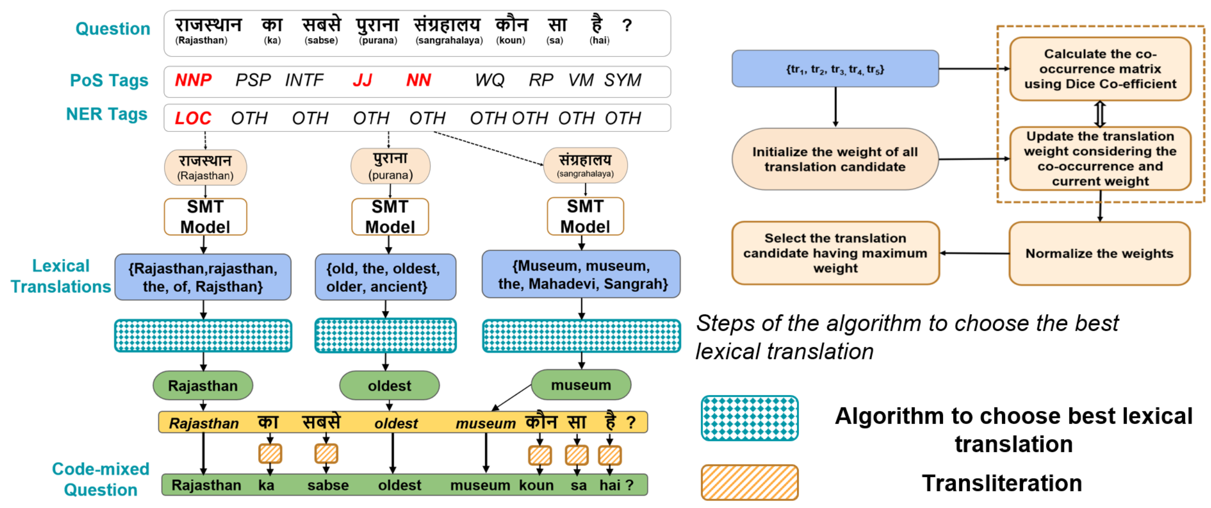
Question
राजस्थान (Rajasthan) का (ka) सबसे (sabse) पुराना (purana) संग्रहालय (sangrahalaya) कौन (koun) सा (sa) है (hai) ?
PoS Tags
NNP PSP INTF JJ NN WQ RP VM SYM
NER Tags
LOC OTH OTH OTH OTH OTH OTH OTH OTH
राजस्थान (Rajasthan)
पुराना (purana)
संग्रहालय (sangrahalaya)
SMT Model
Lexical Translations
{Rajasthan,rajasthan, the, of, Rajsthan}
{old, the, oldest, older, ancient}
{Museum, museum, the, Mahadevi, Sangrah}
Rajasthan
oldest
museum
Rajasthan का सबसे oldest museum कौन सा है ?
Code-mixed Question
Rajasthan ka sabse oldest museum koun sa hai ?
{tr1, tr2, tr3, tr4, tr5}
Calculate the co-occurrence matrix using Dice Co-efficient
Initialize the weight of all translation candidate
Update the translation weight considering the co-occurrence and current weight
Select the translation candidate having maximum weight
Normalize the weights
Steps of the algorithm to choose the best lexical translation
Algorithm to choose best lexical translation
Transliteration

Figure 1: Illustrations of the proposed CMQG. The English transliterations are given in the bracket. The right part of the image shows the basic steps to select the best lexical translation. The red color tags in PoS and NER tags list denote the tags of the words that qualify to the next step.

