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

Machine learning models uncertainty estimation is an important task in different areas. Uncertainty estimation helps to improve system quality, marking inconsistent parts of models behaivior. For such task as ASR or image classification uncertainty estimation is well investigated, but in the machine translation field uncertainty estimation is poorly explored. Machine translation data uncertainty estimation task has been already considered. In this work we investigate total and model uncertainty estimation in neural machine translation for models ensemble. We explore correlation between uncertainty and error in model's predictions.

Index terms - *machine translation, uncertainty estimation, beam search, larger beam size problem, model uncertainty, data uncertainty*

2 Introduction

There are two fundamental approaches in machine translation: statistical and neural machine translation. Neural machine translation models achive state-of-the-art results. As in any field, there are a lot of actual unresolved problems in machine translation. In particular, in neural machine translation quality of prediction degradation appears with a hypothesis search width increase.

Neural models have significant problems with confidence in token probability. *Uncertainty estimation* is a way to measure models‘ confidence level. While we can estimate model‘s confidence, it is possible to re-organise final prediction according to uncertainty estimation.

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 inability to understand the proposed data. In any case, the uncertainty estimates indicate that the model is doubtful about making certain predictions. Moreover, we can use knowledge about uncertainty to improve prediction quality, e.g. as we introduce later it is possible to calibrate output softmax distribution to avoid misclassifications in models‘ predictions.

In this paper, we study the correlation between machine translation model errors and uncertainties. We empiricaly prove the fact, uncertainty estimates can be a misclassification indicator. We explore properties of total, data and model uncertainty estimates and its connection with correct token and error token probabilities.

Moreover, we propose an approach to using uncertainty estimation as a potential way to improve the quality of the model or architecture. We present token probability calibration method based on inesemble models‘ probability variance. It is not improve target quality, but we investigate significant observations, which could be usefull in further works. We explore, how to construct such calibration function, and its influence on token probability distribution.

3 Machine Translation

3.1 Definition

The task of machine translation consists of providing a sentence equivalent in meaning in the target language for the original sentence in the source language. Let $e_1, \ldots, e_n$ be a sequence of an input sentence tokens' vector representation in the source language. The machine translation model is required to build a sequence of tokens $f_1, \ldots, f_m$ in the target language. Moreover, in a target language, the 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 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 encodes an input sequence into some hidden representations $h_1, \ldots, h_p$, which are fed to the decoder as hidden states before building the output prediction. The decoder output is a vector set $\{(p_{i1}, \ldots, p_{id})\}_{i=1}^{m}$, 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. On practice, it is hard to choose final hypothesis using joint tokens‘ probability distributions (too much variants). That is why we construct final hypothesis token by token using conditional probability. Let us consider *autoregressive* model:

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

where $\mathbf{x}$ - input sequence, $\mathbf{y}$ - model‘s final translation, θ - model‘s training parameters, $y_{<i} = (y_1, \ldots, y_{i-1})$.

Autoregressive model constructs final hypothesis step by step. For each token y_i model considers token probability distribution and its cumulative distribution

and choose next translation token according to some rule.

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

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 $\mathbf{y}$. $\mathbf{x}$ - input sentece in a source language and $\mathbf{r}$ - its ground truth translation.

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

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

4 Beam Search

4.1 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.

4.2 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 $(p_1^{00}, \dots, p_d^{00})$, where d - target dictionary power.

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

3 In the next iteration, we will submit to the decoder $(y_1^{00}, \dots, y_d^{00})$ as inputs and the hidden state of the previous step. For each token, we get our output distribution $(p_1^{1i}, \dots, p_d^{1i})$ (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: $y_{<2}^1, \dots, y_{<2}^{beam}$

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

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

6 As the final prediction select $y_{<m}^i, \dots, y_{<m}^i$ 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.

4.3 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

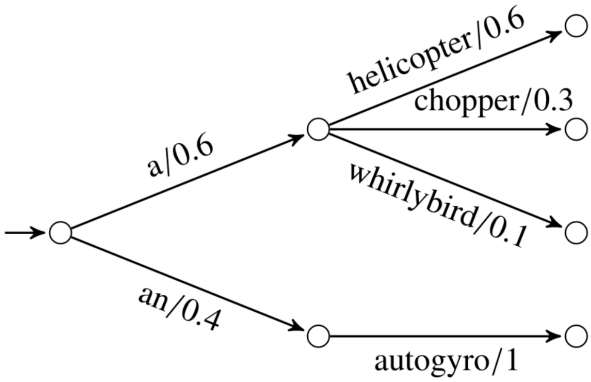


Figure 1 – Label bias causes this toy word-by-word translation model to translate French helicopter incorrectly. This figure is taken from [4].

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. 1. 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.

5 Uncertainty

5.1 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.

5.2 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*

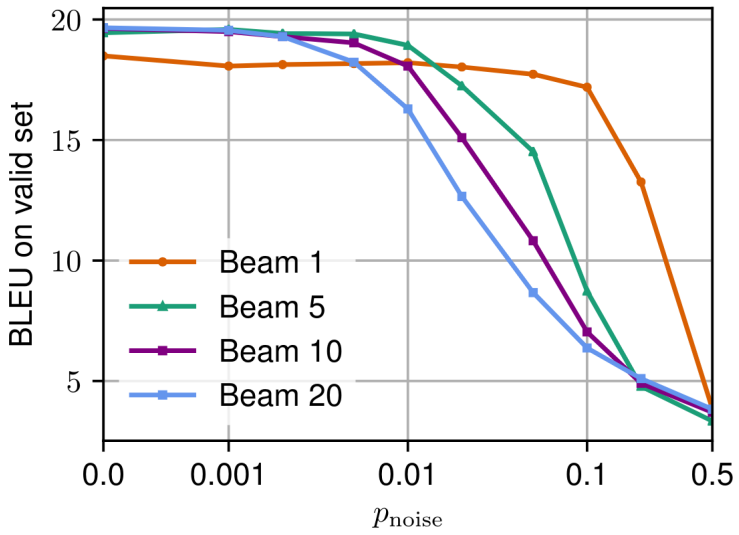


Figure 2 – Translation quality of models trained on WMT'17 English-German news-commentary data with added synthetic copynoise in the training data (x-axis) tested with various beam sizeson the validation set. This figure is obtained from [9].

