Is Sparse Attention a Better Explanation?

Andriy Drozdyuk [1] **Parsa Vafaie** [2] **Nkechinyere Ogbuagu** [2] **Amirhossein Hajianpour** [2]

Abstract

Explanation in many Natural Language Processing (NLP) models have been attributed to attention mechanisms. The basis of the claim is due to the fact that attention provides an importance distribution over the input data, identifying the input unit(s) responsible for the output. To investigate this claim we experiment with multiple approaches to modifying the attention weights at each layer to test the assumption of transparency. We find that while perturbation of weights does produce an effect on the prediction, it is the change in activation function that contributes more to the attention as explanation. In particular, we find that activation function that produces a more spare attention weight distribution produces better explanation.

1. Introduction

In recent years, there has been an increased demand for explanation i.e how input features produce or contribute to model outputs. This demand has been addressed by *attention*. Attention is a mechanism that provides an importance distribution over the input data. It has been widely adopted in Natural Language Processing (NLP) models like RNNs and claimed to equate to interpretability. However, previous work has proved that different attention weights can produce the same output and therefore should not be considered meaningful explanation (Jain & Wallace, 2019).

The Transformer is a non sequential model that relies entirely on a self-attention mechanism. Its advantage over the sequential models is that it is able to learn long range dependencies and can be run in parallel. It uses multiple self attention heads to simultaneously attend to different contextual word embeddings from different input representations at different positions.

In our work, we are curious to see the effects of different distributions of attention weights on the outputs of the Transformer model. We are also interested in examining the relationship between attention activation functions and attention weights. We pre-train a Transformer model with softmax and entmax activation functions which results in different types of attention weights. This forms the basis of our further experiments.

Following the question "Is Attention Explanation", we apply various approaches for validating the output interpretability using attention weights in transformers. We start with Shuffling, where we randomly rearrange the weights in each layer. Then we proceed to Adversarial Attention where we find adversarial weights such that we get *almost* identical outputs for different weight distributions. Then we use Random-K, where we essentially generate random attention weight distributions. We also generate Opposite Adversarial attention which finds the attention weights that result in the opposite classification result. We conclude with the Change of Activation function where we use a different function during evaluation than training.We find that one method of determining attention weights distribution (1.5-entmax) is more resistant to change than the regular softmax.

2. Backgroud

2.1. Attention

Here we introduce the most general definition of attention[1].

Attention is an operation that selects some largest element from some set X where the notion of largest is represented by some set S of scores:

$$\hat{X} = X \cdot \textit{softmax}(S) \tag{1}$$

Here the set $S = \{s(y_i) | y_i \in Y\}$ where $s : Y \rightarrow R$ is a score function that assigns a score to each $y_i \in Y$. Each y_i is some *evidence* according to which a particular x_i is to be selected. Since s is a function, we can learn it (e.g. represent it as a neural network with some parameter θ):

$$S = s(Y^T; \theta) \tag{2}$$

[1]School of Computer Science, Carleton University, Ottawa, Canada [2]School of EECS, University of Ottawa, Ottawa, Canada. Correspondence to: Andriy Drozdyuk <andriy@drozdyuk.com>.

[1]Our definition of attention is due to lecture slides from Yongyi Mao at University of Ottawa.

In the degenerate case when $X = Y$ the operation is called *self-attention*.

Let us walk through an example. Consider a sequence $X = [x_1, x_2, x_3]$, with the corresponding evidence vector $Y = [y_1, y_2, y_3]$. Suppose that a score function s is given. Let us proceed to calculate attention. First, we must compute our attention weights. Our score function takes our evidence and produces a set of scores S, see Figure 1. We take a softmax of this set and get three attention weights: $\alpha_1, \alpha_2, \alpha_3$.

Now that we have our attention weights, we can proceed to compute attention, see Figure 2. We start by multiplying our original elements x_i by the attention weights. This results in vectors $\hat{x}_1, \hat{x}_2, \hat{x}_3$, which are the weighted versions of vectors x_1, x_2, x_3. To emphasize this we make the borders on some of the boxes thicker, representing a bigger weight. Intuitively this is what we refer to as "how much attention" is paid to a particular value x_i. Finally, we sum all the $\hat{x}_i$ vectors and get our attention value $\hat{X}$.

