Federal State Autonomous Educational Institution for Higher Education
National Research University Higher School of Economics

Faculty of Computer Science
EducationalProgram
Applied Mathematics and Information Science

TERM PAPER

Research project

"Uncertainty estimation for Machine Translation"

Prepared by the student of group 171, 3rd year of study
Kuznetsov Dmitriy Sergeevich

Supervisor
research fellow Lobacheva Ekaterina Maksimovna

Moscow 2020

Contents

1 Annotation **3**

2 Keywords **3**

3 Introduction **3**

4 Machine Translation **4**

4.1 Preliminary 4

4.2 Statement 5

5 Beam Search **6**

5.1 Motivation 6

5.2 Method definition 7

5.3 Large Beam Size Problem 8

6 Uncertainty **14**

6.1 Definition 14

6.2 Analyzing Uncertainty in Neural Machine Translation 15

6.3 Ways of uncertainty estimation 17

7 Overview summary **18**

8 Preliminary **19**

9 Misclassification detection **21**

10 Experiment correctness **24**

11 Using on practice **25**

12 Inensemble variance **26**

13 Probability distribution calibration **30**

14 Calibration BLEU **33**

15 Conclusion **34**

1 Annotation

In machine translation, there is a significant problem of improving the quality of models. There are now many studies on the topic of uncertainty estimation. However, there is no almost works on uncertainty estimation in neural machine translation. In this paper, we will study the relationship between uncertainty and the quality of machine translation models.

2 Keywords

machine translation, uncertainty estimation, beam search, larger beam size problem, model uncertainty, data uncertainty

3 Introduction

Nowdays machine translation is one of the most popular areas of machine learning. There are two fundamental approaches to solving this problem: statistical and neural machine translation. In the first case, classical approaches of classical statistical machine learning are used, in the second - modern technologies of deep learning. As in any field, there are a lot of actual unresolved problems in machine translation. In particular, in contrast to statistical machine translation, in neural machine translation quality of prediction degradation with an increase in the width of the hypothesis search appears.

Another relatively modern area in machine learning is uncertainty estimation. Using the methods of uncertainty estimation it is possible to build metrics for confidence in predictions of models. There are several types of uncertainties. Some of them are caused by certain artifacts in the data, while others are caused by the model's ability to understand the proposed data. In any case, the uncertainty estimates indicate that the model is doubtful about making certain predictions.

In this paper, we study the correlation between machine translation model errors and uncertainties in both data and models. We propose an approach to using uncertainty estimation as a way to improve the quality of the model or architecture.

4 Machine Translation

4.1 Preliminary

The task of machine translation is to provide a sentence equivalent in meaning in the target language for the original sentence in the source language. Let $e_1, \dots, e_n$ be a sequence of input sentence tokens' vector representation in the source language. The machine translation model is required to build a sequence of tokens $f_1, \dots, f_m$ in the target language, moreover, in this language, this sequence must have the original meaning.

Most modern neuromachine translation models follow the paradigm Sutskever et al. (2014) (seq2seq). Seq2seq models often consist of two recurrent neural networks or groups of networks. One network that processes input tokens is called *encoder*, the second network that builds output predictions is called *decoder*. The encoder task is to encode the input sequence into some hidden representation $h_1, \dots, h_p$, which is fed to the decoder as hidden initialization before building the output prediction. The decoder output is a vector set $\{(p_{i1}, \dots, p_{id})\}_{i=1}^{m}$, where d - target language dictionary power, m - maximum possible prediction sequence length, p_{ij} - the model's confidence that the ith token of the predicted sentence will be the jth token from the target language dictionary. As a result, the translation model builds a "probability distribution" in the space of the cartesian product of the target dictionary.

The simplest and most obvious way to build a prediction (final translation) is to select the token that the model is most confident in at each step. If $f_1, \dots, f_m$ is a sequence of model hypotheses' tokens, then $f_i := argmax \, \{p_{ij}\}_{j=1}^{d}$. However, for an optimal prediction $f_1^*, \dots, f_m^*$ it is not always true that in terms of model confidence $f_i^* = argmax \, \{p_{ij}\}_{j=1}^{d}$. This phenomenon is also related to the beam problem, which we will discuss in more detail in the following chapters.

4.2 Statement

First of all, let's define the concept of *embeddings*. Embeddinges are some numerical vectors of a special vector space that bijective correspond to words from the dictionary of a certain language. Then consider the following statement. Let $\mathbf{x}, \mathbf{r}$ be embeddings corresponding to the input and output tokens $\mathbf{e}, \mathbf{f}$, respectively. Then the machine translation model sets the following cumulative probability:

$$P(\mathbf{y}|x,\theta) = P(y_1|x,\theta)\prod_{i=2}^{L} P(y_i|y_{<i},\mathbf{x},\theta) \tag{1}$$

Here $\mathbf{y}$ is a certain sequence of tokens that corresponds to the translation from the model's point of view. $y_{<i} = (y_1,\dots,y_{i-1})$, and θ is machine translation model parameters that are configured during training.

Based on these probabilities, a model selects the final embedding sequence $\mathbf{y} = (y_1,\dots,y_m)$ and it bijective restores the final tokens $\mathbf{h} = (h_1,\dots,h_m)$. As we discussed earlier, the deciding rule can be a greedy choice, i.e. at each iteration, an embedding is selected with the maximum conditional probability of $P(y_i|y_{<i},\mathbf{x},\theta)$. Some other approach can be used as the deciding rule, such as beam search, which we will discuss later.

For simplicity, due to bujection, we will further understand tokens as tokens embedding, and we will implicitly mean transferring to the final tokens space. Therefore, our final notation will be: $X = (x_1,\dots,x_n)$ - set of all sentences in the source language, $R = (r_1,\dots,r_n)$ - set of reference translations in the target language. $Y = (y_1,\dots,y_n)$ - a set of translations from the point of view of our model.

The most common quality measure in a machine translation problem is BLEU.

