Towards a Better Understanding of Label Smoothing in Neural Machine Translation

Yingbo Gao **Weiyue Wang** **Christian Herold** **Zijian Yang** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
{gao|wwang|herold|zyang|ney}@i6.informatik.rwth-aachen.de

Abstract

In order to combat overfitting and in pursuit of better generalization, label smoothing is widely applied in modern neural machine translation systems. The core idea is to penalize over-confident outputs and regularize the model so that its outputs do not diverge too much from some prior distribution. While training perplexity generally gets worse, label smoothing is found to consistently improve test performance. In this work, we aim to better understand label smoothing in the context of neural machine translation. Theoretically, we derive and explain exactly what label smoothing is optimizing for. Practically, we conduct extensive experiments by varying which tokens to smooth, tuning the probability mass to be deducted from the true targets and considering different prior distributions. We show that label smoothing is theoretically well-motivated, and by carefully choosing hyperparameters, the practical performance of strong neural machine translation systems can be further improved.

1 Introduction

In recent years, Neural Network (NN) models bring steady and concrete improvements on the task of Machine Translation (MT). From the introduction of sequence-to-sequence models (Cho et al., 2014; Sutskever et al., 2014a), to the invention of the attention mechanism (Bahdanau et al., 2015; Luong et al., 2015), end-to-end sequence learning with attention becomes the dominant design choice for Neural Machine Translation (NMT) models. From the study of convolutional sequence to sequence learning (Gehring et al., 2017a,b), to the prosperity of self-attention networks (Vaswani et al., 2017; Devlin et al., 2019), modern NMT systems, especially Transformer-based ones (Vaswani et al., 2017), often deliver state-of-the-art performances (Bojar et al., 2018; Barrault et al., 2019), even under the condition of large-scale corpora (Ott et al., 2018; Edunov et al., 2018).

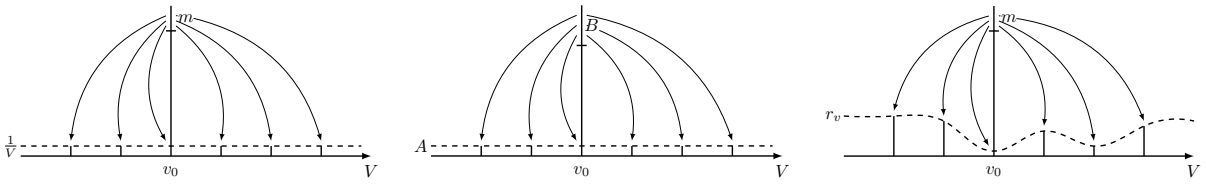
In Transformer-based models, label smoothing is a widely applied method to improve model performance. Szegedy et al. (2016) initially introduce the method when making refinements to the Inception (Szegedy et al., 2015) model, with the motivation to combat overfitting and improve adaptability. In principle, label smoothing discounts a certain probability mass from the true label and redistributes it uniformly across all the class labels. This lowers the difference between the largest probability output and the others, effectively discouraging the model to generate overly confident predictions. Since information entropy (Shannon, 1948) can be thought of as a confidence measure of a probability distribution, Pereyra et al. (2017) add a negative entropy regularization term to the conventional cross entropy training criterion and compare it with uniform smoothing and unigram smoothing. Müller et al. (2019) deliver further insightful discussions about label smoothing, empirically investigating it in terms of model calibration, knowledge distillation and representation learning.

Label smoothing itself is an interesting topic that brings insights about the general learnability of a neural model. While existing methods are rather heuristical in their nature, the fact that simply discounting some probability mass from the true label and redistributing it with some prior distribution (see Figure 1 for an illustration) works in practice, is worth to be better understood.

In this paper, we raise two high-level research questions to outline our work:

1. Theoretically, what is label smoothing (or the related confidence penalty) optimizing for?
2. Practically, what is a good recipe in order to apply label smoothing successfully in NMT?

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 212–223
December 4 - 7, 2020.