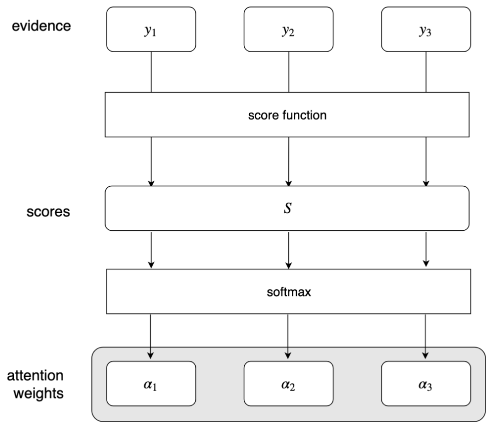

Figure 1. Computing the attention weights. Given some evidence y_i we apply our score function and produce the score set S. We then apply softmax and receive a distribution $\alpha_1, \alpha_2, \alpha_3$ which we call attention weights.

2.2. Explanation

We define explanation as uniqueness of weights for a given model output. So, for example, if we have a set of weights w and we get y as the output, and we change the weights to $w' \neq w$ and get a different output $y' \neq y$ then we say that w is explanation. If we get $y' = y$ then we say that w is not explanation.

2.3. Transformers

Self Attention produces contextual word embeddings, aggregating contextual meaning information into input word embeddings.

Multihead Attention allows the transformer to simultaneously attend to information from different representations at different positions (Vaswani et al., 2017).

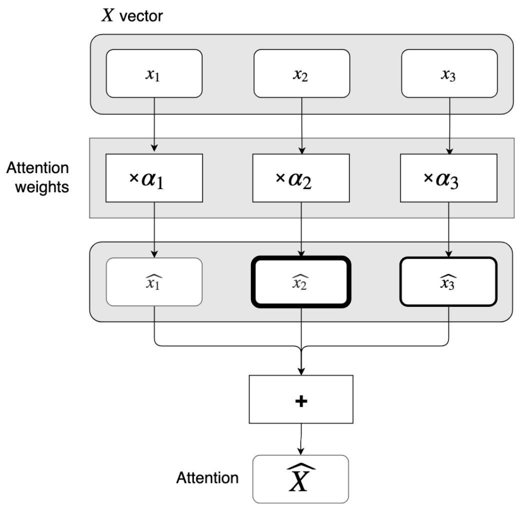

Figure 2. Computing attention. Starting from the top we have: the original vector of vectors $X = [x_1, x_2, x_3]$, the multiplication by attention weights a_i's, the weighted $\hat{x}_i$'s and following a sum the final attention value $\hat{X}$. As an example, we make the border thickness differ for each $\hat{x}_i$ to empathize that a corresponding x_i got more attention (thick border) or less attention (thin border).

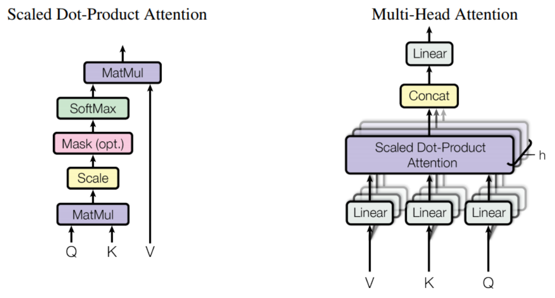

Figure 3. Attention and multi-head attention.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

2.4. Sparsemax

Sparsemax (Martins & Astudillo, 2016) is an activation function similar to softmax, but able to output sparse probabilities.

We start with a definition for a regular softmax:

$$\text{softmax}_i(\mathbf{z}) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

One limitation of softmax is that $\text{softmax}_i(\mathbf{z}) \neq 0$ for every $\mathbf{z}$ and i.

