*Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 3698–3703
Marseille, 11–16 May 2020


# Linguistically Informed Hindi-English Neural Machine Translation

**Vikrant Goyal, Pruthwik Mishra, Dipti Misra Sharma**
Language Technologies Research Center (LTRC)
IIIT Hyderabad, India
{vikrant.goyal, pruthwik.mishra}@research.iiit.ac.in,
dipti@iiit.ac.in

**Abstract**

Hindi-English Machine Translation is a challenging problem, owing to multiple factors including the morphological complexity and relatively free word order of Hindi, in addition to the lack of sufficient parallel training data. Neural Machine Translation (NMT) is a rapidly advancing MT paradigm and has shown promising results for many language pairs, especially in large training data scenarios. To overcome the data sparsity issue caused by the lack of large parallel corpora for Hindi-English, we propose a method to employ additional linguistic knowledge which is encoded by different phenomena depicted by Hindi. We generalize the embedding layer of the state-of-the-art Transformer model to incorporate linguistic features like POS tag, lemma and morph features to improve the translation performance. We compare the results obtained on incorporating this knowledge with the baseline systems and demonstrate significant performance improvements. Although, the Transformer NMT models have a strong efficacy to learn language constructs, we show that the usage of specific features further help in improving the translation performance.

**Keywords:** Neural Machine Translation, Transformer, Linguistic Features

## 1. Introduction

In recent years, Neural Machine Translation (Luong et al., 2015; Bahdanau et al., 2014; Johnson et al., 2017; Wu et al., 2017; Vaswani et al., 2017) (NMT) has become the most prominent approach to Machine Translation (MT) due to its simplicity, generality and effectiveness. In NMT, a single neural network often consisting of an encoder and a decoder is used to directly maximize the conditional probabilities of target sentences given the source sentences in an end-to-end paradigm. NMT models have been shown to surpass the performance of previously dominant statistical machine translation (SMT) (Koehn, 2009) on many well-established translation tasks. Unlike SMT, NMT does not rely on sub-modules and explicit linguistic features in crafting the translation . Instead, it learns the translation knowledge directly from parallel sentences without resorting to additional linguistic analysis.

Although NMT is a promising approach, it still lacks the ability of modeling deeper semantic and syntactic aspects of the language. In machine translation with a low-resource setting, resolving data sparseness and semantic ambiguity problems can help improve its performance. Addition of explicit linguistic knowledge may be of great benefits to NMT models, potentially reducing language ambiguity and alleviating data sparseness further. Some recent studies have shown that incorporating linguistic features in the NMT model can improve the translation performance (Sennrich and Haddow, 2016; Niehues and Cho, 2017; Li et al., 2018). But most of the previous works have shown the effectiveness of usage of linguistic features with the RNN models. However, it is essential to verify whether the strong learning capability of the current state-of-the-art Transformer models make the explicit linguistic features redundant or if they can be easily incorporated to provide further improvements in translation performance.

Also, there is an immense scope in the development of translation systems which cater to the specific characteristics of languages under consideration. Indian languages are not an exception to this, however, they add certain specifications which need to be considered carefully for effective translation. English and Hindi are respectively reported to be the 3rd and 4th largest spoken languages in the world [1] and this fact makes Hindi-English as an ideal language pair for translation studies. But Hindi-English MT is a challenging task because it's a low resource language pair and both the languages belongs to different language families. Hindi is a morphologically rich language and depict unique characteristics, which are significantly different from languages such as English. Some of these characteristics are the relatively free word-order with a tendency towards the Subject-Object-Verb (SOV) construction, a high degree of inflection and usage of reduplication.

Also, when translating between morphologically rich and free word order languages like Hindi and the other end of morphologically less complicated and word order specific languages like English, the well-known issues of missing words and data sparsity arise; and hence affect the accuracy of translation and leads to more out-of-vocabulary (OOV) words.

In this paper, we present our efforts towards building

[1] http://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

a Hindi to English Neural Machine Translation system using the state-of-the-art Transformer models via exploiting the explicit linguistic features on the source side. We generalize the embedding layer of the encoder in the standard Transformer architecture to support the inclusion of arbitrary features, in addition to the baseline token feature, where the token can either be a word or a subword. We add morphological features, part-of-speech (POS) tags and lemma as input features to Hindi-English NMT model.