Let the model for some pair of sentences be $\mathbf{x}, \mathbf{r}$ built a translation of $\mathbf{y}$.

$$BLEUK = \min\left(1, \frac{|\mathbf{y}|}{|\mathbf{r}|}\right)\left(\prod_{i=1}^{K} precision_i\right)^{\frac{1}{K}} \tag{2}$$

Here, $precision_i$ means $precision$ calculated for all i-grams.

5 Beam Search

5.1 Motivation

Beam search is used as a more effective solution for building predictions. Its main idea is that instead of selecting the most likely token at each step, we require the model to store b the most likely predictions' prefixes $y_{b1}, \ldots, y_{bt}$. Using this approach, we do not choose the locally optimal token, but in general the optimal prefix, which will increase the chances of building a correct prediction.

It is easy to understand that with the increase of b (this hyperparameter is called *beam size*), the quality of the prediction should also increase, since the number of greedy searches is growing. In practice, it turns out that in neuromachine translation, starting from a certain threshold, the quality decreases significantly with increasing beam size. An illustration of this phenomenon can be seen in Fig. 1. The model used in the experiment is described in the Jonas Gehring (2017).

This problem is called *larger beam size problem*. Next, we will look at the possible causes of this phenomenon in details, until we note that the following situation is not excluded. Suppose on k_1 iteration step beam search stores the prefix $y_1, \ldots, y_{k_1}$ among b locally optimal, which has the least model confidence among all b prefixes. Let this prefix be constructed in such a way that on the next steps model will find a suffix that complements it in such a way that the final sentence is considered the optimal translation from the model's point of view. There are cases when such a translation can be incorrect. Consider following example. Let us train a translation model on a fairy tale corpus, the

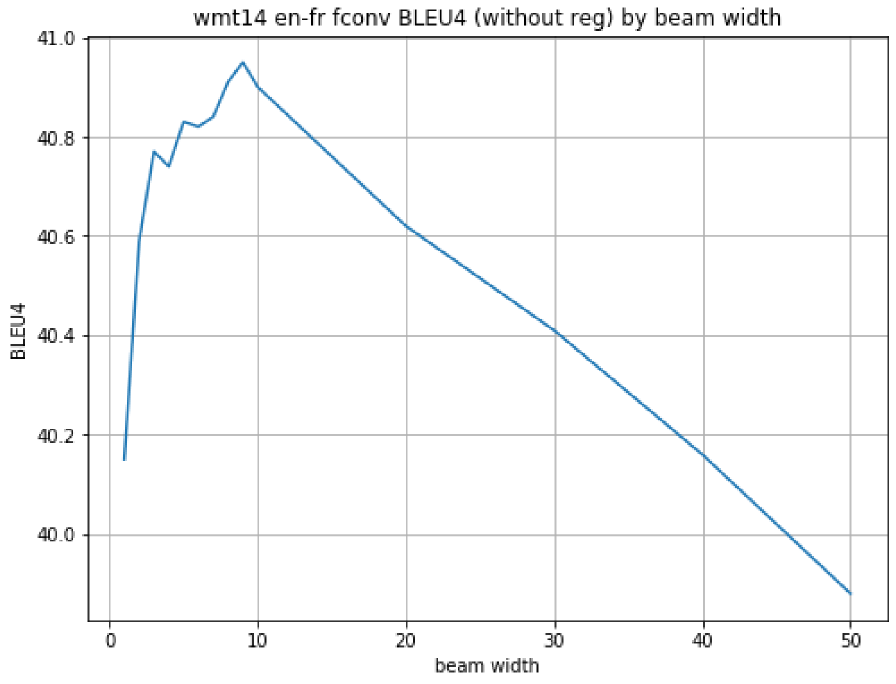

Figure 1 – Convolutional Sequence to sequence model BLEU4 evaluation by beam size. Training set: *WMT'14 English-France*

source language is Russian and the target language is English. Consider the following sentence: *Жульничество на экзамене однажды может закончиться проблемами*. Suppose following sentence is a model prediction: *At the examination once upon a time*. It is obvious that the meaning of the translation does not correspond to the original sentence, but this sentence is optimal from the point of view of our hypothetical model. Beam problem can cause such situations. The prefix *At the examination once* may have been included in a broad beam search and have been earned a low model confidence. However, due to the fact that we train on a large corpus of fairy tales, where there are many examples of sentences like *Once upon a time...* our model decides that after *Once* it is probably to continue prefix with suffix *upon a time*, which lead to a bad translation.

5.2 Method definition

Suggest k-long prediction prefix:

$$y^i_{<k} := (y^i_1, \ldots, y^i_k), \quad \text{where i - beam search branch index} \tag{3}$$

Let us introduce following iterative method:

1 Let us submit a special token of the beginning of the sentence and the encoder hidden state to the decoder input. We get a certain tokens' confidence vector $(p_1^{00}, \ldots, p_d^{00})$, where d - target dictionary power.

2 Take tokens whose confidence values are *beam* highest among $(p_1^{00}, \ldots, p_d^{00})$. Here *beam* is the hyperparameter of the method and is called *beam size.* As a result, we will remember: $y_{<1}^{1}, \ldots, y_{<1}^{beam}$.

3 In the next iteration, we will submit to the decoder $(y_1^{00}, \ldots, y_d^{00})$ as inputs and the hidden state of the previous step. For each token, we get our output distribution $(p_1^{1i}, \ldots, p_d^{1i})$ (here 1 is the iteration number, i is the token number).

4 For all token $i \in \overline{1, beam}$ consider $\forall j \in \overline{1, d}$: $p(f_{:1}^{i}) * p_j^{1i}$. We get a set of prefixes of length 2. Let us choose among all $beam * d$ prefixes *beam* with the greatest confidence. Remember them: $y_{<2}^{1}, \ldots, y_{<2}^{beam}$

5 Iteratively, we will continue the operation $\forall k \in \overline{3, m}$.

We will get: $y_{<k}^{1}, \ldots, y_{<k}^{beam}$

6 As the final prediction select $y_{<m}^{i}, \ldots, y_{<m}^{i}$ such that its confidence is greatest $\forall i \in \overline{1, beam}$

In summary, this method stores *beam* of the "most likely" prefixes at each decoding iteration, in contrast to the naive approach, which greedily selects 1 locally optimal hypothesis at each iteration.