Let $\Delta^{K-1} := \{\mathbf{p} \in \mathbb{R}^K | \mathbf{1}^T\mathbf{p} = 1, \mathbf{p} \geq \mathbf{0}\}$ be the $(K-1)$-dimensional simplex. Softmax is one example of a function that maps from $\mathbb{R}^K$ to probability distribution in Δ^{K-1}. Another one is sparsemax.

Sparsemax is defined as follows:

$$\text{sparsemax}(\mathbf{z}) := \underset{\mathbf{p}\in\Delta^{K-1}}{\text{argmax}} \, ||\mathbf{p} - \mathbf{z}||^2.$$

Sparsemax is similar to softmax with the additional property that it produces sparse distributions.

2.5. Entmax

The α-entmax family is a generalization of sparsemax and softmax (Correia et al., 2019). It is defined as:

$$\alpha\text{-entmax}(z) := \underset{p\in\Delta^d}{\text{argmax}} \, p^T z + H_\alpha^T(p),$$

where $\Delta^d := p \in R^d : \sum_i p_i = 1$ is the probability simples, and, for $alpha \geq 1$, H_α^T is the Tsallis continuous family of entropies:

$$H_\alpha^T(p) := \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j (p_j - p_j^\alpha), & \alpha \neq 1, \\ -\sum_j p_j \log p_j, & \alpha = 1. \end{cases}$$

We can recover our sparsemax function by setting $\alpha = 2$, and softmax by setting $\alpha = 1$. See Figure 4 for an illustration.

For our experiments, we use a value of $\alpha = 1.5$ which is also known as 1.5-entmax (Peters et al., 2019). When we refer to *entmax* in the rest of the paper we mean 1.5-entmax variant of entmax.

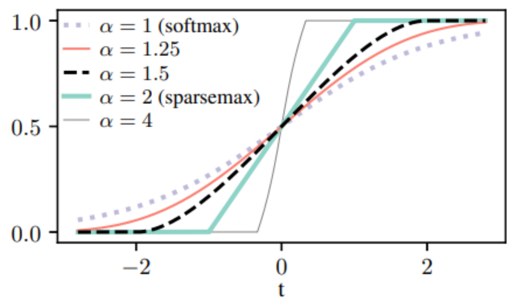

Figure 4. Entmax in the two dimensional case α-entmax. As we can see all functions except softmax saturate at some value of t. Also note that for $\alpha \geq 2$ it is a piecewise linear mapping while $1 < \alpha < 2$ has smooth corners. Figure from (Peters et al., 2019).

2.6. Is Attention Explanation?

While it is possible that a variety of attention weights can result in the same output, answering this question constitutes an expectation that the following property holds: alternative attention weight distributions yield corresponding changes in prediction (Jain & Wallace, 2019). This property assumes that attention weights are *identifiable* (i.e. unique) for each output.

Generally, the identifiability of a set of parameters rests on the absence of collinearity in the input data. However, inputs to NLP tasks are encoded in word embeddings with collinear relations.

Hence, the interpretability of an output is dependent on attention weights, which is in turn dependent on the non-collinearity of the input features.

2.7. Is Attention Explanation in Transformers?

In transformers specifically, attention weights for an attention head are identifiable if they can be uniquely identified from the head's output (Brunner et al., 2019).

The head's output is:

$$\text{Attention}(Q, K, V)H = AEW^V H = AT$$

Self Attention is only identifiable when the dimension of the null space of T is zero, which occurs when the sentence length is less than or equal to attention head dimension.

$$\begin{aligned} \text{rank}(T) &< \min(\text{rank}(E), \text{rank}(W^V), \text{rank}(H)) \\ &\leq \min(d_s, d, d, d_v, d_v, d) \\ &= \min(d_s, d_v). \end{aligned} \tag{3}$$

$$\text{null}(T) = \{\tilde{x}^T \in \mathbf{R}^{1\times d_s} | \tilde{x}^T T = 0\}$$

for $\tilde{A} = [\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_{d_s}]^T$ where $\tilde{x}_i^T$ is any vector in this null space,

$$(A + \tilde{A})T = AT$$

"Due to the Rank Nullity theorem, the dimension of the null space of T is":

$$\dim(\text{null}(T)) = d_s - \text{rank}(T) \geq d_s - \min(d_s, d_v)$$

$$= \begin{cases} d_s - d_v, & \text{if } d_s > d_v \\ 0, & \text{otherwise} \end{cases}$$

3. Methods

Pre-training with Softmax and Entmax. We pre-train a trainsformer encoder model with two types of activations: softmax and entmax. We save each model at the peak of its performance and use it in further experiments below.