## 2. Neural Machine Translation

### 2.1. Encoder-Decoder Framework

Given a bilingual sentence pair $(x, y)$, an NMT model learns its parameters $\theta$ by maximizing the log-likelihood $P(y|x;\theta)$, which is usually decomposed into the product of the conditional probability of each target word: $P(y|x;\theta) = \prod_{t=1}^{m} P_\theta(y_t|y_1, y_2, .., y_{t-1}, x;\theta)$, where $m$ is the length of sentence $y$.

An encoder-decoder framework (Bahdanau et al., 2014; Luong et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) is usually adopted to model the conditional probability $P(y|x;\theta)$. The encoder maps the input sentence $x$ into a set of hidden representations $h$, and the decoder generates the target token $y_t$ at position $t$ using the previously generated target tokens $y_{<t}$ and the source representations $h$. Both the encoder and decoder can be implemented by different structure of neural models, such as RNN (LSTM/GRU) (Bahdanau et al., 2014; Luong et al., 2015), CNN (Gehring et al., 2017) and self-attention (Vaswani et al., 2017). Besides the basic component of the encoder and decoder, a source-target attention mechanism (Bahdanau et al., 2014) is usually adopted to selectively focus on the source representations when generating a target token.

### 2.2. The Transformer Architecture

The Transformer (Vaswani et al., 2017) model is the first NMT model relying completely on self-attention mechanism to compute representations of its input and output without using recurrent neural networks (RNN) or convolutional neural networks (CNN).

RNNs read one word at a time, having to perform multiple steps before generating an output that depends on words that are far away. But it has been shown that the more steps required, the harder it is for the network to learn to make these decisions (Bahdanau et al., 2014). RNNs being sequential in nature, do not effectively exploit the modern computing devices such as GPUs which rely on parallel processing.

The Transformer is also an encoder-decoder model that was conceived to solve these problems. Without using any recurrent layer, the model takes advantage of the positional embedding as a mechanism to encode order within a sentence. The encoder, typically stacks 6 identical layers, in which each of them makes use

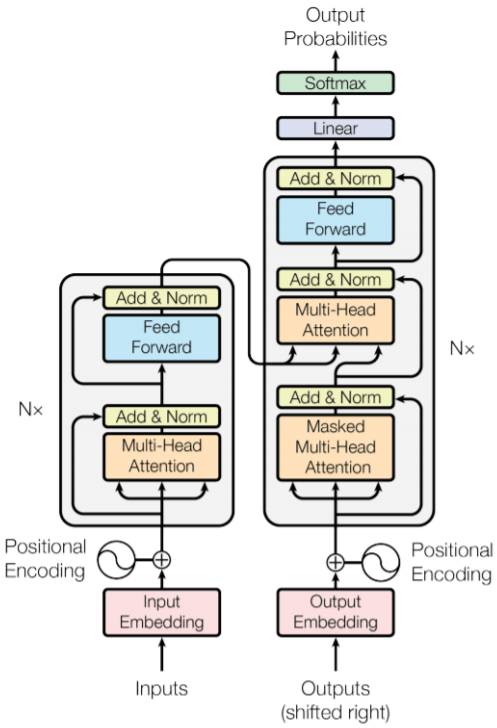


Figure 1: Transformer model architecture from (Vaswani et al., 2017)

of the so called multi-head attention and of a 2 sub-layers feed-forward network, coupled with layer normalization and residual connection (see Figure1). The multi-head attention mechanism computes attention weights, i.e., a softmax distribution, for each word within a sentence, including the word itself. Specifically:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

where the input consists of queries $Q$ and keys $K$ of dimension $d_k$, and values $V$ of dimension $d_v$. The queries, keys and values are linearly projected $h$ times, to allow the model to jointly attend to information from different representation, concatenating the result,

$$Multihead(Q, K, V) = Concat(head_1, ..., head_h)W^o \quad (2)$$

where,

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

with parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W_i^o \in \mathbb{R}^{hd_v \times d_{model}}$.

On top of the multi-head attention there is a feed-forward network that consists of two layers with a ReLU activation in between. Each encoder layer takes as input the output of the previous layer, allowing it to attend to all positions of the previous layer.
The decoder has the same architecture as the encoder,

stacking 6 identical layers of multi-head attention with feed-forward networks. However, here there are two multi-head attention sub-layers: i) a decoder self-attention and ii) a encoder-decoder attention. The decoder self-attention attends on the previous predictions made step by step, masked by one position. The second multi-head attention performs an attention between the final encoder representation and the decoder representation.