5.3 Large Beam Size Problem

In General, in neuromachine translation tasks large beam size problem is a phenomenon, as a result of which, starting from a certain threshold value, the quality metric decreases with the growth of the beam size. In this section, we will

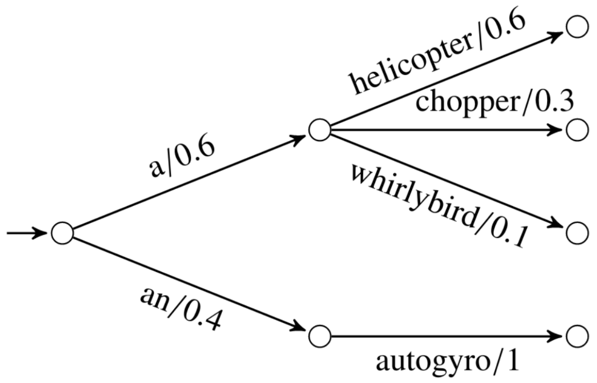

Figure 2 – Label bias causes this toy word-by-word translation model to translate French helicopter incorrectly. This figure is taken from [4].

look at some approaches that do not solve the problem completely, but significantly reduce the effect of metric degradation.

The article Murray et al. (2018) raises two issues: the beam problem and the tendency of NMT models to make short predictions. According to the author, solving the short prediction problem involves solving the beam problem.

As a demonstration of the reasons listed in this article that cause the beam problem, consider the following example from the article.

On the Fig.2. we can see a tree of Beam Search predictions for the word *un hélicoptère*. Let the size of the beam search be 2 and on the first iteration we saved two hypotheses: *a* and *an*. Note that all 4 following hypotheses can be potential translation options, but *a helicopter* in this example is a correct translation. However, *autogyro* is an only one continuation for prefix *an*, thefore the model confidence for this suffix is 1, and as a result, the model confidence to predict *an autogyro* is equal to 0.4. At the same time, the probability mass for the *a* continuation suffixes is divided among the words: *helicopter, chopper, whirlybird*. As a result, the confidence for the correct translation of *a helicopter* from the model's point of view is 0.36. As a result, the model will give a bad prediction. Larger beam size we take, higher probability we get in this situation, hence the loss of quality.

Note that this example demonstrates the fact that if a certain prefix involves a low-entropy suffix, the model tends to ignore the second input token and predict with high confidence one of several highly probable low-entropy suffixes. As a result, the low entropy of the suffix leads to overestimation of this prediction branch and as a result, poor quality. In addition, there is a high probability of getting short predictions due to low entropy, since the model begins to ignore some tokens in favor of high confidence of low-entropy suffixes. For this reason, the authors of the article consider the short prediction problem and the beam problem together.

Standard quality functional for a machine translation task is *cross-entropy*:

$$s(\mathbf{y}) = \sum_{i=1}^{L} \log p(y_i | y_{<i}, \mathbf{x}, \theta), \quad \text{where } L \text{ - translation length} \tag{4}$$

The authors consider the following quality functional adjustments as solutions:

1 Length normalization

$$s'(\mathbf{y}) = s(\mathbf{y})/L \tag{5}$$

2 Google's NMT system (2016). Length normalization.

$$s'(\mathbf{y}) = s(\mathbf{y}) \Big/ \frac{(5+L)^\alpha}{(5+1)^\alpha} \tag{6}$$

3 Word reward

$$s'(\mathbf{y}) = s(\mathbf{y}) + \gamma L \tag{7}$$

Let us look at the results presented in the article for the *WMT'17 Russian-English* model. The architecture presented in the article Bahdanau et al. (2015) was chosen as the baseline. Appealing to experiments results from the paper we observe that the solution of the short prediction problem using methods proposed above significantly decreases the influence of the beam problem on the considered dataset. Comparing *word reward* and length normalization, we can see that they earn quite close BLEU values and lager beam size problem affects models in a same way, when baseline model suffers from beam problem significantly.

Let us now look at the article Koehn et al. (2017). This is a review article on

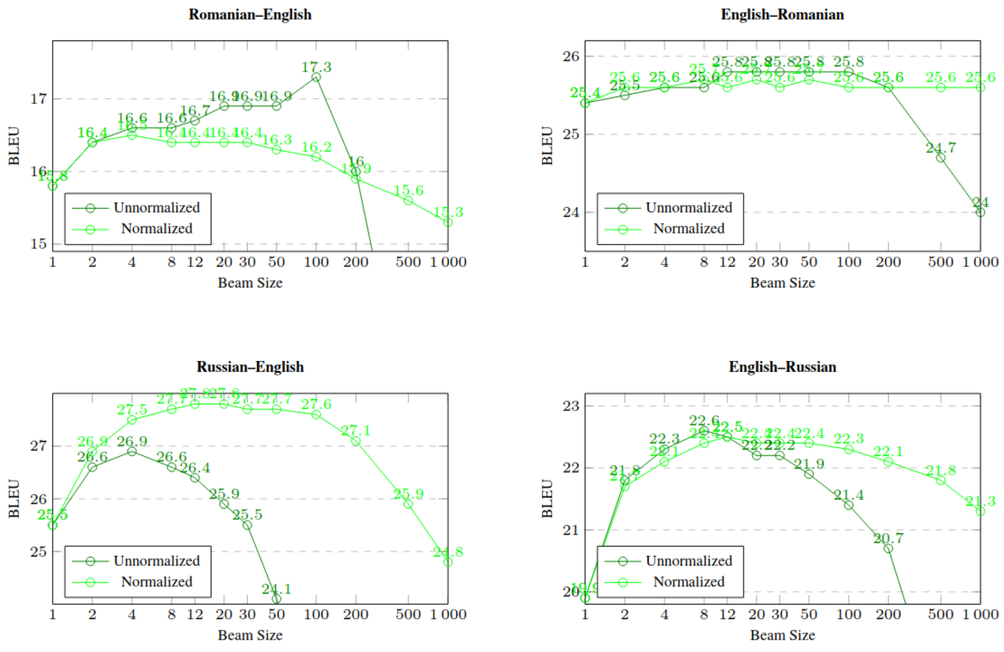

Figure 3 – Translation quality with varying beam sizes. For large beams, quality decreases, especially when not normalizing scores by sentence length. This figure is taken from [7].

the main problems of neuromachine translation, including the beam problem.