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.

The authors conducted an experiment, the results of which can be seen in Fig. 2, 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. 3. They are fully consistent with the conclusions obtained earlier.

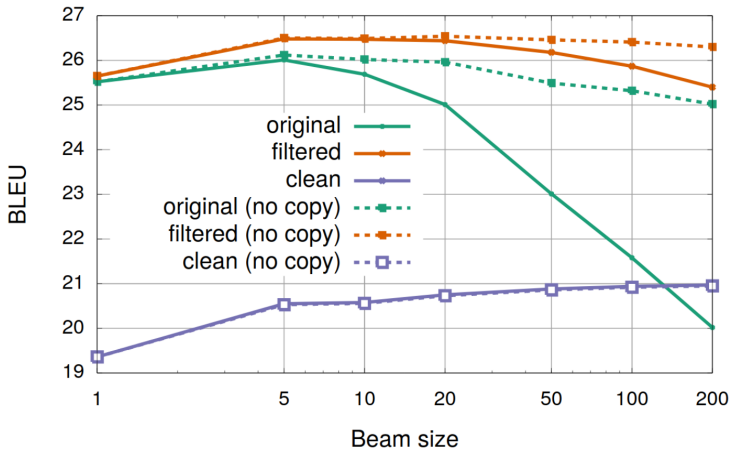


Figure 3 – BLEU on newstest2017 as a function of beam width for models trained on all of the WMT'17 En-De training data (original), a filtered version of the training data (filtered) and a small but clean subset of the training data (clean). They also show results when excluding copies as a post-processing step (no copy). This figure is obtained from [9].

5.3 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:

$$\underbrace{\mathcal{I}[y, \theta | \mathbf{x}, \mathcal{D}]}_{\text{Knowledge Uncetainty}} = \underbrace{\mathcal{H}[P(y|\mathbf{x}, \mathcal{D})]}_{\text{Total Uncertainty}} - \underbrace{\mathbb{E}_{q(\theta)}[\mathcal{H}[P(y|\mathbf{x}, \theta)]]}_{\text{Data Uncertainty}} \tag{4}$$

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:

$$\mathcal{K}[y, \theta] = \mathbb{E}_{q(\theta)q(\hat{\theta})}[\mathrm{KL}[P(y|\mathbf{x}, \theta)||P(y|\mathbf{x}, \hat{\theta})] \tag{5}$$

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.

6 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.

7 Preliminary

Recall the workaround of our work.

Let $X = (x^1, \ldots, x^N)$, $R = (r^1, \ldots, r^N)$ 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 en-

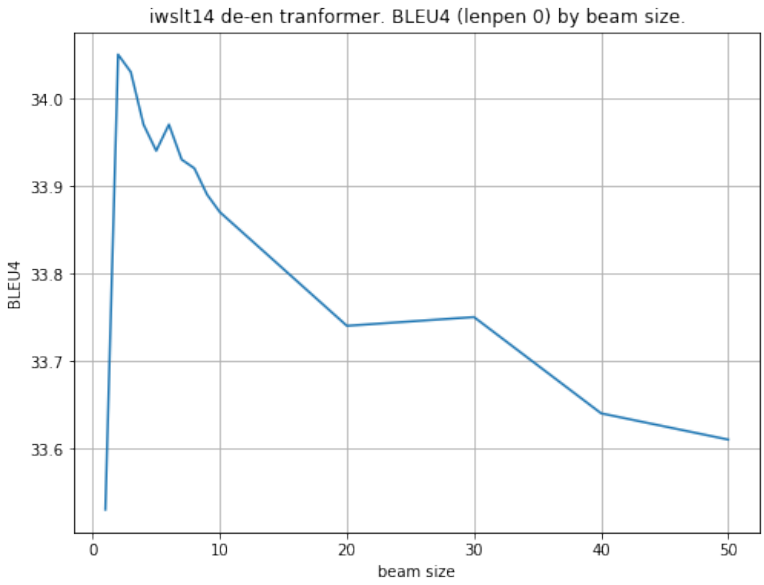


Figure 4 – BLEU on test set for transformer model by beam size

semble 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 graph of the results can be seen on Fig. 4.

An ensemble of 5 models with the Gehring et al. (2017) architecture was also trained. The transformer showes 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.

8 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:

$$correct(y^i, \mathbf{r}) := [y^i \in \mathbf{r}] \tag{6}$$

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

$$TU(y_i) := \mathcal{H}[P(y_i|y_{<i}, \mathcal{D})] \tag{7}$$

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.