To summarize, the Transformer model consists of three different attentions: i) the encoder self-attention, in which each position attends to all positions in the previous layer, including the position itself, ii) the encoder-decoder attention, in which each position of the decoder attends to all positions in the last encoder layer, and iii) the decoder self-attention, in which each position attends to all previous positions including the current position.

### 2.3. Adding Input Linguistic Features

Our main innovation over the standard Transformer encoder-decoder architecture is that we represent its encoder input as a combination of features (Alexandrescu and Kirchhoff, 2006).

Let $E \in \mathbb{R}^{m \times K}$ be the word embedding matrix for the standard Transformer encoder with no input features where $m$ is the word embedding size and $K$ is the vocabulary size of the source language. Therefore, the m-dimensional word embedding $e(x_i)$ of the token $x_i$ (one-hot encoded representation i.e. 1-of-$K$ vector) in the input sequence $x = (x_1, x_2, ..., x_n)$ can be written as:

$$e(x_i) = Ex_i \tag{4}$$

We generalize this embedding layer to some arbitrary number of features $|F|$:

$$\bar{e}(x_i) = merge_{j=1}^{|F|}(E_j x_{ij}) \tag{5}$$

where $E_j \in \mathbb{R}^{m_j \times K_j}$ are the feature embedding matrices with $m_j$ as the feature embedding size and $K_j$ as the vocabulary size of the $j$th feature. Basically we look up separate embeddings for each feature, which are then merged by some merge function. In this work, we experiment with concatenation of separate embeddings (each with some different embedding sizes) for each feature as the merge operation similar to what was done by (Sennrich and Haddow, 2016) in a RNN based attentional NMT system. The length of the final merged vector matches the total embedding size, that is $\sum_{j=1}^{|F|} m_j = m$ and the rest of the model remains unchanged.

## 3. Linguistic Input Features

Our generalized model described in the the previous section supports an arbitrary number of input features, where each of the feature embeddings can also be merged with some merge function other than concatenation. In this paper, we focused on a number of well known linguistic features. The main empirical question that we address in this paper is if providing linguistic input features to the state-of-the art Transformer model improves the translation quality of Hindi-English neural machine translation systems, or if the information emerges from training encoder-decoder models on raw text, making its inclusion via explicit features redundant. All linguistic features are predicted automatically; we use *Hindi Shallow Parser* [2] to annotate Hindi raw text with the linguistic features. We here discuss the individual features in more detail.

### 3.1. Lemma

In a normal NMT model each word form is treated as a token in itself. This means that the translation model treats, say, the Hindi word *pustak* (book) completely independent of the word *pustakein* (books). Any instance of *pustak* in the training data does not add any knowledge to the translation of *pustakein*. In the extreme case, while the translation of *pustak* may be known to the model, the word *pustakein* may be unknown and the system will not be able to translate it. While this problem does not show up as strongly in English due to the very limited morphological inflection in English, it does constitute a significant problem for morphologically rich languages such as Hindi, Telugu, Tamil etc. Lemmatization can reduce data sparseness, and allow inflectional variants of the same word to explicitly share a representation in the model. In principle, neural models can learn that inflectional variants are semantically related, and represent them as similar points in the continuous vector space (Mikolov et al., 2013). However, while this has been demonstrated for high-frequency words, we expect that a lemmatized representation increases data efficiency. We verify the use of lemmas in both word based model and also in a subword model.

### 3.2. POS Tags

Linguistic resources such as part-of-speech (POS) tags have been extensively used in statistical machine translation (SMT) frameworks and have yielded better performances. POS tags provide the linguistic knowledge and the syntactic role of each token in the context, which helps in information extraction and reducing data ambiguity. However, usage of such linguistic annotations has not been explored much in neural machine translation (NMT) systems.

### 3.3. Morphological Features

Machine Translation suffers data sparseness problem when translating to/from morphologically rich and complex languages such as Hindi. Thus morphological analysis may help to handle data sparseness and improve translation quality. Different word types in

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

Hindi have different sets of morph features. For example, verbs have person, number, gender, tense, aspect and nouns have case, number, gender. For some words the features may also be underspecified. Therefore, we treat the concatenation of all morphological features of the word as a string and treat this string as a separate feature value for each word along with other linguistic features.