4.1 Token and Sequence Encoding

From the given code-mixed question Q and passage P, we first obtain the respective token-level embeddings $\{x_t^Q\}_{t=1}^m$ and $\{x_t^P\}_{t=1}^n$ from the pre-trained word embedding matrix. Due to the code-mixed nature, our model faces the out-of-vocabulary (OOV) word issue. To tackle this, we adopt character-level embedding to represent each token of the question and passage. These are denoted by $\{c_t^Q\}_{t=1}^m$ and $\{c_t^P\}_{t=1}^n$ for question and passage, respectively. The character-level embeddings are generated by taking the final hidden states of a bi-directional gated recurrent units (Bi-GRU) (Chung et al., 2014) applied to the character embedding of the tokens. The final representations of each token u_t^Q and u_t^P of question and passage, respectively, are obtained through the Bi-GRU as follows:

$$\begin{aligned} u_t^Q &= \text{Bi-GRU}(u_{t-1}^Q, [x_t^Q \oplus c_t^Q]) \\ u_t^P &= \text{Bi-GRU}(u_{t-1}^P, [x_t^P \oplus c_t^P]) \end{aligned} \tag{1}$$

where, $\oplus$ is the concatenation operator. In order to encode the token sequence, we apply convolution followed by Bi-GRU operation as follows: First, the convolution operation is performed on the zero-padded sequence $\bar{u}^P$ over the passage sequence u^P, where $\bar{u}_t^P \in \mathbb{R}^d$. A set of k filters $F \in \mathbb{R}^{k \times l \times d}$, is applied to the sequence. We obtain the convoluted features c_t^P at given time t for $t = 1, 2, \ldots, n$ by the following formula.

$$c_t^P = tanh(F[\bar{u}_{t-\frac{l-1}{2}}^P \ldots \bar{u}_t^P \ldots \bar{u}_{t+\frac{l-1}{2}}^P]) \tag{2}$$

The feature vector $\bar{C}^P = [\bar{c}_1^P, \bar{c}_2^P \ldots \bar{c}_n^P]$ is generated by applying the max pooling on each element c_t^P of C^P. This sequence of convolution feature vector $\bar{C}^P$ is passed through a Bi-GRU network. The same convolution operations are also performed over the question sequence u^Q and the convolution feature vector $\bar{C}^Q$ is obtained. Similar to e.q. 1, we compute Bi-GRU outputs v_t^P (v_t^Q) by giving the inputs v_{t-1}^P (v_{t-1}^Q) and $\bar{c}_t^P$ ($\bar{c}_t^Q$). We represent the question and passage representation matrix by $V^Q \in \mathbb{R}^{m \times h}$ and $V^P \in \mathbb{R}^{n \times h}$, respectively, where h is the number of hidden units of the Bi-GRUs.

4.2 Question-aware Passage Encoding

When a single passage contains the answer of two or more than two different questions then the passage encoding obtained from the previous layer (c.f. section 4.1) will not be effective enough to provide the answer of each question. It is because the obtained passage encoding does not take into account the question information. In this layer first we compute an attention matrix $M \in \mathbb{R}^{n \times m}$ as follows:

$$M_{i,\,j} = 1/(1 + dist(V^P[i,:],\ V^Q[j,:])) \tag{3}$$

$M_{i,\,j}$ is the similarity score between the i^{th} element of the passage encoding V^P and j^{th} element of the question encoding V^Q. The $dist(x, y)$ function is an euclidean distance[3] between x and y.

[3]We observe that e.q. 3 performs well, when $dist$ is an euclidean distance.

Thereafter, the normalization of element $M_{i,j}$ of matrix M is performed with respect to the i^th row.

$$\overline{M}_{i,\,j} = \frac{M_{i,\,j}}{\sum_{k=0}^{m} M_{i,\,k}} \tag{4}$$

Intuitively, it calculates the relevance of a word in the given passage with each word in the question. We compute the question vector $Q \in \mathbb{R}^{n \times h}$ corresponding to all the words in the passage as $Q = \overline{M} \times V^Q$. Each row t of the question vector Q denotes the encoding of the passage word t with respect to all the words in the question. The question aware passage encoding will be computed by the word-level concatenation of the passage encoding v_t^P and question vector of the t^{th} row Q_t. More formally, the question aware passage encoding a_t of the word at time t will be $a_t = v_t^P \oplus Q_t$. Finally, we apply a Bi-GRU to encode the question aware information over time. It is computed as follows:

$$s_t = \text{Bi-GRU}(s_{t-1}, a_t) \tag{5}$$

We can represent the question aware passage encoding matrix as $S \in \mathbb{R}^{n \times h}$.