Change of Activation. In this experiment we train our Transformer with the entmax function and compare it to the softmax. To train entmax effectively we reduce the learning rate to 0.0001. To be fair we set the softmax learning rate to the same values.

Next, we evaluate the softmax-trained model using entmax during inference. Similarly we evaluate the entmax trained model using softmax.

Adversarial. We train a new set of Keys which produce the same output, but are different from the original set of Keys. Our adversarial model is a simple MLP with a relu activation with a hidden size of 256. It takes in a Keys of shape: (batch, num_layers, seq_len, d_model) and produces a new tensor of the same size. The loss it is trained on is given by:

$$\mathcal{L} = \alpha \mathbf{BCE}(\mathbf{y}, \hat{\mathbf{y}}) - (1 - \alpha)||k - k_{adv}||_2$$

The first term is the classification loss when using the adversarial keys in the transformer and the second term is the euclidean distance between the original and adversarial keys, which is being maximized.

Shuffling. For this experiment we randomly rearrange the entries of K.

Random-K. For this experiment, we use the same adversarial model, but the input K to the adversarial model is generated from a Uniform Random Distribution.

The loss for the Adversarial model is simply the classification loss of the transformer:

$$\mathcal{L}(K_{\text{random}}) = \mathbf{BCE}(\mathbf{y}, \hat{\mathbf{y}})$$

Opposite Adversarial. For this experiment, we train a new set of keys which produce the opposite label of the sample. The loss for this Adversarial model is the classification loss of the transformer.

$$\mathcal{L} = \mathbf{BCE}(\mathbf{1 - y}, \hat{\mathbf{y}})$$

The goal of this experiment is to see how changing the label effects the adversarial weights. We evaluate this on the opposite label from the one it was trained on.

4. Results

4.1. Original Attention Weights

We achieve an accuracy of 85.05% for the entmax function and **86.45**% for the softmax. As we can see the decrease in accuracy for entmax is neglible. See Table 1.

Table 1. Accuracy of the Adversarial models compared to the original.

Activation	Model	Accuracy
Softmax	Original	86.45%
	Change of Activation	79.23%
	Adversarial	83.5%
	Shuffling	79.4%
	Random-K	74.9%
	Opposite Adversarial	82.1%
Entmax	Original	85.05%
	Change of Activation	69.49%
	Adversarial	81.9%
	Shuffling	58.3%
	Random-K	53.6%
	Opposite Adversarial	82.5%

In Figure 5 we can see the comparison between the softmax weights and entmax attention weights. We use this for comparison with all the results that follow.

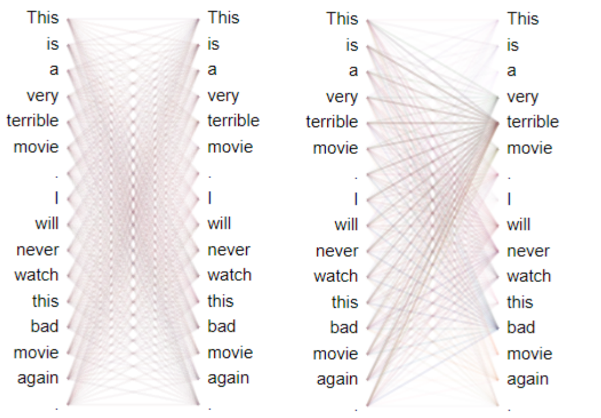

Figure 5. Original attention weights. Left: softmax. Right: entmax

4.2. Change of Activation

The softmax-trained model evaluated using entmax during inference achieves an accuracy of 79.23%. The entmax trained model evaluated using softmax achieves an accuracy of 69.49%. While the entmax-trained model drops in accuracy significantly when using a different attention distribution, the softmax trained model does not. See Table 1.