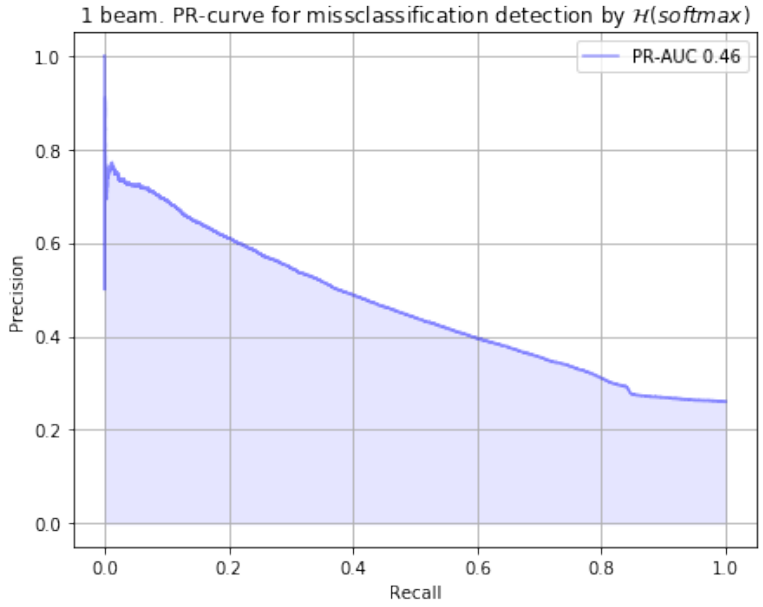


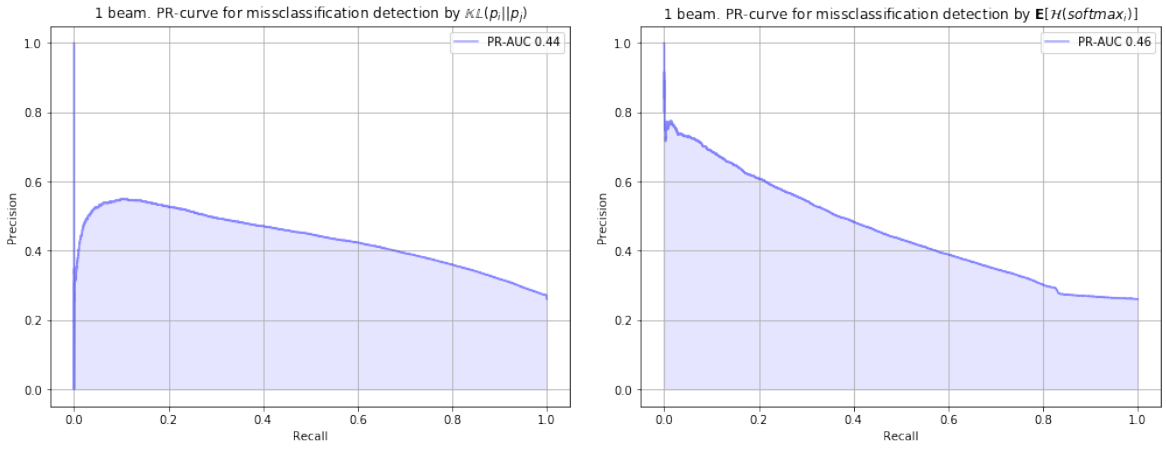
Figure 5 – PR-curve for misclassification detection by TU

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 Fig. 5. If we try to maximize the F1-score, results turned out to be worse than the random predictor. However, if we are intresting 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 easy correctness indicator (any token that is not in the translation at all is marked 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 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



(a) $\mathcal{K}[y, \theta]$ (b) Data uncertainty

Figure 6 – PR-curve for misclassification detection

achieve higher quality.

Malinin considers another way of uncertainty estimation:

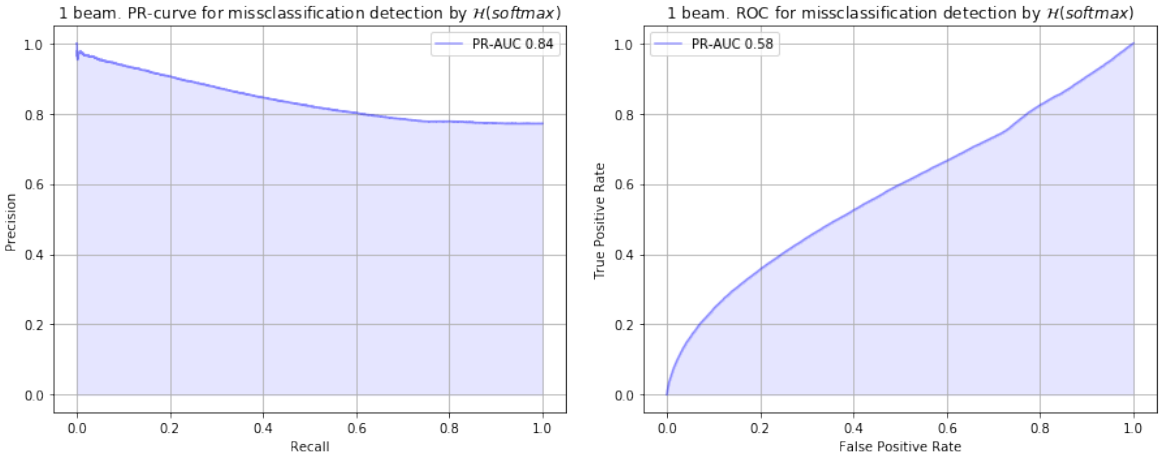
$$\mathcal{K}[y, \theta] = \mathbb{E}_{q(\theta)q(\hat{\theta})}[\mathrm{KL}[P(y|\mathbf{x}, \theta)||P(y|\mathbf{x}, \hat{\theta})]$$

Recall that this metric estimates knowledge uncertainty (model uncertainty). Let us take a look at experiment results for this case (Fig. 6a). As we can see $\mathbb{KL}$ solves misclassification problem worse than previous one. This results conform with the result that Malinin gets in his paper for ASR model (Knowledge uncertainty metrics estimate worse than total uncertainty estimations). Nevertheless, $\mathbb{KL}$ could be effective, if we define more resonable correctness indicator, as a total uncertainty estimation could be.

One more uncertainty estimation, that Malinin consideres in his paper is data uncertainty estimation:

$$\mathbb{E}_{q(\theta)}[\mathcal{H}[P(y|\mathbf{x}, \theta)]]$$

Expirement results, using data uncertainty estimation, are presented in Fig. 6b. Obtained score is as bad as in TU case (as previous one, it conforms with Malinin ASR results). As we can see, data uncertainty estimation gains more score, than model uncertainty estimation. That is why we can suggest that, if considered metrics is resonable, so uncerainty in data influences on model prediction quality



(a) PR-curve (b) ROC

Figure 7 – Misclassification detection by TU. $\delta(y^i, \mathbf{r})$

more than uncertainty in models generalization ability in ensemble.