4.3 Bilinear Attention on Passage

Question aware passage encoding accounts the relevance of the words in a question with the given passage. If the answer spans more than one token (i.e. a multi-word tokens), it is important to compute the relevance between the constituents of the multi-word tokens. We calculate the bilinear attention matrix $B \in \mathbb{R}^{n \times n}$ on question aware passage encoding $S \in \mathbb{R}^{n \times h}$ as follows:

$$B = S \times \mathbf{W_b} \times S^T \tag{6}$$

where, $\mathbf{W_b} \in \mathbb{R}^{h \times h}$ is a bilinear weight matrix. Similar to e.q. 4, normalization is performed on B, and the normalized attention matrix is denoted as $\overline{B}$. The element $\overline{B}_{i,j}$ is the measure of relevance between the i^{th} and j^{th} words of the passage. Similar to the question vector Q, we calculate the passage vector $R \in \mathbb{R}^{n \times h}$ as computed on $R = \overline{B} \times S$. The concatenation (word wise) of question dependent passage encoding vector s_t and passage vector r_t is performed to obtain $\overline{R_t}$ and form the matrix $\overline{R} \in \mathbb{R}^{n \times 2h}$. Similar to Wang et al. (2017b), we introduce a gating mechanism to control the impact of $\overline{R}$ and denote it as the $G \in \mathbb{R}^{n \in 2h}$. In order to identify the start and end indices of the answer from the passage, we employ two Bi-GRU with input as G. Similar to e.q. 1, output of the Bi-GRUs is computed as $P_s \in \mathbb{R}^{n \times h}$ and $P_e \in \mathbb{R}^{n \times h}$.

4.4 Answer-type Focused Answer Extraction

The answer-type of a question provides the clues to detect the correct answer from the passage. Consider a code-mixed question **Q:** *Kaun sa Portuguese player, Spanish club Real Madrid ke liye as a forward player khelta hai?* (**Trans:** *Which Portuguese player plays as a forward for Spanish club Real Madrid?*.) The answer-type of the question Q is 'person'. Even though the network has the capacity to capture this information up to a certain degree, it would be better if the model takes into account this information in advance while selecting the answer span. Li and Roth (2002) proposed a hierarchical question classification based on the answer-type of a question. Based on the coarse and fine classes of Li and Roth (2002), we train two separate answer-type detection networks on the Text REtrieval Conference (TREC) question classification dataset[4]. First, we translate[5] 5952 TREC English questions into Hindi and thereafter transform the Hindi questions into the code-mixed questions by using our proposed CMQG algorithm. We train the answer-type detection network with code-mixed questions and their associated labels using the technique as discussed in (Gupta et al., 2018b). The network learns the encoding of coarse ($C_{at} \in \mathbb{R}^h$) and fine class ($F_{at} \in \mathbb{R}^h$) of answer-types obtained from the answer-type detection network. The attention matrix M calculated in e.q. 3 undergoes the max-pooling over the columns to capture the most relevant parts of the question.

$$Q_p^j = \text{max-pool}(M[:, j]) \tag{7}$$

The max-pooled representation of question and answer-type representation are concatenated in the following way:

$$Q_f = Q_p.V^P \oplus C_{at} \oplus F_{at} \tag{8}$$

A feed-forward neural network with *tanh* activation function is used to obtain the final output $\overline{Q}_f \in \mathbb{R}^h$. The probability distribution of the beginning of answer A_s and the end of answer A_e is computed as:

$$\begin{aligned} prob(A_s) &= \text{softmax}(\overline{Q}_f \times P_s) \\ prob(A_e) &= \text{softmax}(\overline{Q}_f \times P_e) \end{aligned} \tag{9}$$

[4] http://cogcomp.org/Data/QA/QC/

[5] We use Google Translate because of its better performance on EN → HI translation.

# of CM Questions	5,535	# of Hindi Words	37,300
# of words	59,733	Average # of Hindi Words/Question	6.7389
Average Length of CM Questions	10.79	# of English Words	22,433
Code-Mixing Index (CMI) Score	37.14	Average # of English Words/Question	4.05

Table 1: Statistics of manually formulated CM questions

To train the network, we minimize the sum of the negative log probabilities of the ground truth start and end position by the predicted probability distributions.