(a) uniform distribution (b) confidence penalty (c) arbitrary distribution

Figure 1: An illustration of label smoothing with various prior distributions. m and B are discounted probabiltiy masses. V is the vocabulary size and v_0 is the correct target word. $\frac{1}{V}$, A and r_v are prior distributions. Smoothing with (a), m is equally redistributed across the vocabulary. Smoothing with (b), A is implicitly $\frac{1}{V}$ everywhere as well, and the exact value of B can be obtained (Section 3.2). Smoothing with (c), m goes to each class in proportion to an arbitrary smoothing prior r_v (Section 4.3).

The presentation of our results is organized into three major sections:

- First, we introduce a generalized formula for label smoothing and derive the theoretical solution to the training problem.
- Second, we investigate various aspects that affect the training process and show an empirically good recipe to apply label smoothing.
- Finally, we examine the implications in search and scoring and motivate further research into the mismatch between training and testing.

2 Related Work

The extensive use of NNs in MT (Bojar et al., 2016, 2017, 2018; Barrault et al., 2019) is a result of many pioneering and inspiring works. Continuous-valued word vectors lay the foundation of modern Natural Language Processing (NLP) NNs, capturing semantic and syntactic relations and providing numerical ways to calculate meaningful distances among words (Bengio et al., 2001; Schwenk et al., 2006; Schwenk, 2007; Sundermeyer et al., 2012; Mikolov et al., 2013a,b). The investigations of sequence-to-sequence learning (Cho et al., 2014; Sutskever et al., 2014b), the studies of attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) and the explorations into convolutional and self-attention NNs (Gehring et al., 2017a,b; Vaswani et al., 2017) mark steady and important steps in the field of NMT. Since the introduction of BERT (Devlin et al., 2019), the Transformer model (Vaswani et al., 2017) becomes the de facto architectural choice for many competitive NLP systems. Among the numerous ingredients that make Transformer networks successful, label smoothing is one that must not be overlooked and shall be the focus of this work.

The idea of smoothing is not new in itself. For instance, many smoothing heuristics and functions are investigated in the context of count-based language modeling (Jelinek and Mercer, 1980; Katz, 1987; Church and Gale, 1991; Kneser and Ney, 1995; Chen and Goodman, 1996). Interestingly, when training NNs, the idea of smoothing comes in a new form and is applied on the empirical one-hot target distributions.

Proposed to counteract overfitting and pursue better generalization, label smoothing (Szegedy et al., 2016) finds its first applications in NNs in the field of computer vision. Later, the method is shown to be effective in MT (Vaswani et al., 2017). Furthermore, it is also helpful when applied in other scenarios, e.g. Generative Adversarial Networks (GANs) (Salimans et al., 2016), automatic speech recognition (Chiu et al., 2018), and person re-identification (Ainam et al., 2019). Since the method centralizes on the idea of avoiding over-confident model outputs on training data, it is reanalyzed in Pereyra et al. (2017). The authors include an additional confidence penalty regularization term in the training loss, and compare it to standard label smoothing with uniform or unigram prior. While label smoothing boosts performance significantly compared to using hard target labels, the difference in performance gains when comparing different smoothing methods is relatively small. Müller et al. (2019) bring recent advancements towards better intuitive understandings of label smoothing. They observe a clustering effect of learned features and argue that label smoothing improves model calibration, yet hurting knowledge distillation when the model is used as a teacher for another student network.

As a regularization technique in training, label smoothing can be compared against other methods such as dropout (Srivastava et al., 2014) and Dis-

turbLabel (Xie et al., 2016). Intuitively, dropout can be viewed as ensembling different model architectures on the same data and DisturbLabel can be viewed as ensembling the same model architecture on different data, as pointed out in Xie et al. (2016). Interestingly, label smoothing can also be understood as estimating the marginalized label dropout during training (Pereyra et al., 2017). In this paper, we propose two straightforward extensions to label smoothing, examining token selection and prior distribution. Salimans et al. (2016) and Zhou et al. (2017) investigate a similar issue to the former. In the context of GANs, they select only those positive examples to smooth while we consider the task of MT, discussing how many tokens to smooth and how they should be selected. Pereyra et al. (2017) and Gao et al. (2019) talk about ideas similar to the latter. In their respective contexts, one experiments with unigram probabilities for label smoothing and the other uses Language Model (LM) posteriors to softly augment the source and target side of MT training data.