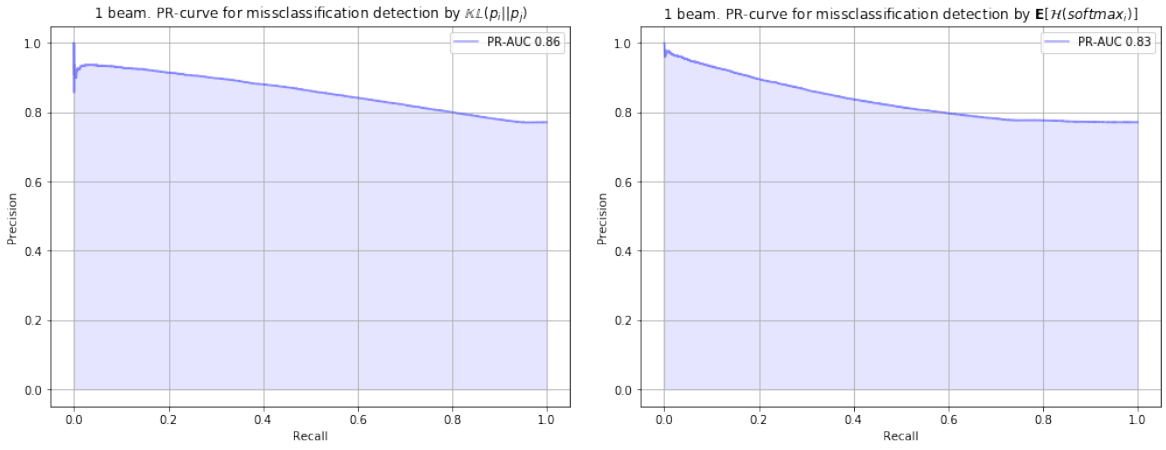
Total uncertainty, data uncertainty and knowledge uncertainty estimates low results caused by same reasons, that we discussed above. That is why let us go further.

Let us now conduct similar experiments for the new correctness indicator:

$$\delta(y^i, \mathbf{r}) := [y^i = r^i] \tag{8}$$

And consider the results in Fig. 7a. This time there are good results. However, they are not caused by the fact that we have selected a more reasonable 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 Fig. 7b. 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 positive class prevails and as a result, the balance of classes does not worsen experiment results.

As for previous correctness indicator, we considers data unceratainty estimate and $\mathbb{KL}$ estimate on Fig. 8.



(a) $\mathcal{K}[y, \theta]$ (b) DU

Figure 8 – Misclassification detection. $\delta(y^i, \mathbf{r})$

In all considered cases results are unreasonable. We can not trust metrics, while we use groundless correctness indicator.

8.1 Experiment correctness

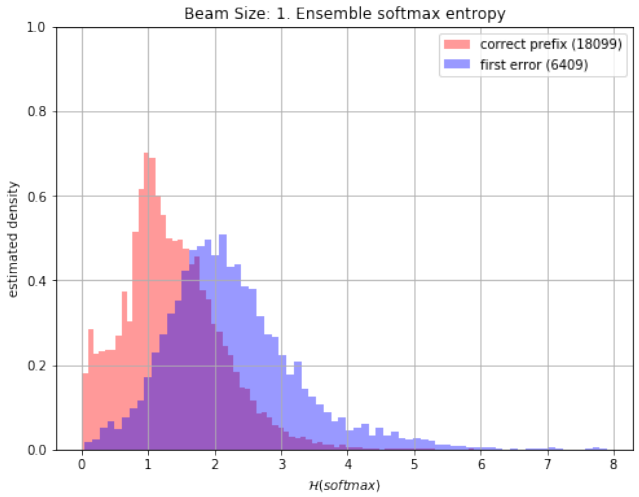
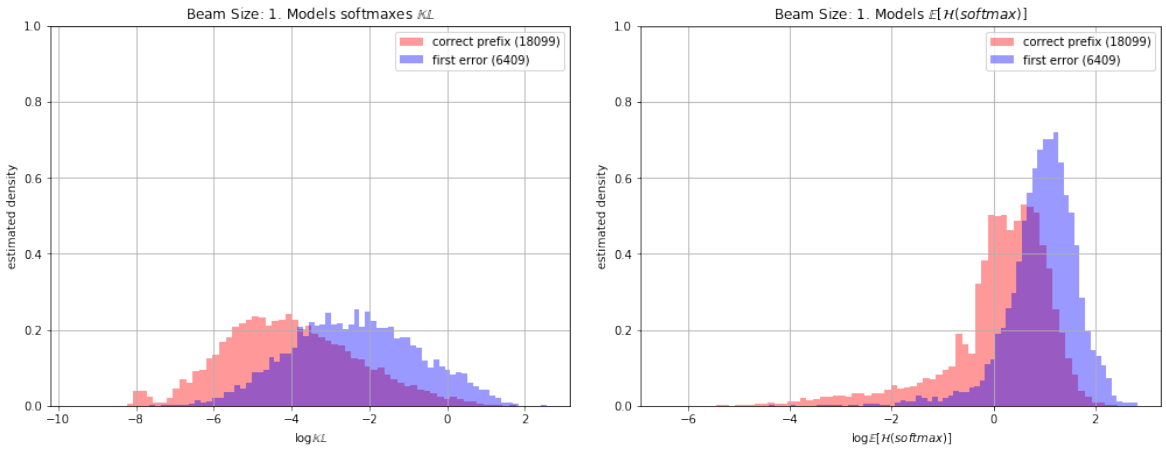


Figure 9 – Ensemble softmax Shannon entropy

An important question arises: why a classification error can correlate with the uncertainty estimate TU.

Let us 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 Fig. 9.

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



(a) $\mathcal{K}[y, \theta]$ (b) DU

Figure 10 – Uncertainty estimates distribution for correct and error tokens

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 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.

Further, let us collect same distribution but for:

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

Firstly, we should interpret this kind of uncertainty estimate. $\mathbb{KL}$-divergence measures difference between two distributions (more - further), this estimate is always non-negative. While we take expectation of estimate, we try to average difference between softmaxes among all possible models pairs. We can interpret $\mathcal{K}[y, \theta]$ as average value of difference between models predictions. Therefore, this metric shows us how models in ensemble conform with each other. Prerequisite to use this kind of uncertainty estimate is that it is possible, more inconsistent

prediction models in ensemble make - higher probability we get to make a mistake on current token. Less formal, $\mathcal{K}[y, \theta]$ correlate with errors in predictions.