We visualize the Change of Activation attention weights in Figure 6.

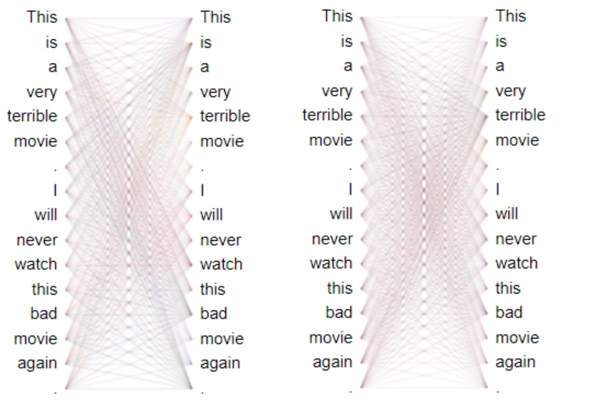

Figure 6. Change of Activation. Left: Model trained with softmax that uses entmax for inference. Right: Model trained with entmax that uses softmax at inference.

4.3. Shuffling

The change in weights of the first layer seems to have more effect on the accuracy. Figure 7 demonstrates the weights under shuffling.

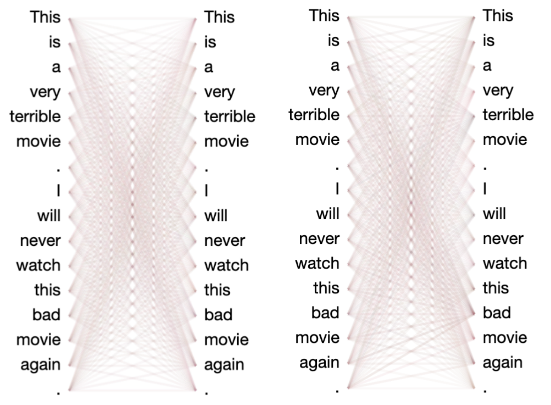

Figure 7. Shuffling of attention weights. Left: softmax. Right: entmax.

4.4. Adversarial

The results demonstrate that we can have a different set of keys for each input, but still get the same classification. Adversarial training seems to focus more on the most relevant words and remove the noise.

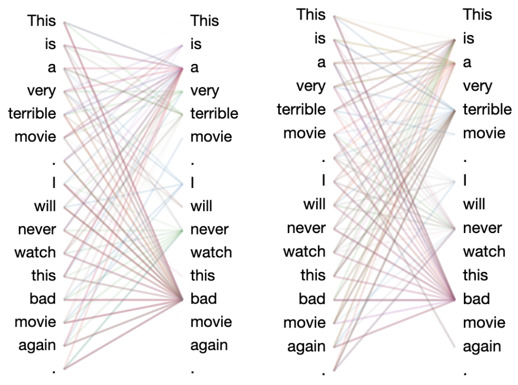

Figure 8. Adversarial attention weights. Left: softmax. Right: entmax.

4.5. Random-K

It can be seen in Figure 9 that the attention focuses on the same words as when using the Adversarial training. It seems that using random uniform distribution further removes noise from the attention weights.

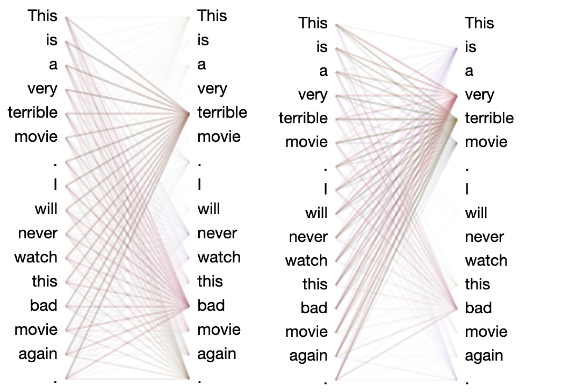

Figure 9. Random-K attention weights. Left: softmax. Right: entmax.

4.6. Opposite Adversarial

We can see in Figure 10 that although the transformer is focusing on negative words, the classification is positive.