3 Solving the Training Problem

The standard label smoothing (STN) loss, as used by Vaswani et al. (2017), can be expressed as:

$$L^{\text{STN}} = -\sum_{n=1}^{N}\sum_{v=1}^{V}\left((1-m)p_v + m\frac{1}{V}\right)\log q_v \tag{1}$$

where L^{STN} denotes the cross entropy with standard label smoothing, n is a running index in the total number of training tokens N, v is a running index in the target vocabulary V, m is the hyperparameter that controls the amount of probability mass to discount, p_v is the one-hot true target distribution and q_v is the output distribution of the model.

The confidence penalty (CFD) loss, as used by Pereyra et al. (2017), can be expressed as:

$$L^{\text{CFD}} = -\sum_{n=1}^{N}\sum_{v=1}^{V}\left(p_v - m'q_v\right)\log q_v \tag{2}$$

where L^{CFD} denotes the confidence-penalized cross entropy, m' in this case is the hyperparameter that controls the strength of the confidence penalty and thus differs from m in Equation 1.

In both cases, the outer summation is over all of the training tokens N, implicating that all of the target token probabilities are smoothed. The dependencies of q_v and p_v on n are omitted for simplicity.

Additionally for Equation 1, authors of both papers (Vaswani et al., 2017; Pereyra et al., 2017) point out that the uniform prior can be replaced with alternative distributions over the target vocabulary. One more thing to notice is the negative sign in front of the non-negative term m' in Equation 2, which means that $p_v - m'q_v$ is not a probability distribution anymore. One can nonetheless apply tricks to normalize the term inside the parentheses so that it becomes a probability distribution, e.g.:

$$L^{\text{CFD}}_{\text{normalized}_1} = -\sum_{n=1}^{N}\sum_{v=1}^{V}\log q_v \quad \cdot \frac{(p_v - m'q_v) - \min(p_v - m'q_v)}{\sum_{v'=1}^{V}(p_{v'} - m'q_{v'}) - \min(p_{v'} - m'q_{v'})} \tag{3}$$

or

$$L^{\text{CFD}}_{\text{normalized}_2} = -\sum_{n=1}^{N}\sum_{v=1}^{V}\log q_v \quad \cdot \frac{\exp(p_v - m'q_v)}{\sum_{v'=1}^{V}\exp(p_{v'} - m'q_{v'})} \tag{4}$$

and implement it as an additional layer of activation during training, where v' is an alternative running index in the vocabulary. In any case, the integration of Equation 2 into the form of Equation 1 cannot be done without significantly modifying the original confidence penalty, and we leave it for future work.

3.1 Generalized Formula

In an effort to obtain a unified view, we propose a simple generalized formula and make two major changes. First, we separate the outer summation over the tokens and divide it into two summations, namely "not to smooth" and "to smooth". Second, we modify the prior distribution to allow it to depend on the position, current token and model output. In this case, r could be the posterior from some helper model (e.g. an LM), and during training, obtaining it on-the-fly is not expensive, as previously shown (Bi et al., 2019; Wang et al., 2019). The generalized label smoothing (GNR) loss can be expressed as:

$$L^{\text{GNR}} = -\sum_{n\in\mathcal{A}}\sum_{v=1}^{V}p_v\log q_v \quad - \sum_{n\in\mathcal{B}}\sum_{v=1}^{V}\left((1-m)p_v + mr_{v,q_v}\right)\log q_v \tag{5}$$

where L^{GNR} denotes the generalized cross entropy, $\mathcal{A}$ is the set of tokens not to smooth, $\mathcal{B}$ is the set of tokens to smooth, r_{v,q_v} is an arbitrary prior distribution for smoothing and again we drop the dependencies of p_v, q_v and r_{v,q_v} on n for simplicity.