The authors of the article do not present new solutions for the problem, but conduct a review study. As a baseline, they use the same neural network as in the previous article: attention-based encoder-decoder. Fig. 3 presents their results for various *WMT* datasets. Here, normalization is the Length normalization from the previous article.

The results obtained in this article are consistent with the results and conclusions of the previous article.

Consider an article that offers a different method for solving the problem Kumar et al. (2019).

In this article the beam problem solution is based on the calibration of the probability distribution $p_{it} := p(y_{i,t}|y_{i,<t})$, here i is the index of the sample example. Before starting the review, let us introduce the definition of "well-calibrated" distribution. Output distribution $p(y_{i,t}|y_{i,<t})$ *well calibrated* if:

$$\forall \beta \in [0,1] : \frac{|\{y \in V \,|\, p(y) = \ beta\}|}{d} = \beta$$

where V is the target language dictionary.

Let us denote calibration metrics **expected calibration error (ECE)**:

Let $\mathbf{r}_i = (r_{i1}, \ldots, r_{im})$ be reference sentece $\mathbf{x}_i$.

Model predictions: $\mathbf{y}_{it} := argmax_y p(y|y_{i,<t})$.

$C_{it}(f) := \delta(r_{it} = y)$, where δ - Kronecker delta.

Let us divide probability space $[0, 1]$ by M equal bins: $I_1, \ldots, I_M$.

Let $L = \sum_i^N |\mathbf{r}_i|$ be a total output token length.

Then:

$$ECE = \frac{1}{L} \sum_{b=1}^{M} \Big| \sum_{i,t:\, p(\mathbf{y}_{it}) \in I_b} C_{it}(\mathbf{y}_{it}) - p(\mathbf{y}_{it}) \Big| \tag{8}$$

The reasons why the probability distribution calibration can solve problems with beam size follow from the examples we discussed above.

The authors consider the classic *temperature scaling as the basic method*:

The method is quite simple. It is necessary to raise all probability distribution to the power of $\frac{1}{T}$, where T is the hyperparameter of the method:

$$p_{it} \to p_{it}^{\frac{1}{T}}$$

This approach does not change the relative values of probabilities, i.e. it does not locally change the optimality of a particular choice, but it helps to smooth out the consequences of overestimating or underestimating certain prefixes or suffixes.

The authors consider 3 models trained on *WMT* as a baseline: attention-based encoder-decoder (2015), GNMT Wu et al. (2016), transformer Vaswani et al. (2017). Authors build probability plots for each model output distribution with and without classic temperature scaling. Results show that transformer model output distribution is closer to perfect calibration than other models, even without scaling. Classic attention-based encoder-decoder, trained on English-Vietnamese WMT dataset, shows the worst calibrated output distribution without using temperature scaling. Other model in a list, trained on other sets of data, shows quite similar results without calibration method and with temerature scaling, this results differ a little from transformer values. Graphs show that using classic temperature scaling probability plot goes to perfect, ECE decreases even in the worst case scenarios.

In order not to overweight the literary review, we will not formally introduce the method proposed by the authors. We will consider conceptual differences from the default temperature scaling. To study the formal definition of the method, you can refer to the article Kumar et al. (2019).

As a result of experiments, the authors found out that end of sequence (EOS) token probability is the worst calibrated among all other tokens. Therefore, first of all, the authors calibrate the distribution of the EOS token. This observation strongly correlates with the conclusions from the articles reviewed earlier. In contrast to the classic temperature scaling and the hyperparameter selection on a validation set, the authors propose to make the hyperparameter flexible, different for each example and token index, i.e. not to smooth the entire probability mass, but to smooth it depending on the offset of the probability density on the token. To find the values of hyperparameters, it is proposed to use two fully connected two-layer neural networks with 3 neurons on the hidden layer. The composition of two neural networks with optimization *Negative Log Likelihood (NLL)* is proposed to train to predict the value of the hyperparameter T for a given example and a token in the prediction.

Authors present experiments results, that comparing ECE and BLEU for baseline models with authors calibration method, classic temperature scaling and without any calibration. In generall we can say that temperature scaling gives significant fall comparing to models without any calibration (about several points of number). Authors' calibration method shows the best results except only one case, ECE of output distribution is less than ECE of temperature scaling models (about several points of number after the dot). It can also be seen that the authors' calibration method achieves the best results on the most part of the baseline models from the BLEU point of view.

Now, turning to Fig. 4, let us look at the effect of the authors' calibration on

Model	B=10	B=20	B=40	B=80
En-Vi NMT	23.8	-0.2	-0.4	-0.7
+ calibrated	24.1	-0.2	-0.2	-0.4
En-De GNMT4	23.9	-0.1	-0.2	-0.4
+ calibrated	23.9	-0.0	-0.0	-0.1
En-De GNMT8	24.6	-0.1	-0.3	-0.5
+ calibrated	24.7	-0.1	-0.4	-0.6
De-En GNMT	28.8	-0.2	-0.3	-0.5
+ calibrated	28.9	-0.1	-0.2	-0.3
De-En NMT	28.0	-0.1	-0.4	-0.6
+ calibrated	28.2	-0.0	-0.2	-0.2
En-De T2T*	26.5	-0.2	-0.7	-1.2
+ calibrated	26.6	-0.1	-0.3	-0.4

Figure 4 – BLEU with increasing beam on the devset. *Beam sizes for Transformer/T2T: 4, 8, 10 and 12. This figure is taken from [8].

the beam size problem. It is easy to see that calibration significantly reduces the impact of the beam problem.

In this section, we investigated the possible causes of the beam problem, as well as methods of counteraction. Unfortunately, it is not possible to draw unambiguous conclusions from the results of the article about which method shows the best results due to the fact that the experiments in the articles were carried out in different configurations. According to approximate estimates, both methods show good results, are easy to implement, but do not solve the problem completely.