### 3.4. Using Word-level Features in the Subword Model

Neural Machine Translation relies on first mapping each word into the vector space, and traditionally we have a word vector corresponding to each word in a fixed vocabulary. Addressing the problem of data scarcity and the hardness of the system to learn high quality representations for rare words, (Sennrich et al., 2015) proposed to learn subword units and perform translation at a subword level. Subword segmentation of words is achieved using Byte Pair Encoding (BPE) which has been shown to work better than UNK replacement techniques. In this work, we also experiment with subword models. With the help of BPE, the vocabulary size is reduced drastically, thereby decreasing the OOV (out of vocabulary words) rate and we no longer need to prune the vocabularies. After the translation, we do an extra post processing step to convert the target language subword units back to normal words. We find this approach to be very helpful in handling rare word representations when translating from Hindi to English.

But we note that in BPE segmentation, some subword units are potentially ambiguous, and can either be a separate word, or a subword segment of a larger word. Also, text is represented as a sequence of subword units with no explicit word boundaries. Explicit word boundaries are potentially helpful to learn which symbols to attend to, and when to forget information in the Transformer layers. We use an annotation of subword structure similar to popular IOB format for chunking and named entity recognition, marking if a subword unit in the text forms the beginning (B), inside (I), or end (E) of a word. A separate tag (O) is used if a subword unit corresponds to the full word. To incorporate the word-level linguistic features in a subword model, we copy the word's feature values to all of its subword units.

## 4. Experimental Settings

### 4.1. Dataset

In our experiments, we use IIT-Bombay (Kunchukuttan et al., 2017) Hindi-English parallel data. The training corpus consists of data from mixed domains. There are roughly 1.5M samples in the training data from diverse sources, while the development and test sets are from news domains.

Table 1: Statistics of our processed parallel data.

| Dataset | Sentences | Tokens |
|---|---|---|
| IITB Train | 1,528,631 | 21.5M / 20.3M |
| IITB Test | 2,507 | 62.3k / 55.8k |
| IITB Dev | 520 | 9.7k / 10.3k |

### 4.2. Data Processing

We use Moses (Koehn et al., 2007) toolkit for tokenization and cleaning the English side of the data. Hindi side of the data is first normalized with Indic NLP library[3] followed by tokenization with the same library. As our preprocessing step, we remove all the sentences of length greater than 80 from our training corpus and lowercase the English side of the data. We use BPE segmentation with 32k merge operations. We use *Hindi Shallow Parser* [4] to extract all the linguistic features (i.e POS tags, morph features, lemma) and annotate Hindi text with the same. We also remove all punctuations from both Hindi and English data to avoid any possible errors thrown by the shallow parser. All the linguistic features are joined with the original word or a subword using the pipe ("|") symbol.

### 4.3. Training Details

For all of our experiments, we use OpenNMT-py (Klein et al., 2018) toolkit. We use Transformer model with 6 layers in both encoder and decoder each with 512 hidden units. The word embedding size is set to 512 with 8 heads. The training is done in batches of maximum 4096 tokens at a time with dropout set to 0.3. We use Adam optimizer to optimize model parameters. We validate the model every 5,000 steps via BLEU (Papineni et al., 2002) and perplexity on the development set.

Table 2: Size of embedding layer of linguistic features, in a system that includes all features and contrastive experiments that add a single feature over the baseline.

| Features | Embedding size | |
|---|---|---|
| | all | single |
| subword tags | 6 | 5 |
| POS tags | 10 | 10 |
| Morph Features | 20 | 20 |
| Lemma | 100 | 150 |
| Word or subword | * | * |

The embedding layer size of the all the linguistic features used varies, and is set to bring the total embedding layer size to 512 so as to ensure that the performance improvements are not simply due to an increase in the number of model parameters. All the features have different vocabulary sizes and after performing various experiments we found the optimum embedding

[3] https://anoopkunchukuttan.github.io/indic_nlp_library/

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

size for each of the features listed in table 2, which is basically a hyperparameter in our setting.

## 5. Results

We report the results of usage of linguistic features both in a normal word based model and also in a subword model. We also perform contrastive experiments in which only a single feature is added to the baseline. Table 3 shows our main results for Hindi-English word based model. All the linguistic features added in isolation proved to be effective in improving the translation performance of the word based model. But the combination of all linguistic features together gave us the lowest improvement of 0.19 BLEU. Experiments demonstrates that the gain from the different features is not fully cumulative and the information encoded in different features overlaps.