5 Datasets and Experiments

In this section, we report the datasets and the experimental setups.

5.1 Datasets (CMQG)

For CMQG task, we require the input question to be in Hindi. We use the manually created Hindi questions obtained from the Hindi-English question answering dataset (Gupta et al., 2018b). We generate the code-mixed questions by our proposed approach (c.f. Section 3). In order to evaluate the performance of our proposed CMQG algorithm, we also manually formulate[6] the Hindi-English code-mixed questions. Details of this dataset are shown in Table 1. We compute the complexity of code-mixing using the metric, Code-mixing Index (CMI) score (Gambäck and Das, 2014). We name this code-mixed question dataset as '*HinglishQue*'. We observe that our *HinglishQue* dataset has higher CMI score as compared to the FIRE[7] 2015 (CMI=11.65) and ICON[8] 2015 (5.73) CM corpus (Soumil Mandal and Das, 2018)[9]. This implies that our *HinglishQue* dataset is more complex and challenging in comparison to the other Hindi-English code mixing (CM) dataset. The CMI score of the system generated code-mixed questions is 37.22.

5.2 Datasets (CMQA)

(1) CM-SQuAD: We generate the CMQA dataset from the portion of SQuAD (Rajpurkar et al., 2016) dataset. We translate the English questions into Hindi and use our approach of CMQG (c.f. Section 3) to transform the Hindi questions into the code-mixed questions. We manually verify the questions to ensure the quality. We use the corresponding English passage to find the answer pair of the code-mixed question. Detailed statistics of the dataset are shown in Table 2. We randomly split the dataset into training, development and test set.

(2) CM-MMQA: We experiment with a recently released multilingual QA dataset (Gupta et al., 2018b). It provides Hindi questions along with their English passages. Similar to the CM-SQuAD dataset, we create code-mixed questions and their respective answer pairs. Details of the dataset are shown in Table 2.

Datasets	Train	Dev	Test	Total
CM-SQuAD	16,632	2,080	2,080	20,792
CM-MMQA	2,746	341	341	3,428

Table 2: Detailed statistics (# of question-passage pairs) of the derived CMQA datasets

5.3 Experimental Setup for CMQG

The tokenization and PoS tagging are performed using the publicly available Hindi Shallow Parser[10]. The Polyglot[11] Named Entity Recognizer (NER) (Al-Rfou et al., 2015) is used for named entity recognition. The lexical translation set is obtained by the lexical translation table generated as an intermediate output of Statistical Machine Translation (SMT) training by Moses (Koehn et al., 2007) on publicly available[12] English-Hindi (EN-HI) parallel corpus (Bojar et al., 2014). We aggregate the output probability $p(e|h)$ and inverse probability $p(h|e)$ along with their associated words in both English (e) and Hindi (h) languages. We choose a threshold (5) to filter out the least probable translations. The co-occurrence weight (Dice Co-efficient) is calculated on the available[13] n-gram dataset consisting of unique $2,86,358$ bigrams and $3,33,333$ unigrams. For Devanagari (Hindi) to Roman (English) transliteration, we use the transliteration system[14] based on Ekbal et al. (2006). We evaluate

[6]The question formulators are the undergraduate and post-graduate students having good proficiencies in English and Hindi.