In the next section, we look at a different view of the beam problem, from the perspective of uncertainty.

6 Uncertainty

6.1 Definition

This information was obtained from the article Malinin et al. (2018)

Uncertainty in machine learning is usually understood as a measure of the nondegeneration of the distribution on the model predictions at a fixed input. Less formally, if we were able to estimate uncertainty, we could tell how much we can trust the model's predictions.

There are several types of uncertainties: data uncertainty, model uncertainty, and distributive uncertainty. *Data uncertainty (aleatoric uncertainty)* - type of uncertainty caused by the nature of the data, including noise, class balance, etc. *Model uncertainty (epistemic uncertainty)* measures uncertainty caused by a model specifity, how good a model understands given data. *Distributional uncertainty* measures similarity between training and test data, caused by unknown or strange test examples.

It is important for the model to evaluate all types of uncertainty, because they are caused by different reasons and have different negative effects.

6.2 Analyzing Uncertainty in Neural Machine Translation

In the article Ott et al. (2018) *data uncertainty* is being investigated.

Let us introduce several base definition from the article. *Intrinsic uncertainty* is the result of existence of several semantically equivalent translations of the same source sentence, for instance there are several ways to express the same meaning. Also situations when target language is more complex than a source language, it can cause an uncertainty. In such situations it could be impossible to earn additional information, such as gender or tense, which is necessary to build correct prediction in a source language. Web data crawling to construct train corpus can cause *extrinsic uncertainty*. For example, authors notice that at least 1% sentences in WMT datasets is just a copy of a source sentences.

The architecture described in the article Gehring et al. (2017) is specified as the baseline of the experiments.

One of the main problems that is raised in this article is an effect of *uncertainty* on the degradation of quality with the growth of beam size. Authors notices copies (extrinsic uncertainty) are overrepresented in the output of beam search. Also they say that bigger beam size we take, more copies appear in a beam search tree, and this growth is quite significant.

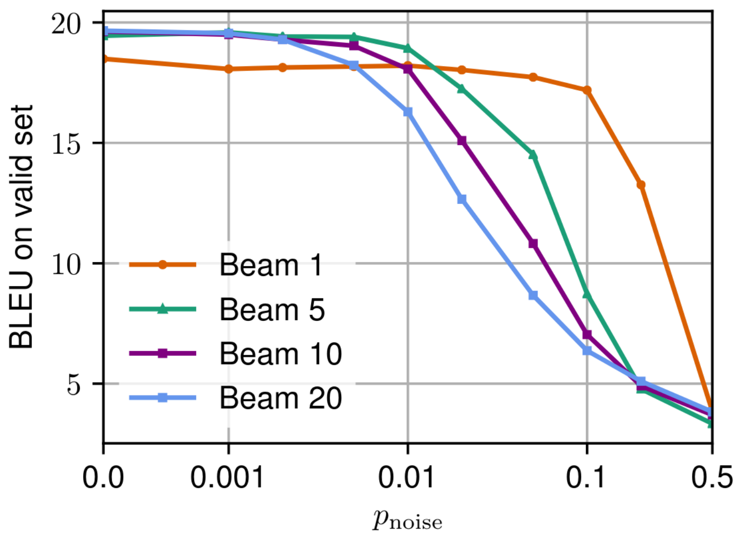

Figure 5 – Translation quality of models trained on WMT'17 English-German news-commentary data with added synthetic copynoise in the training data (x-axis) tested with various beam sizeson the validation set. This figure is obtained from [9].

The authors conducted an experiment, the results of which can be seen in Fig. 5, in which WMT'17 added explicitly replaced random examples from the training with copies of the original sentence. It is clearly seen that this leads to quality degradation with increasing noise. Moreover, models with a larger beam size suffer more from extrinsic uncertainty, which confirms the thesis put forward in the previous paragraph.

The authors suggest two approaches to preparing data for training. First, they remove train examples, which has low score in a model trained on the news-commentary part of WMT'17 En-De point of view (filtered). Second, they restrict beam search hypotheses, which has BLEU with source sentence more than 50% (no copy). The results can be seen in Fig. 6. They are fully consistent with the conclusions obtained earlier.

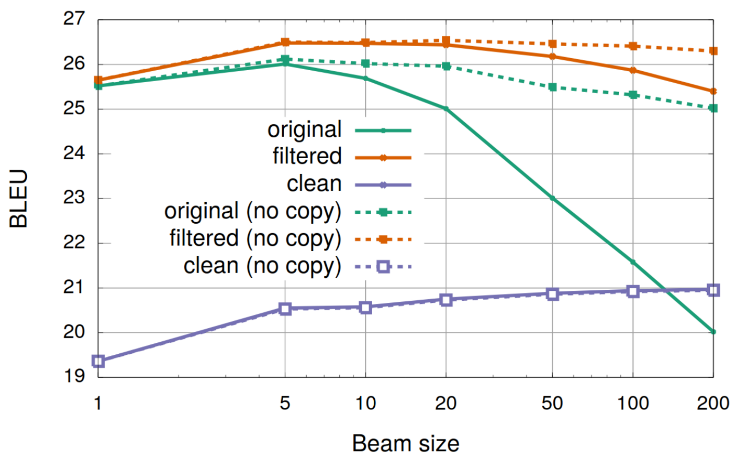

Figure 6 – BLEU on newstest2017 as a function of beam width for models trained on all of the WMT'17 En-De training data (original), a filtered version of the training data (filtered) and a small but clean subset of the training data (clean). They also show results when excluding copies as a post-processing step (no copy). This figure is obtained from [9].

6.3 Ways of uncertainty estimation

In this section, we will refer to the article Malinin et al. (2020), which discusses various approaches to estimating uncertainty for structured predictions and some its applications. In this section, we will consider problems in which solutions are described by the probabilistic model (1).

The author identifies two main approaches to estimating uncertainties for structured predictions: sequence-level and token-level. In the case of sequence-level, the uncertainty is estimated in the space of final predictions (the Cartesian product of the dictionary). In the case of token-level, the uncertainty is evaluated at the level of individual sequence tokens (the dictionary space). Estimating uncertainties at different levels allows us to solve different types of problems.