A natural question when explicitly writing out $\mathcal{A}$ and $\mathcal{B}$, s.t. $\mathcal{A} \cap \mathcal{B} = \varnothing$ and $|\mathcal{A} \cup \mathcal{B}| = N$, is which tokens to include in $\mathcal{B}$. Here, we consider two simple ideas: uniform random sampling (RND) and an entropy-based uncertainty heuristic (ENT). The former chooses a certain percentage of tokens to smooth by sampling tokens uniformly at random. The latter prioritizes those tokens whose prior distributions have higher entropy. The logic behind the ENT formulation is that when the prior distribution is flattened out, yielding a higher entropy, the helper model is uncertain about the current position, and the model output should thus be smoothed. Formally, the two heuristics can be expressed as:

$$\mathcal{B}^{\text{RND}} = \{n; \rho_n \sim U(0,1), \rho_n \leq \pi\} \quad (6)$$

$$\mathcal{B}^{\text{ENT}} = \{b_1, b_2, ..., b_{\lceil \pi N \rceil}\} \quad (7)$$

where ρ_n is a sample from the uniform distribution U in $[0,1]$, π is a hyperparameter controlling the percentage of tokens to smooth and $\{b_1, b_2, ..., b_N\}$ is a permutation of data indices $\{1, 2, ...N\}$ in descending order of the entropy of prior r, i.e. $\forall 1 \leq i \leq j \leq N, -\sum_V r_{b_i} \log r_{b_i} \geq -\sum_V r_{b_j} \log r_{b_j}$.

The hyperparameter m in Equation 5 deserves some further notice. This is essentially the parameter that controls the strength of the label smoothing procedure. When it is zero, no smoothing is done. When it is one and $|\mathcal{B}| = N$, the model is optimized to output the prior distribution r. One can obviously further generalize it so that m depends also on n, v and q_v. However in this work, we focus on the outer summation in N and alternative priors r, and leave the exploration of adaptive smoothing strength m_{n,r,q_v} for future work.

3.2 Theoretical Solution

When it comes to the analysis of label smoothing, previous works focus primarily on intuitive understandings. Pereyra et al. (2017) observe that both label smoothing and confidence penalty lead to smaller gradient norms during training. Müller et al. (2019) argue that label smoothing helps beam-search by improving model calibration. They further visualize the learned features and show a clustering effect of features from the same class. In this work, we concentrate on finding a theoretical solution to the training problem, and show exactly what label smoothing and confidence penalty are optimizing for.

Consider the optimization problem when training with Equation 1:

$$\min_{q_1,q_2,...,q_V} L_n^{\text{STN}}, \quad \text{s.t.} \sum_{v=1}^{V} q_v = 1 \quad (8)$$

While in practice we use gradient optimizers to obtain a good set of parameters of the NN, the optimization problem actually has well-defined analytical solutions locally:

$$\tilde{q}_v^{\text{STN}} = (1-m)p_v + m\frac{1}{V} \quad (9)$$

which is simply a linear interpolation between the one-hot target distribution p_v and the smoothing prior $\frac{1}{V}$, with $m \in [0,1]$ being the interpolation weight. One can use either the divergence inequality or the Lagrange multiplier method to obtain this result (see Appendix A).

Consider the optimization problem when training with Equation 2:

$$\min_{q_1,q_2,...,q_V} L_n^{\text{CFD}}, \quad \text{s.t.} \sum_{v=1}^{V} q_v = 1 \quad (10)$$