[7]http://fire.irsi.res.in/fire/2015/home

[8]http://ltrc.iiit.ac.in/icon2015/

[9]Please note that these two datasets are not related to QA

[10]http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php

[11]http://polyglot.readthedocs.io/en/latest/NamedEntityRecognition.html

[12]http://ufal.mff.cuni.cz/hindencorp

[13]http://norvig.com/ngrams/

[14]https://github.com/libindic/indic-trans

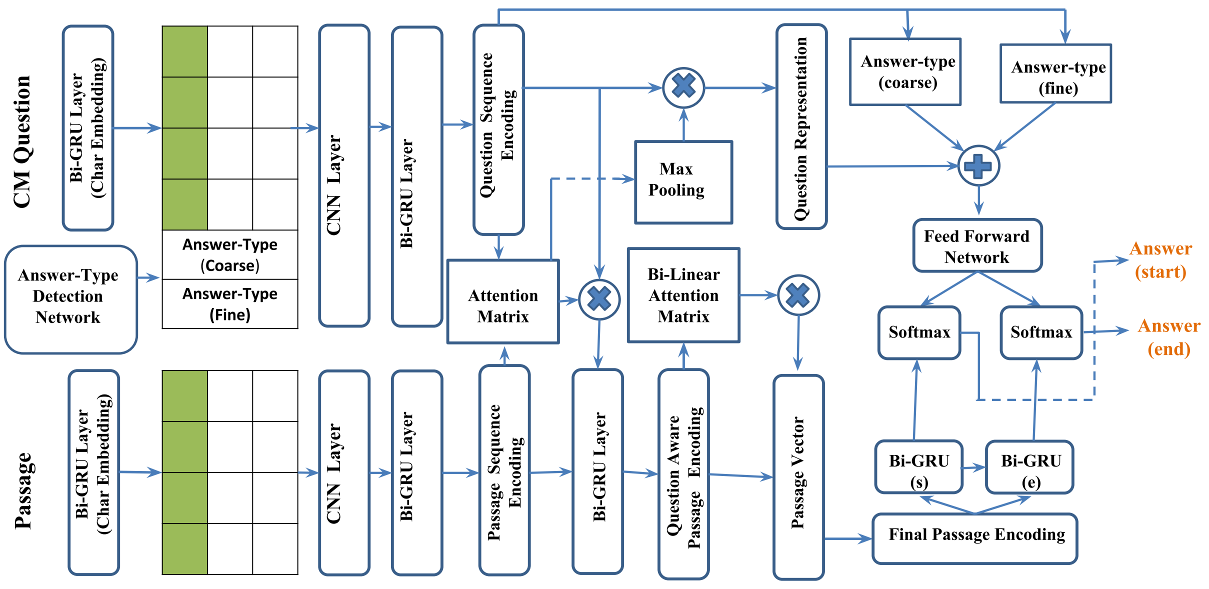

Figure 2: Proposed CMQA model architecture. The green color column denotes the character embeddings.

the performance of CMQG in terms of accuracy, BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) score.

5.4 Experimental Setup for CMQA

CMQA datasets contain the words both in Roman script and English. For English, we use the *fastText* (Bojanowski et al., 2016) word embedding of dimension 300. We use the Hindi sentences from Bojar et al. (2014), and then transliterate it into the Roman script. These sentences are used to train the word embeddings of dimension 300 by the word embedding algorithm (Bojanowski et al., 2016). Finally, we align monolingual vectors of English and Roman words in an unified vector space using a linear transformation matrix learned by the approach as discussed in Smith et al. (2017). Other optimal hyper-parameters are set to: character embedding dimension=50, GRU hidden unit size=150, CNN filter size=150, filter size=$3, 4$, batch size=60, # of epochs=100, initial learning rate=0.001. Optimal values of the hyperparameters are decided based on the model performance on the development set of CM-SQuAD dataset. Adam optimizer (Kingma and Ba, 2014) is used to optimize the weights during training. For the evaluation of CMQA, we adopt the exact match (EM) and F1-score (Rajpurkar et al., 2016).

5.5 Baselines

5.5.1 Baselines (CMQG)

We portray the problem of code-mixed question generation with respect to sequence to sequence learning where the input sequence comprises of Hindi question and the output sequence is the code-mixed EN-HI question. A seq2seq with attention (Sutskever et al., 2014; Bahdanau et al., 2014) network is trained using the default parameters of Nematus (Sennrich et al., 2017). The training dataset of the pair of Hindi translated question and code-mixed questions from CM-SQuAD dataset (c.f. Section 5.2) is used for training the seq2seq network. We evaluate the network on the manually created CMQG dataset (c.f. Section 5.1).

5.5.2 Baselines (CMQA)

To compare the performance of our proposed CMQA model, we define the following baseline models.