The main tool for estimating uncertainty in the article is the use of models ensemble from a certain parametric family of models $\mathcal{F}(\theta)$, where $q(\theta)$ - an empirical estimate of a prior distribution of models $p(\theta)$. Informally speaking, the following idea is a prerequisite for using ensembles. If many different models from the same family vote for the same verdict, it is highly probable that we can trust this verdict. In the opposite situation, if the models are not consistent in their predictions, this observation can become an indicator of uncertainty, in particular, it may indicate that the presented problem is out of the general pattern of the data that the models were able to train in their generality.

The starting point for entering uncertainty estimation in the article is the following relation:

$$\underbrace{\mathcal{I}[y, \theta|\mathbf{x}, \mathcal{D}]}_{\text{Knowledge Uncetainty}} = \underbrace{\mathcal{H}[P(y|\mathbf{x}, \mathcal{D})]}_{\text{Total Uncertainty}} - \underbrace{\mathbb{E}_{q(\theta)}[\mathcal{H}[P(y|\mathbf{x}, \theta)]]}_{\text{Data Uncertainty}} \tag{9}$$

Here $\mathcal{H}$ is Shannon entropy and $\mathcal{I}$ - mutual information.

Knowledge uncertainty can also be evaluated using a different metric suggested in the article, which correlates with the one already considered:

$$\mathcal{K}[y, \theta] = \mathbb{E}_{q(\theta)q(\hat{\theta})}[\mathrm{KL}[P(y|\mathbf{x}, \theta)||P(y|\mathbf{x}, \hat{\theta})] \tag{10}$$

We have considered only a small part of the uncertainty metrics proposed by the author. The full study can be found in the original article. We will discuss the metrics described above in more detail in the following chapters.

Malinin in his work presents some experiments demonstrating the effectiveness of uncertainty estimation by these methods at various estimation levels for the ASR problem. In particular, experiments: out of distribution detection and misclassification detection. However, this article does not deal well with uncertainty estimation for a machine translation problem.

7 Overview summary

We have reviewed work workaround. In this paper, we plan to study the correlation between uncertainty and translation errors. In particular, we want to find out how to improve the quality by adding some knowledge about the uncertainty of prediction to the model. Due to the fact that the token-level estimation for the machine translation task is poorly understood, we will consider this approach. We will also try to find out the correlation between the uncertainty estimation and the larger beam size problem.

8 Preliminary

Recall the workaround of our work.

Let $X = (x^1, \dots, x^N)$, $R = (r^1, \dots, r^N)$ a parallel corpus of sentences, where X corresponds to the source language, and R corresponds to the target language (the one we are translating to). The final translation of a certain sentence x is based on the following probabilistic model (our model of neuromachine translation):

$$P(\mathbf{y}|x,\theta) = P(y_1|x,\theta)\prod_{i=2}^{L} P(y_i|y_{<i},x,\theta)$$

The purpose of this work is to use uncertainty estimation (total, knowledge) to determine situations when the model makes an error in predicting the next token y_i. We will look at several approaches and ways to use them.

The baseline model of this work is the transformer Vaswani et al. (2017), implemented in the fairseq Facebook library. Next, in all experiments, we will consider an ensemble of 5 transformers. The final probability model of the ensemble has a form:

$$P(y_i|y_{<i},x,\mathcal{D}) = \frac{1}{K}\sum_{j=1}^{K} P(y_i|y_{<i},x,\theta_j)$$

In our case, $K = 5$. Each model in the ensemble was trained on shared data X, R, but from different initialization points.

IWSLT'14 De-En was taken as a training, test, and validation samples for transformer models. The target metric for this work is *BLEU4*.

The final graphs of the results can be seen at image 7.

An ensemble of 5 models with the Gehring et al. (2017) architecture was also trained. the final metrics for this ensemble can be seen at image 8. As you can

Figure 7 – BLEU on test set for transformer model by beam size

see, the transformer shows higher results, so we will consider the transformer as the final model for experiments.

The configuration of each model in the ensemble of transformers was as follows:

1 $clip\_norm = 0$

2 $learning\_rate = 5 * 10^{-4}$

3 $label\_smoothing = 0.1$

4 $epochs = 75$

5 $lenpen = 0$ (We want to know how uncertainty estimation will affect Bleu degradation by beam size)

The graphs of the final metrics show that with the growth of beam size BLEU degrades. As we found out earlier, this can be caused by overestimating or underestimating the probabilities of certain tokens. In this paper, we will also try to identify the correlation of this problem with the uncertainty.

Figure 8 – BLEU on test set for fconv model by beam size

9 Misclassification detection

In Malinin et al. (2020), an attempt to detect falsely predicted tokens on the ASR problem is considered. Unfortunately, the author did not conduct a similar experiment for NMT due to the complexity of defining the correct target metric in machine translation.

In our work, we decided to explicitly check whether some uncertainty estimates can detect false predictions of the next token.

The structure of the experiment will be as follows. We consider the corpus of sentences X, R. Let $y = (y^1, \dots, y^N)$ - corresponding translations constructed by the ensemble. Let r_i, y_i be ith true translation and model translation tokens, respectively. Enter the following indicator of the ith token correctness in translation:

$$correct(y^i, \mathbf{r}) := [y^i \in \mathbf{r}] \tag{11}$$

This indicator value will be our target function, which we want to predict based on the uncertainty estimation. Now we introduce the *total uncertainty estimation*:

$$TU(y_i) := \mathcal{H}[P(y_i|y_{<i}, \mathcal{D})] \tag{12}$$

Figure 9 – PR-curve for misclassification detection by TU

Here, the Shannon entropy is considered as $\mathcal{H}$. This metric does not separate the types of uncertainties that we have discussed earlier, it is an indicator for any manifestation. It shows how much the probability mass of the final ensemble is sparse by dictionary tokens.

Then for each $y_i^j \in H$ token, we have the target functions $correct(y_i^j, r^j)$ and uncertainty estimates $TU(y_i^j)$. We scale the uncertainty estimates to $[0, 1]$ and get some model that predicts confidence in the error of a particular token prediction.