Final statistics can be seen at Fig. 10a. As in previous case, correct prefix density separable from first error token distribution. Moreover, expected value of kullback-leibler divergence is less than first error token's one. It meens for us, that correlation we discussed earlier exists and it could be effective to use this kind of uncertainty estimate in misclassification problem.

Our final uncertainty estimate in this section is:

$$DU = \mathbb{E}_{q(\theta)}[\mathcal{H}[P(y|\mathbf{x}, \theta)]]$$

As always, we should explore its meaning. While we evaluate data uncertainty estimate, we consider average single model self-confidence in ensemble. This metric is an attempt to evaluate how common considered situation ($p(y_i|y_{<i}, \mathbf{x}, \theta)$) is, how presented data blends with average model generalization ability. Therefore, DU could be a good measure of data uncertainty. Prerequisite to use DU is same as previous.

Final statistics for DU can be seen at Fig. 10b. We can see that our prerequisite is confirmed.

Overall, we considered three types of uncertainty estimation and emperically proved correlation between all types of uncertainty and predictions errors. But anyway our main barier to use uncertainty estimates in misclassification problem is token correctness indicator.

8.2 Using on practice

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

Unfortunately, there are no many opportunities. All we can do is to mark 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 estimates itself. Now this is not possible, because the predicting 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 us introduce a metric that will correlate with KL, but will be built for each token.

8.3 Beam search

We conduct missclassification detection experiment for different beam size (1, 5, 9, 20, 50). We explore that for both correctness indicator with beam size growth PR-AUC degradate. It is not a significant result, because the reason of this phenomena lies in correctness indicator definition. For examle, if we consider $correct(y^i, \mathbf{r})$, while beam size grows, probability to meet incorrect token in beam search grows, too, because of low positive class ratio (~ 0.25).

Moreover, we do not find any correlation between beam size and uncertainty. As follow, considered uncertainty estimates can not solve larger beam size problem explicitly (in the way we do uncertainty estimation).

9 Inensemble uncertainty estimate

Recall that for each token, we have denoted probability:

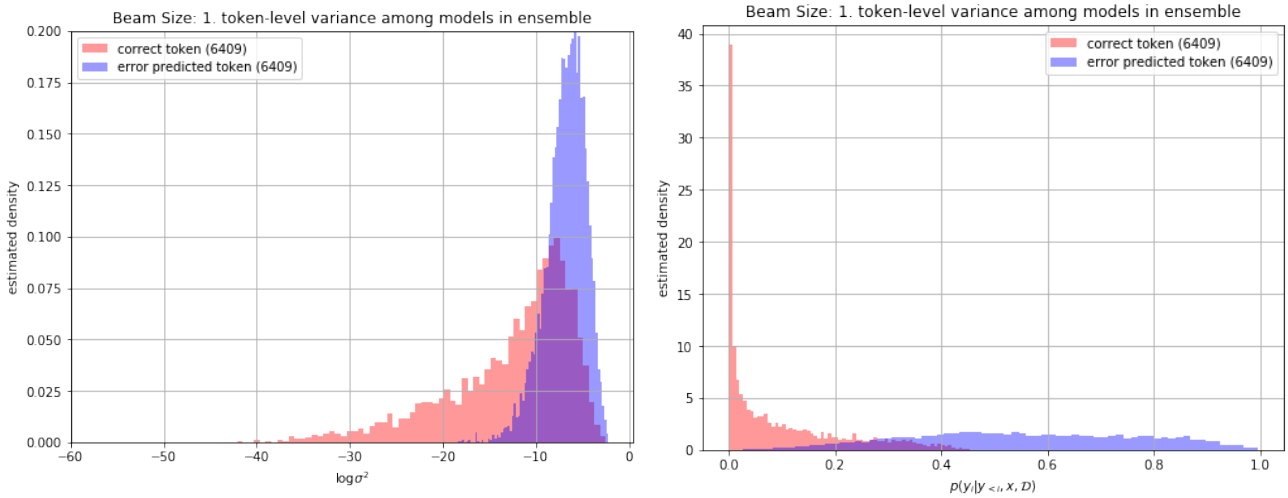
$$P(y_i|y_{<i}, x, \mathcal{D}) = \frac{1}{K}\sum_{j=1}^{K} P(y_i|y_{<i}, x, \theta_j)$$

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 us introduce an unbiased sample variance of the token probability along the models in the ensemble as a measure of uncertainty:

$$inens\_var(y_i) := S^2(P(y_i|y_{<i}, x, \theta_1), \ldots, P(y_i|y_{<i}, x, \theta_K)) \tag{9}$$

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, 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.

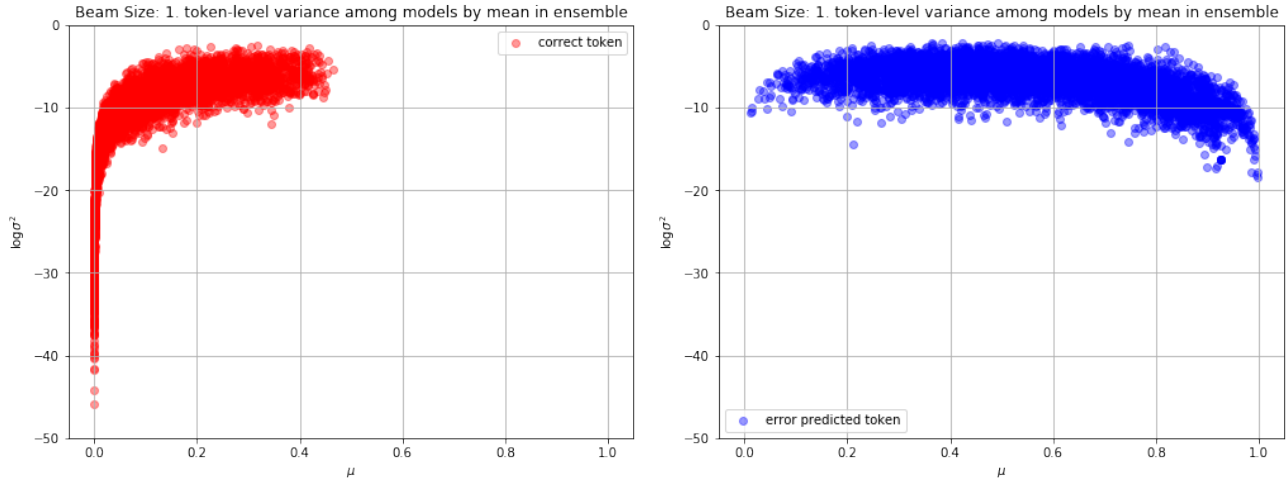
Consider the following graph (Fig. 11a). Here is a model of a new uncertainty estimation for two types of tokens: the first false token in the prediction (the prefix before this token was predicted correctly) and the true token for this position with the same prefix. You can immediately notice that, 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.