1) IR based model: This baseline is our implementation of the *WebShodh* (Chandu et al., 2017) with improvements in some existing components. We replaced *WebShodh*'s support vector machine based (SVM) based question classification with our recurrent CNN based answer-type detection network (c.f. Section 4.4). In spite of searching the answer on the web (as *WebShodh* does), we search it within the passage. We choose the high-

Datasets →	CM-SQuAD (1)						CM-MMQA (2)			
	Dev		Test		Test (2)		Dev		Test	
Models	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
IR (Chandu et al., 2017)	5.82	9.51	5.02	8.92	-	-	5.52	9.66	6.10	10.64
BiDAF (Seo et al., 2016)	21.44	29.18	21.63	28.45	22.26	37.54	22.38	33.10	22.09	32.82
R-Net (Wang et al., 2017b)	24.17	31.12	23.76	30.74	24.47	39.15	24.27	37.33	23.72	36.86
Proposed Approach	**31.12**	**37.78**	**31.05**	**36.97**	**30.91**	**46.18**	**28.14**	**46.25**	**30.56**	**46.10**

Table 3: Performance comparison of the proposed CMQA algorithm with the IR-based and neural-based baselines. **Test (2)** refers the test set of CM-MMQA.

Models	Accuracy	Bleu	ROUGE-1	ROUGE-2	ROUGE-L
Seq2Seq	39.24	52.18	53.28	56.05	52.11
Proposed Algorithm	67.11	86.17	95.15	90.53	95.13

Table 4: Performance comparison of the proposed CMQG algorithm with *seq2seq* baseline.

est ranked answer as our final answer.

2) R-Net (Wang et al., 2017b): This is a deep neural network based comprehension reading (RC) model. We train the R-Net model with the hyper-parameters as described in Wang et al. (2017b).

3) BiDAF (Seo et al., 2016): This is another state-of-the-art neural model for RC. We trained this model with the same hyperparameters as given in (Seo et al., 2016).

6 Results and Analysis

We demonstrate the evaluation results of our proposed CMQG algorithm on the *HinglishQue* dataset in Table 4. For evaluation, we employed three annotators who were instructed to assign the label (*same* or *different*) depending upon whether the system generated and manually created questions are similar or dissimilar. The agreement among the annotators was calculated by Cohen's Kappa (Cohen, 1960) coefficient, and it was found to be 92.45%. Evaluation of question generation

Model Components	CM-SQuAD		CM-MMQA	
	EM	F1	EM	F1
Proposed	31.12	37.78	28.14	46.25
(-) Convolution	29.46	36.14	26.19	43.76
(-) Bilinear Attention	26.42	33.31	25.36	41.29
(-) Answer-type Focused	28.41	35.14	26.69	42.37

Table 5: Effect of the various components of the CMQA model on the development set of CM-SQuAD and CM-MMQA dataset. **(-) X** denotes the model architecture after removal of '**X**'.

shows that our proposed CMQG algorithm performs better than the seq2seq based baseline. One reason could be the insufficient amount (16,632) of training instances and the out-of-vocabulary (only 62.35% words available in the vocab) issue. Performance improvement in our proposed model over the baseline is statistically significant as $p < 0.05$. In literature, we find only one study on English-Hindi code-mixed question classification i.e. Raghavi et al. (2015). They used only 1,000 code-mixed questions, and used Support Vector Machine (SVM) to classify the questions into coarse and fine-grained answer-types. They reported to achieve 63% and 45% accuracies for coarse and fine-grained answer-type detection, respectively under 5-fold cross validation setup. In contrast, we manually create 5,535 code-mixed questions and train a CNN model that shows 87.21% and 83.56% accuracies for coarse and fine answer types, respectively, for the 5-fold cross validation.

Results of CMQA for both the datasets are shown in Table 3. Performance of IR based baseline (Chandu et al., 2017) on both the datasets are poor. This may be because Chandu et al. (2017)'s system was mainly developed to answer pure factoid questions based only on the named entities denoting person, location and organization. However, the datasets used in this experiment have different types of answers beyond the basic factoid questions. We also perform a cross-domain experiment, where the test data of CM-MMQA is used to evaluate the system trained on CM-SQuAD. Performance improvements in our proposed model over the baselines are statistically significant as $p < 0.05$. Experiments show that the performance of CM-MMQA is better than CM-SQuAD. This might be due to the relatively smaller length passages in CM-MMQA, extracting answers from which are easier.