The results of the experiment can be seen in image 9. If we try to maximize the F1-score, the indicators turned out to be worse than the random predictor. However, if we are intesting only in a small percentage of errors, we can get a relatively good detection accuracy.

The results obtained by us are consistent with the thesis put forward by Malinin. It would seem that we have fixed a very easy correctness indicator (any token that is not in the translation at all is considered as an error), but the uncertainty estimation works worse than a random prediction. The problem lies in the fact that if the translation is a close meaning synonym, in which the ensemble models are strongly confident, then from the point of view of our indicator - this is an error, but the uncertainty estimation will not detect any anomalies. As noted by Malinin in his article, the solution to the problem will be human manual data

Figure 10 – PR-curve for misclassification detection by TU. $\delta(y^i, \mathbf{r})$

markup to identify more meaningful targets in this experiment.

However, we will still try some other approaches and try to compare them with the current. We may be able to use other ratings or similarity indicators to achieve higher quality.

TODO KL + DU

Let's now conduct similar experiments for the new correctness indicator:

$$\delta(y^i, \mathbf{r}) := [y^i = r^i] \tag{13}$$

And consider the results in image 10. This time, there are very good results. However, they are not caused by the fact that we have selected a more relevant correctness indicator, but by the fact that this indicator defines the concept of error more strictly. Now it is more likely to meet a misclassification and as a result, the uncertainty estimates have become easier to deal with this. Therefore, these results are not very reasonable. This is easy to see if we consider the ROC in image 11. Our empirical model does not do much better than a random algorithm. It is important to note that after we strengthened the correctness condition, the percentage of positive class increased significantly (0.76 against 0.26 in the previous statement). In this statement, we can trust ROC, since the

Figure 11 – ROC for misclassification detection by TU. $\delta(y^i, \mathbf{r})$

positive class prevails and as a result, the balance of classes does not worsen results.

TODO KL + DU

Пассаж про beam size

10 Experiment correctness

An important question arises: why a classification error can correlate with the uncertainty estimate TU. Let's try to answer this question.

Let's try to estimate the density of the softmax entropy distribution for two cases: we correctly predict the token and we made the first error in the sentence (we predicted the entire prefix before this token correctly). In the other words, for each such case, we will calculate the softmax entropy and construct empirical density estimates (normalized histograms). The results can be seen in image 12.

As we can see, the densities in both cases are distributed almost normally, unimodally. Moreover, the entropy distribution mode of correct tokens lies strictly to the left of the distribution mode for the first error, i.e. most of the probability mass of correct tokens has a lower softmax entropy (the TU metric).

How can we interpret this? The higher the entropy of the distribution - the

Figure 12 – Ensemble softmax Shannon entropy

distribution is more uniform, i.e. more and more competing hypotheses appear, i.e. our ensemble identifies more and more equally probable variants, namely, less confident in its answer. Conversely, if the entropy is very low, then the distribution is close to singular, i.e. one hypothesis has about 1. confidence from the point of view of the ensemble.

Thus, if the TU is large enough, it can be an indicator that we are not very confident in the token we are predicting, which allows us to estimate uncertainty in prediction.

TODO KL + DU + Beam size

11 Using on practice

Let's assume that we were able to achieve impressive results by constructing some uncertainty estimate $PerfectTU$. How can we use this to improve quality?

Unfortunately, there are not many opportunities. All we can do is mark the positions in the sentences where we are wrong and, as a result, change our predictions. Due to the consistency of machine translation models, it would be logical to build a re-prediction not based on the original model, but to refer to some Oracle that produces more reliable predictions. In practice, such an Oracle could be some assessor or moderator. However, this approach imposes strong restrictions on the use of uncertainty estimation, since not every project can afford the support of human resources. Moreover, if you need to make fast online predictions in a task, you can no longer use moderators.

Thus, we need to change the way we evaluate uncertainty. We need to find a way of estimating so that our model can change the predictions based on the uncertainty assessment itself. Now this is not possible, because the predicted tokens with a fixed prefix have the same uncertainty rating. As our next approach, we will not build estimates for the entire softmax at once, but for each token in the softmax separately.

In the previous sections we have considered KL as an evaluation of knowledge uncertainty. Let's introduce a metric that will correlate with KL, but will be built for each token.

12 Inensemble variance

Recall that for each token, we have denoted probability:

$$P(y_i|y_{<i}, x, \mathcal{D}) = \frac{1}{K}\sum_{j=1}^{K} P(y_i|y_{<i}, x, \theta_j)$$

Thus, for each token, we have: a conditional probability of the token for the ensemble and K conditional probabilities of the token for the models. Let's introduce an unbiased sample variance of the token probability along the models in the ensemble as a measure of uncertainty:

$$inens\_var(y_i) := S^2(P(y_i|y_{<i}, x, \theta_1), \ldots, P(y_i|y_{<i}, x, \theta_K)) \tag{14}$$

This estimate will serve as a measure of model consistency in predicting the probability of a given token (some analog of KL in previous chapters). Note that the entropy criterion is more suitable for us as a measure of uncertainty,

Figure 13 – Inensemble variance among models for correct token and first error token

since using variance it is difficult to separate cases: a singular distribution and a uniform distribution. However, variance still allows you to separate distributions with a high level of competing hypotheses. We choosed variance, because the probabilities of models do not form a probability measure (no distribution is set on these probabilities, the values may be completely inconsistent), so the calculation of entropy is incorrect in this case.

Consider the following graph (image 13). Here is a model of a new uncertainty estimation for two types of tokens: the first false token in the prediction (the prefix before this token was predicted correctly) and the true token for this position with the same prefix. You can immediately notice that, as in the previous case, most of the probability mass of the correct token has lower variance than that of the incorrect token. This observation gives us hope that the models in the ensemble produce more consistent predictions of probabilities for the correct token and more doubt about the probability prediction for the final token.

There is a logical question: why, despite the lower consistency, the final prediction turned out to be the token that turned out to be. To answer this question, we need to look at the probability distributions for these same tokens (image 14).