(a) Inensemble variance among models

(b) Token probability

Figure 11 – Statistics for correct token and first error token in softmax



(a) Correct token

(b) Error predicted token

Figure 12 – Token-level ensemble probability by inensemble variance for correct token and first error 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 (Fig. 11b).

As we can see from the charts, we make an error in this position, because 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. Refer to Fig. 12. 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 12. 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 false one. Therefore, although not on the entire probability mass, but on some of its part, our estimation of uncertainty gives adequate predictions.

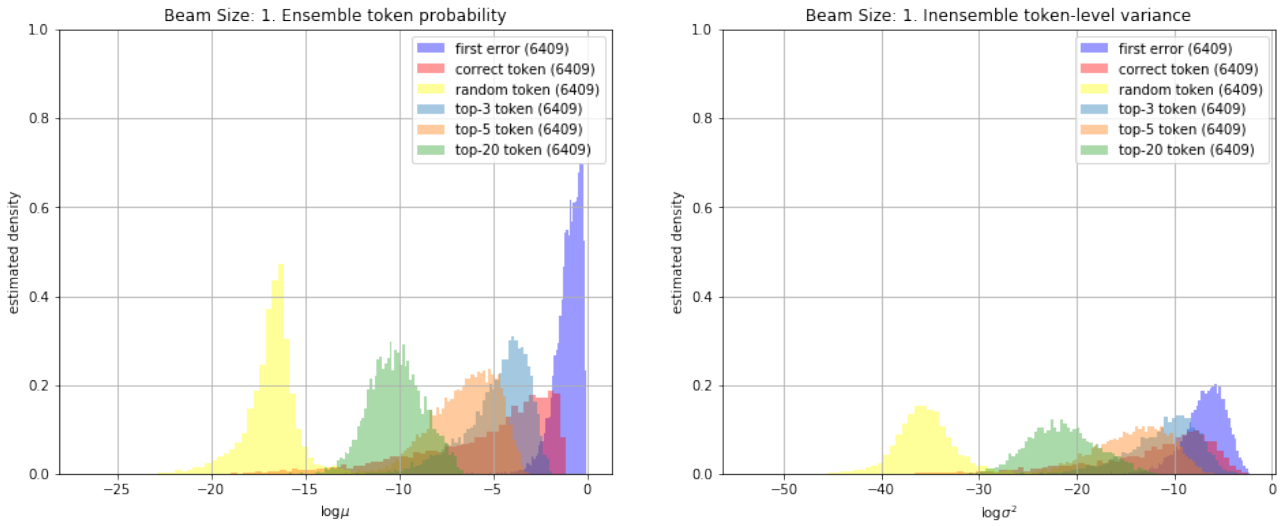


Figure 13 – Token-level ensemble probability and inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens

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 Fig. 13.

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.

9.1 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:

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

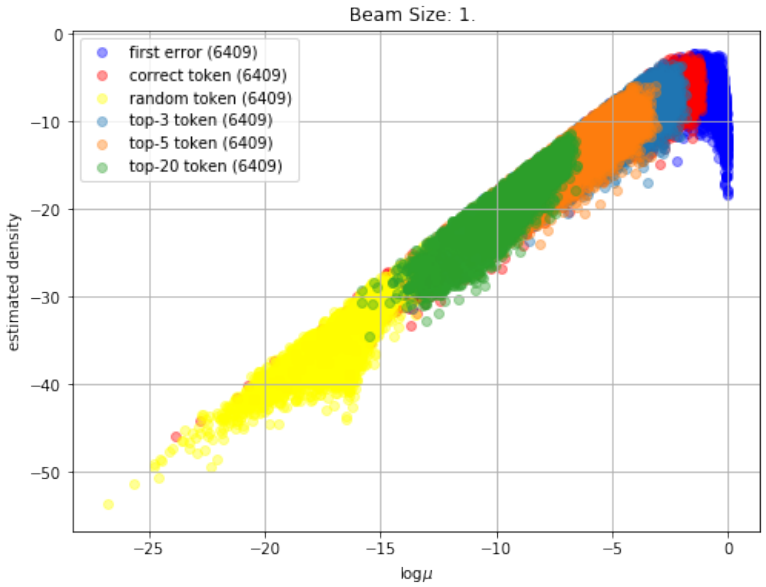
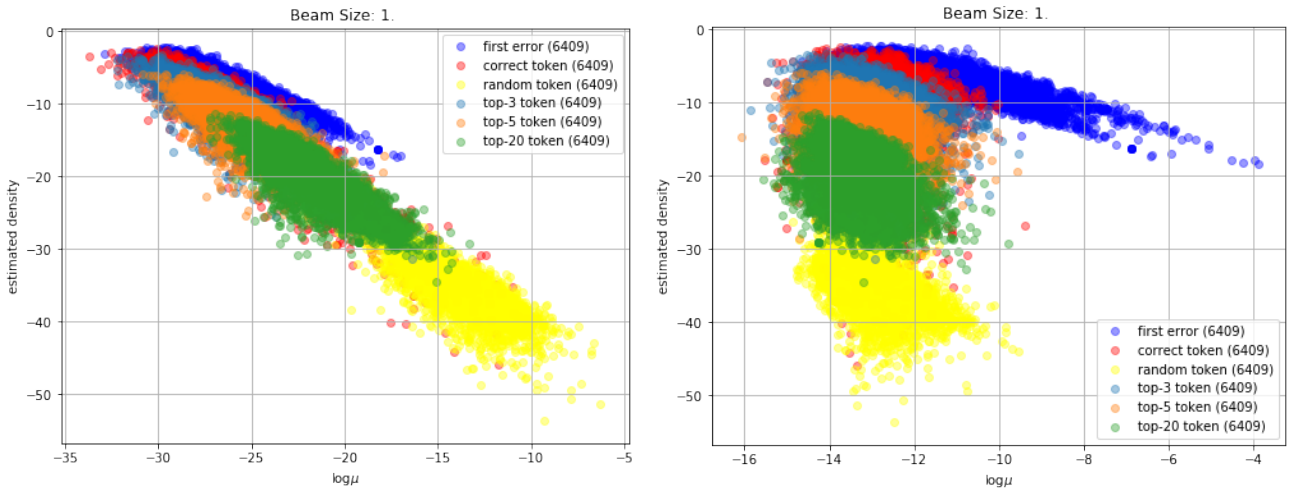