We perform ablation study to observe the effects of various components of the CMQA model. Results are shown in Table 5. The component convolution refers to the convolution operation per-

Sr. No.	Reference Questions	System Generated Questions
1	Maharaja Ranjit Singh ne Mandi par kab **occupy** kar liya tha?	Maharaja Ranjit Singh ne Mandi par kab *Czechoslovakia* kar liya tha?
2	Babur kaa **death** kab ho gaya tha?	Babur kaa *died* kab ho gaya tha?
3	**IMF** kaa primary purpose kya hai?	*Imef* kaa primary purpose kya hai?
4	**Demographics** kya hai?	*Population* kya hai?

Table 6: Some examples from the *HinglishQue* dataset depicting the errors occurred. The **correct** and *incorrect* words in the questions are denoted with **bold** and *italic* fonts, respectively.

formed before the Bi-GRUs in sequence encoding.

6.1 Error Analysis

We analyze the errors encountered by our CMQG and CMQA systems. The CMQG algorithm uses several NLP components such as PoS tagger, NE tagger, translation, transliteration etc. The errors occurred in these components propagate towards the final question generation. We list some of the major causes of errors with examples in Table 6. As in **(1)**, the algorithm could not find the correct lexical translation from the lexical table itself and therefore selected an irrelevant word 'Czechoslovakia' instead of 'occupy'. In **(2)** and **(4)**, the algorithm picked the words 'died' and 'population' instead of 'death' and 'demographics', respectively. It could be because the word 'died' and 'population' have higher n-gram frequencies compared to the words 'death' and 'demographics' in the n-gram corpus. In **(3)**, the system generated incorrect word ('imef') instead of 'IMF'. Here, the Hindi word 'आईएमएफ' is incorrectly tagged as 'Other' instead of 'Organization'. Thereafter, the transliteration system provides an incorrect transliteration ('imef') of the abbreviated Hindi word 'आईएमएफ' (**Trans**: IMF).

We observe that sometimes our CMQA system incorrectly predicts the answer words which are actually very close to some other word in the shared embedding space ((c.f. section 5.4), and hence gets high attention score in the bilinear attention module. For example, in this passage '*...India* was ruled by the **Bharata** clan and ...', the system predicted the answer 'India' instead of 'Bharata' (reference answer) because the word 'Bharata' is the transliteration form of भारत and भारत is the correct translation form of the word 'India'.

Our close analysis to the prediction of CM-SQuAD and CM-MMQA development data reveals that the systems suffer mostly due to the errors where the answer strings are relatively longer. The CM-MMQA dataset has some definitional questions (requires at least one-sentence long answer). We evaluate the performance on CM-MMQA dataset after removing those questions (92), and obtain the EM and F1 scores of 40.50% and 53.73%, respectively. These are much higher (28.14%, 46.25%) than the model where all the questions are considered. Due to ambiguity in selecting answers (between two candidate answers, location type answer) the system sometimes predicts incorrectly. We also observed some other types of errors which were mainly due to the context mismatch as well as long-distance dependence between the answer and the context words.

7 Conclusion

In this work, we have proposed a linguistically motivated unsupervised algorithm for CMQG and a neural framework for CMQA. We have proposed a bilinear attention and answer-type focused neural framework to deal with CMQA. We have evaluated the performance of CMQG on manually created code-mixed questions involving English and Hindi. For CMQA, we have created two CMQA datasets. Experiments show that our proposed models attain state-of-the-art performance. In the future, we would like to scale our work for other code-mixed languages.

8 Acknowledgment

Asif Ekbal acknowledges the Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia). We would like to thank Sukanta Sen, IIT Patna for extending his support in machine translation experiment.

References

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-NER: Massive Multilingual Named Entity Recognition. *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, British Columbia, Canada, April 30 - May 2, 2015.*

Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of Question Generation From Sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473.*

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code Mixing: A Challenge for Language Identification in the Language of Social Media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 13–23.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling Biological Processes for Reading Comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606.*

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Khyathi Raghavi Chandu, Manoj Chinnakotla, Alan W Black, and Manish Shrivastava. 2017. WebShodh: A Code Mixed Factoid Question Answering System for Web. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 104–111. Springer.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas. Association for Computational Linguistics.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level Language Identification using CRF: Code-switching Shared Task Report of MSR India System. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 73–79.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555.*

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and psychological measurement*, 20(1):37–46.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-Attention Neural Networks for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–602. Association for Computational Linguistics.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A Modified Joint Source-Channel Model for Transliteration. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 191–198. Association for Computational Linguistics.