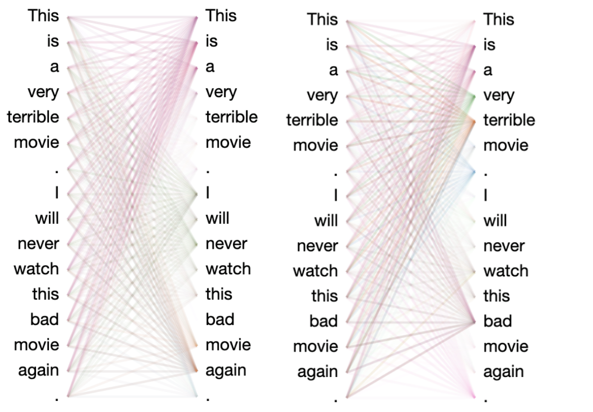

Figure 10. Opposite Adversarial attention weights. Left: softmax. Right: entmax.

5. Discussion

For a model trained with softmax activation function we observe that Adversarial training is able to find a significantly different set of weights while preserving a relatively high accuracy, only a drop of 3%. Shuffling the attention weights on a softmax model reduces the accuracy by 7%. Picking a completely Random-K which results in random weights reduces the accuracy to 74.9% by 11.55%.

For a model trained with softmax activation function we observe that Adversarial training is also able to find a significantly different set of weights while preserving a relatively high accuracy, only a drop of 3%. Shuffling the attention weights on the entmax model however reduces the accuracy by 26.75%. Picking a completely Random-K which results in random weights reduces the accuracy to 31.45%.

In both models if we train the adversarial model to find a K that will classify the opposite of the true class, we find that the accuracy on the swapped labels is close to the original accuracy, 82.1% and 82.5% for softmax and entmax respectively.

These findings reinforce the previous studies that a given set of attention weights is not explanation, since we can, via adversarial training, find another set of weights that yields very similar output while being completely different from the original weights.

According to Table 1 we can see that Change of Activation, Adversarial, Shuffling and Random-K all reduce accuracy more in the case of entmax than in the case of softmax. For example, for the model trained with entmax, the original accuracy was 85.05%. If we Shuffle the attention weights we do significantly worse at 58.3%, which is a reduction in accuracy of 26.8%. However, for softmax, original accuracy being 86.45%, Shuffling only reduces it to 79.4%. This is a reduction of only 7%. The only difference is in the Opposite Adversarial, but since it was trained to classify the opposite class, the results are not really comparable with other weight changes.

Based on this we think that a model trained with entmax is more resistant to attention weight changes. In other words, given a set of attention weights produced by entmax changing any of them will produce a different outcome to a greater degree than in softmax. We therefore speculate that entmax provides a better explanation.

6. Conclusion and Future Work

In this paper we have looked at how explanation is affected by various methods. The five methods we looked at were: changing the activation function, randomly shuffling attention weights, adversarially training another set of attention weights, training a set of weights to predict the opposite class and using a different activation function during inference than during training.

We conclude that changing the activation function contributes more to explanation than changing the attention weights.

For the future work we would like to experiment with 2-entmax (also called sparsemax) which should produce even more sparse attention weight distributions, and perhaps provide even clearer explanation.

We would also like to train a model with more parameters and on a larger dataset to see if this phenomena still persists.

References

Brunner, G., Liu, Y., Parmar, N., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. On identifiability in transformers. *CoRR*, abs/1908.04211, 2019. URL http://arxiv.org/abs/1908.04211.

Correia, G. M., Niculae, V., and Martins, A. F. T. Adaptively sparse transformers, 2019.

Jain, S. and Wallace, B. C. Attention is not explanation. *CoRR*, abs/1902.10186, 2019. URL http://arxiv.org/abs/1902.10186.

Martins, A. F. T. and Astudillo, R. F. From softmax to sparsemax: A sparse model of attention and multi-label classification. *CoRR*, abs/1602.02068, 2016. URL http://arxiv.org/abs/1602.02068.

Peters, B., Niculae, V., and Martins, A. F. T. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1146. URL https://www.aclweb.org/anthology/P19-1146.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.