Figure 14 – Token-level ensemble probability by inensemble variance for correct token, first error token, random token, top-3 top-5 and top-20 tokens

If we look at Fig.. 14, 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



(a) $g = g_{naive}$ (b) $g = g_{less\_naive}$

Figure 15 – 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

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:

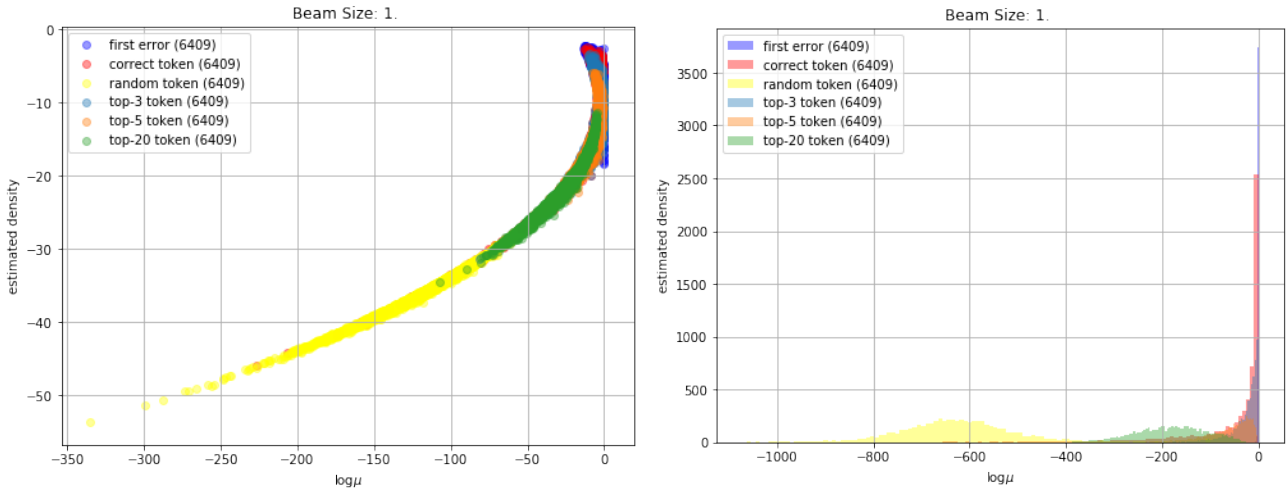
Here and further, let us denote: $\sigma^2 := inens\_var$

$$g_{naive} := \log p(y|x, \mathcal{D}) - \log(\sigma^2) \tag{10}$$

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 that we want to get (Fig. 15a). 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 variance that we discussed earlier. As a result, the model simply produces completely random translations.

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

$$g_{less\_naive} := \log p(y|x, \mathcal{D}) - \log(\sigma) \tag{11}$$



(a) Calibrated probabilities by variance

(b) Calibrated probabilities histogram

Figure 16 – g_2 statistics for correct token, first error token, random token, top-3 top-5 and top-20 tokens.
$g_2 = \log p(y|x, \mathcal{D}) - (\log(\sigma) + 4)^2$

As we can see from Fig. 15b, this did not bring any success, since we did not even change the relative order of probability distributions.

If we take another look at Fig. 14, 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 by any other distribution.

Then let us take a function:

$$g_2 := \log p(y|x, \mathcal{D}) - (\log(\sigma) + \alpha)^2 \tag{12}$$

Where α is a certain hyperparameter that will depend on which point the maximum will be reached. Let us consider a special example (Fig. 16a).

As we can see, now most probability mass of correct token moved forward to first positions in probability variational series, but also top tokens moved forward too. By the reason of fact, all tokens' distributions are mixed, we can not certainly separate correct token distribution from any other. But we can make correct token more prevailing over any other token on a fixed variance counter level. As we can

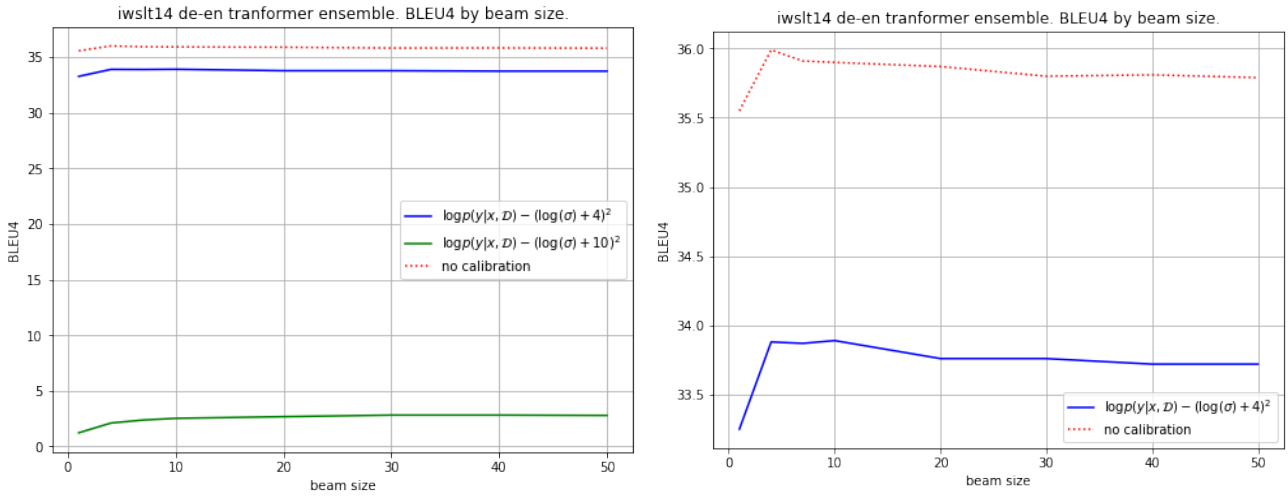
see on a chart, we achieve this goal.