As we can see from the charts, we make an error in this position, because

Figure 14 – Token-level ensemble probability for correct token and first error token

often we have about zero probability for a true token, so we prefer the more likely token (and its continuation).

We would also like to see how the variance values and probability values relate to each other. Let's go to figure 15. This graph reveals a serious disadvantage of considering variance. Consider the following example. Let us have some sample $t = (t_1, \ldots, t_m)$ and let it correspond to the sample variance: $S^2(t)$. Calculate:

$$S^2(\alpha t) = S^2((\alpha t_1, \ldots, \alpha t_m)) = \alpha^2 S^2(t)$$

Thus, the value of our metric depends not only on the total relation between values in the sample, but also on the absolute values of these values. This is the effect we saw on chart 15. For a higher probability mass of a valid token, the low value of the uncertainty estimate that we selected is not due to the consistency of the models, but rather to the values of probabilities, i.e. the variance of near zero probabilities always lower, because of property above.

However, despite this, by comparing the variance of the correct and false token with equal probabilities, we can observe that the expected variance for the correct token is lower than for the erroneous one. Therefore, although not on the entire probability mass, but on some of its part, our estimation of uncertainty gives

Figure 15 – Token-level ensemble probability by inensemble variance for correct token and first error token

adequate predictions.

In order to make sure of this, you need to see how our rating behaves on other tokens of the same softmax. To do this, we will consider: a random token, tokens with top-3, top-5, and top-20 probabilities in a descending series of variations. The final graphs of the probability distribution and uncertainty estimates can be seen in figure 16.

As we can see, correct tokens are not so hopeless. A high proportion of the probability mass of correct tokens has relatively high probabilities (the expected position in the variation series of probabilities == 2). Moreover, the models ' confidence in the correct tokens is not inferior to the variance of other tokens. We can use these observations to help the model find the correct tokens.

Figure 16 – Token-level ensemble probability and inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens

13 Probability distribution calibration

Our task now is to renormalize the probability distributions of tokens with the uncertainty metric in such a way that the distribution of the correct token becomes more prevalent in terms of choosing the final hypothesis by the model. Thus, the problem is reduced to finding some function g:

$$\hat{p}(y|x, \mathcal{D}) = softmax(g(p(y|x, \mathcal{D}), \; inens\_var(y)))$$

If we look at graph 17, it becomes clear that by selecting g, we can only bend this graph along the x axis. In visual terms, we would like to pull the density of the correct token to the right edge along each contour line of the variance, and the density of any other token to the left.

Remind that we want to penalize for high values of uncertainty, but not so much as to overweight completely irrelevant tokens. A bad example of the g function would be: Здесь и далее положим: $\sigma^2 := inens\_var$

$$g_{naive} := \log p(y|x, \mathcal{D}) - \log(\sigma^2) \tag{15}$$

Figure 17 – Token-level ensemble probability by inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens

We explicitly reward tokens with a very low variance value and, conversely, heavily penalize them for high uncertainty values. This naive approach leads to a diametrically opposite situation (image 18). We can see that such strong penalties lead to a high probability of accepting random tokens. This is because the variance of random tokens is extremely low due to the property of the metric that we discussed earlier. As a result, the model simply produces completely random translations.

Let's try to slightly weaken the penalty for uncertainty, divide the probability by the root of the variance:

$$g_{less\_naive} := \log p(y|x, \mathcal{D}) - \log(\sigma) \tag{16}$$

As we can see from graph 19, this did not bring any success, since we did not even change the relative order of probability distributions.

If we take another look at graph 17, it will be clear that we cannot use linear functions g to rearrange the order of distributions in terms of probabilities (recall that the graph is presented in log scales). Thus, we come to the conclusion that the calibration must not be linear by $\log \sigma^2$, moreover, the peak (maximum point) must fall on a part of the distribution of the correct token, which is not overlapped

Figure 18 – Token-level ensemble probability by inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens for g_{naive}

by any other distribution.

Then let's take a function:

$$g_2 := \log p(y|x, \mathcal{D}) - (\log(\sigma) + \alpha)^2 \tag{17}$$

Where α is a certain hyperparameter that will depend on which point the maximum will be reached. Let's consider a special example (graph 20).

TODO Conclusion

Note that the density of the wrong token still remains to the right of the density of the correct token. Unfortunately or fortunately, we can't do anything with these points, because if a certain token has a high probability on the same contour line of variance, obviously, we should give preference to this particular token. However, even this results can lead to good results, because in the case of a non-single beam size, even a small overweight can lead to the choice of correct hypotheses, because the choice of hypothesis is based on cumulative probabilities, not on a greedy choice.

Figure 19 – Token-level ensemble probability by inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens for g_{less_naive}

Figure 20 – Token-level $softmax(g_2)$ by inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens

14 Calibration BLEU

TODO

15 Conclusion

TODO

References

1. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin. Facebook AI Research. Convolutional Sequence to Sequence Learning. PMLR 70, 2017.

2. Ilya Sutskever, Oriol Vinyals, Quoc V.Le. Sequence to Sequence Learning with Neural Networks. Google. NIPS 2014.

3. Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015.

4. Kenton Murray, David Chiang. Department of Computer Science and Engineering, University of Notre Dame. Correcting Length Bias in Neural Machine Translation. WMT 2018.

5. Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi. Google's Neural Machine Translation System: Bridging the Gapbetween Human and Machine Translation. ArXiv 2016

6. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser. Attention Is All You Need. NIPS 2017.

7. Philipp Koehn, Rebecca Knowles. Six Challenges for Neural Machine Translation. NMT@ACL 2017.

8. Aviral Kumar, Sunita Sarawagi. Calibration of Encoder Decoder Models for Neural Machine Translation. ArXiv 2019.

9. Myle Ott, Michael Auli, David Grangier, Marc'Aurelio Ranzato. Analyzing Uncertainty in Neural Machine Translation. PMLR 80, 2018.

10. Andrey Malinin, Mark Gales. Predictive Uncertainty Estimation via Prior Networks. NeurIPS 2018.

11. Andrey Malinin, Mark Gales. Uncertainty in Structured Prediction. 2020.