Björn Gambäck and Amitava Das. 2014. On Measuring the Complexity of Code-Mixing. In *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India*, pages 1–7.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2018a. A Deep Neural Network based Approach for Entity Extraction in Code-Mixed Indian Social Media Text. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Deepak Gupta, Surabhi Kumari, Asif Ekbal, and Pushpak Bhattacharyya. 2018b. MMQA: A Multi-domain Multi-lingual Question-Answering Framework for English and Hindi. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Deepak Gupta, Ankit Lamba, Asif Ekbal, and Pushpak Bhattacharyya. 2016a. Opinion Mining in a Code-Mixed Environment: A Case Study with Government Portals. In *International Conference on Natural Language Processing*, pages 249–258. NLP Association of India.

Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Jain, and Shubhashis Sengupta. 2018c. Can Taxonomy Help? Improving Semantic Question Matching using Question Taxonomy. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 499–513. Association for Computational Linguistics.

Deepak Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2016b. A Hybrid Approach for Entity Extraction in Code-Mixed Social Media Data. *MONEY*, 25:66.

Deepak Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2017. SMPOST: Parts of Speech Tagger for Code-Mixed Indic Social Media Text. *arXiv preprint arXiv:1702.00167*.

Michael Heilman and Noah A Smith. 2010. Good Question! Statistical Ranking for Question Generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-Speech Tagging for Code-Mixed English-Hindi Twitter and Facebook Chat Messages. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 239–248.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics, COLING 2002*, pages 1–7. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Anutosh Maitra, Shubhashis Sengupta, Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Mukhopadhyay Abhisek, and Tom Jain. 2018. Semantic Question Matching in Data Constrained Environment. In *Proceedings of the 21st International Conference on Text, Speech and Dialogue (TSD-2018)*.

Christof Monz and Bonnie J Dorr. 2005. Iterative Translation Disambiguation for Cross-language Information Retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 520–527. ACM.

Gillian Moss. 1992. Cognate Recognition: Its Importance in the Teaching of ESP Reading Courses to Spanish Speakers. *English for specific purposes*, 11(2):141–158.

Carol Myers-Scotton. 1997. *Duelling Languages: Grammatical Structure in Codeswitching*. Oxford University Press.

Carol Myers-Scotton. 2002. *Contact Linguistics: Bilingual Encounters and Grammatical Outcomes*. Oxford University Press on Demand.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Khyathi Chandu Raghavi, Manoj Kumar Chinnakotla, and Manish Shrivastava. 2015. Answer ka type kya he? Learning to Classify Questions in Code-Mixed Language. In *Proceedings of the 24th International Conference on World Wide Web*, pages 853–858. ACM.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding Language Preference for Expression of Opinion and Sentiment: What do Hindi-English Speakers do on Twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1131–1141.

Nathan Safran. 2015. *Psychology of the Searcher: Patterns in How Searchers Formulate Queries*. Blue Nile Research.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional Attention Flow for Machine Comprehension. *arXiv preprint arXiv:1611.01603*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598. Association for Computational Linguistics.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. ReasoNet: Learning to Stop Reading in Machine Comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline Bilingual Word Vectors, Orthogonal Transformations and the Inverted Softmax. *CoRR*, abs/1702.03859.

Sainik Kumar Mahata Soumil Mandal and Dipankar Das. 2018. Preparing Bengali-English Code-Mixed Corpus for Sentiment Analysis of Indian Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural Language Comprehension with the EpiReader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Austin, Texas. Association for Computational Linguistics.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS Tagging of English-Hindi Code-Mixed Social Media Content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

Shuohang Wang and Jing Jiang. 2016. Machine Comprehension Using Match-LSTM and Answer Pointer. *CoRR*, abs/1608.07905.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017a. A Joint Model for Question Answering and Question Generation. *CoRR*, abs/1706.01450.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.