Table 3: Contrastive experiments for a word based Hindi-English Transformer model with individual linguistic features.

| System | BLEU |
|---|---|
| Word baseline | 17.13 |
| POS tags | 17.51 (+0.38) |
| Lemma | **17.65 (+0.52)** |
| Morph features | 17.44 (+0.31) |
| All features | 17.32 (+0.19) |

Table 4 shows our results for a subword Hindi-English model. Except the lemma feature, all other features used independently in the subword model shows significant BLEU improvements. The reason behind the lemma feature not being helpful in improving the translation performance can be the nature of subword model itself. Translation at subword level inherently captures the linguistic information at the root level. The best performance is achieved when using IOB tags, POS tags and morph features in a subword model.

Table 4: Contrastive experiments for a subword Hindi-English Transformer model with individual linguistic features.

| System | BLEU |
|---|---|
| Subword baseline | 18.47 |
| IOB tags | 18.64 (+0.17) |
| POS tags | 19.11 (+0.64) |
| Lemma | 17.99 (-0.48) |
| Morph features | 19.02 (+0.55) |
| IOB, POS tags and Morph features | **19.21 (+0.74)** |
| All features | 18.34 (-0.13) |

## 6. Related Work

Factored translation models are often used in phrase-based SMT (Koehn and Hoang, 2007) as a means to incorporate extra linguistic information. However, neural MT can provide a much more flexible mechanism for adding such information. Because phrase-based models cannot easily generalize to new feature combinations, the individual models either treat each feature combination as an atomic unit, resulting in data sparsity, or assume independence between features, for instance by having separate language models for words and POS tags. In contrast, we exploit the strong generalization ability of neural networks, and expect that even new feature combinations, e.g. a word that appears in a novel syntactic function, are handled gracefully. Linguistic features have also been used in Neural MT but all of them have shown the effectiveness of usage of linguistic features with the RNN model (Sennrich and Haddow, 2016; Niehues and Cho, 2017; Li et al., 2018). However, it is essential to verify whether the strong learning capability of the current state-of-the-art Transformer models make the explicit linguistic features redundant or if they can be easily incorporated to provide further improvements in performance. Also, the effectiveness of linguistic features in building NMT models for a low resource language pair like Hindi-English where Hindi is a Morphologically rich language has not been shown earlier.

## 7. Conclusion and Future Work

In this paper, we investigate whether the linguistic input features are helpful in improving the translation performance of the state-of-the-art Transformer based NMT model, and our empirical results show that this is the case. We show our results on Hindi-English, a low resource language pair where Hindi is a morphologically rich and a free word order language whereas on the other end we have English which is morphologically less complicated and word order specific language. We empirically test the inclusion of various linguistic features, including lemmas, part-of-speech tags, morphological features and IOB tags for a (sub)word model. Our experiments show that linguistic input features improve both word-based and subword baselines. The subword NMT model with IOB tags, POS tags and morph features outperforms the simple word based NMT baseline by more than 2 BLEU points.

In the future, we expect several developments on the usefulness of linguistic input features in neural machine translation. The machine learning capability of neural architectures is likely to increase, decreasing the benefit provided by the features we tested. Therefore in future, to overcome the data sparsity issue, we will will work on better methods to incorporate linguistic information in such neural architectures.

## 8. References

Alexandrescu, A. and Kirchhoff, K. (2006). Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 1–4. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., et al. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Klein, G., Kim, Y., Deng, Y., Nguyen, V., Senellart, J., and Rush, A. M. (2018). Opennmt: Neural machine translation toolkit. *arXiv preprint arXiv:1805.11462*.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 868–876.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

Kunchukuttan, A., Mehta, P., and Bhattacharyya, P. (2017). The iit bombay english-hindi parallel corpus. *arXiv preprint arXiv:1710.02855*.

Li, Q., Wong, D. F., Chao, L. S., Zhu, M., Xiao, T., Zhu, J., and Zhang, M. (2018). Linguistic knowledge-aware neural machine translation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(12):2341–2354.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Niehues, J. and Cho, E. (2017). Exploiting linguistic resources for neural machine translation using multi-task learning. *arXiv preprint arXiv:1708.00993*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (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.

Sennrich, R. and Haddow, B. (2016). Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2017). Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.