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.

If we take a look at Fig. 16b, we can see ensemble probability distributions after g_2 calibration. Our hypothesis about correct token prevailation confirms.

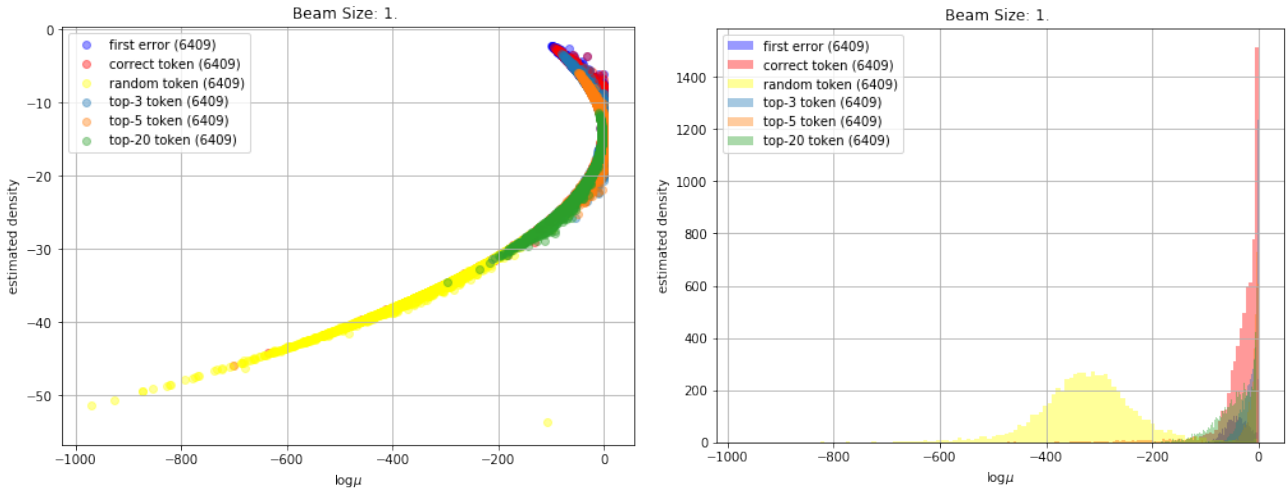
An important note about results is that, while we use beam search with beam size equal 1, we will not earn better results using g_2 calibration, because most part of first error token is still most probable. But if we consider beam search model with beam size more than 1, we can expect quality improvement, because non-greedy beam search not always chooses most likely token and estimate next token probability using cumulative prefix probability.

9.2 Calibration BLEU



(a) (b)

Figure 17 – BLEU4 by beam size for calibrated transformer ensemble



(a) Calibrated probabilities by variance

(b) Calibrated probabilities histogram

Figure 18 – g_2 statistics for correct token, first error token, random token, top-3 top-5 and top-20 tokens.
$g_2 = \log p(y|x, \mathcal{D}) - (\log(\sigma) + 10)^2$

Let us consider BLEU4 evaluation on a test set for calibrated ensemble (Fig. 17).

As we can see, baseline ensemble without any calibration overperforms all our tested calibrated models. The worst results are presented by square calibration with bias equal 10. BLEU is less than 5 for this case. The reason of this poor results can be seen at Fig. 18. After calibration all top-20 (including all more probable tokens) mix together that is why in lots of cases softmaxes have more entropy (recall it meens that our model more uncertain in predictions), so we get a lot of more random predicted tokens. The problem is we choose bias too large and calibration scatter peak is not on correct token distribution but on irrelevant tokens (top-5 top-20 tokens). Therefore, we get too low BLEU.

More significant results are presented by calibration with bias equal 4. The problem in this case is similar to previous one. Now peak is placed on more relevant tokens, but still irrelevant (top-3) tokens enter in resulting predictions. Therefore, let us conduct same experiment but with less bias.

Overall, we consider different kinds of calibration, but we did not achieve quality improvement. The reason of poor final results lie on a weak uncertainty estimate. If we look at Fig. 13, we can see that inensemble variance distribution

for correct token overlap all tokens distributions (except random one) that is why, it is too hard to separate correct token from others. Too small part of correct token distribution for variance is separable from others. This part is too small to get enough quality improvement.

10 Conclusion

In this work we conduct misclassification detection experiments, using different types of uncertainty estimates, on neural machine translation task. We explore correlation between uncertainty and predictions error, analysing uncertainty estimates distributions. Our conclusion is that it is possible to detect prediciton error via uncertainty estimation, because considered uncertainty estimates correlate with errors (correct tokens have lower expected values than others). At the same time, it is hard to evaluate correct quality estimation for misclassification detection via uncertainty, because there are lots of problems with defining reasonable tokens correctness measure.

Moreover, we go deeper softmax level and explore new way of uncertainty estimation. We introduce inensemble uncertainty estimate for every token. We approve correlation between new measure and predictions error. As well, we try to recalibrate token probabilities using token-level inensemble varinace.

Unfortunately, we do not achive model quality improvement, but also we do not claim that this hypothesis is wrong. Further studies should conduct same experiments using different uncertainty estimates. As we conclude inensemble variance distribution has weak correlation with error predictions to achive high quality improvement.

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.