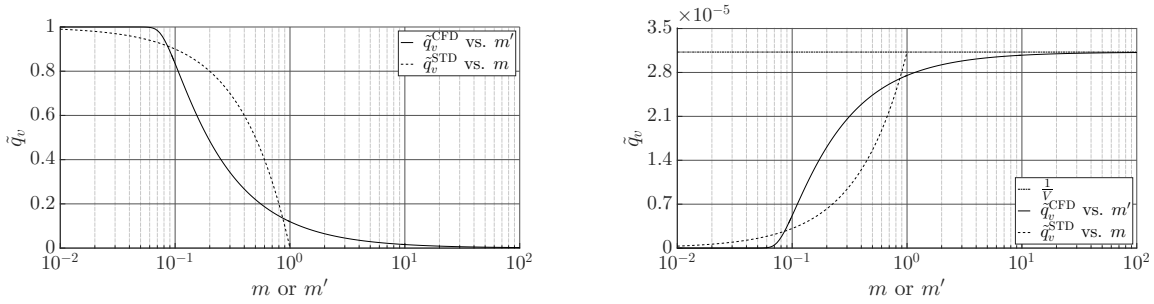
The problem becomes harder because now the regularization term also depends on q_v. Introducing the Lagrange multiplier λ and solving for optima will result in a transcendental equation. Making use of the Lambert W function (Corless et al., 1996), the solution can be expressed as (see Appendix A for detailed derivation):

$$\tilde{q}_v^{\text{CFD}} = \frac{p_v}{m' W_0\left(\frac{p_v}{m'} e^{1+\frac{\lambda}{m'}}\right)} \quad (11)$$

where W_0 is the principal branch of the Lambert W function and λ is the Lagrange multiplier, which is numerically solvable[1] when non-negative m' and probability distribution p_v are given. Equation 11 essentially gives a non-linear relationship between $\tilde{q}_v^{\text{CFD}}$ and p_v, controlled by the hyperparameter m'.

Now that theoretical solutions are presented in Equation 9 and 11, it is possible to plot the graphs of optimal $\tilde{q}_v$, with respect to m and m'. Shown in Figure 2, as expected for both STN and CFD, the overall effect is to decrease q_v when $p_v = 1$ and increase q_v when $p_v = 0$. When m or m' gets large

[1] One can use $\lim_{m' \to 0} \tilde{q}_v^{\text{CFD}}$ to avoid division by zero.



(a) $p_v = 1$ (b) $p_v = 0$

Figure 2: Graphs of optimal $\tilde{q}_v$ w.r.t. m or m'. Note the logarithmic scale in horizontal axes, with $m \in [0, 1]$ and $m' \geq 0$. In order to obtain numerical solutions for $\tilde{q}_v^{\mathrm{STN}}$ and $\tilde{q}_v^{\mathrm{CFD}}$, we set $V = 32000$, which is a common vocabulary size when operating on sub-word levels.

enough, the total probability mass is discounted and $\frac{1}{V}$ is redistributed to each token in the vocabulary. The graph of GRN[2] is similar to STD, only changing the limit from $\frac{1}{V}$ to r_v as m approaches one, and not included here for brevity. One last thing to notice is that the outer summation over the tokens is ignored. If it is taken into consideration, $\tilde{q}$ is dragged towards the empirical distribution given by the corpus[3].

4 Finding a Good Recipe

In this section, we describe our results and insights towards a good recipe to successfully apply label smoothing. We experiment with six IWSLT2014 datasets: German (`de`), Spanish (`es`), Italian (`it`), Dutch (`nl`), Romanian (`ro`), Russian (`ru`) to English (`en`), and one WMT2014 dataset: English to German. The statistics of these datasets are summarized in Table 1. To prepare the subword tokens, we adopt joint byte pair encoding (Sennrich et al., 2016), and use 10K and 32K merge operations on IWSLT and WMT, respectively. When preprocessing IWSLT, we remove sentences longer than 175 words, lowercase both source and target sides, randomly subsample roughly 4.35% of the training sentence pairs as development data and concatenate all previously available development and test sets as test data, similar to Gehring et al. (2017a). As for the preprocessing of WMT, we follow the setup in Ott et al. (2018). Using the Transformer architecture (Vaswani et al., 2017), we apply the base setup for IWSLT and the big setup for WMT. For all language pairs, we share all three embedding matrices. All helper models are also Transformer-based. We conduct all experiments using fairseq (Ott et al., 2019), monitor development set perplexity during training, and report BLEU (Papineni et al., 2002) scores on test sets after beam search.

4.1 Token Selection

The first thing to determine is how to select tokens for smoothing and how many tokens to smooth.

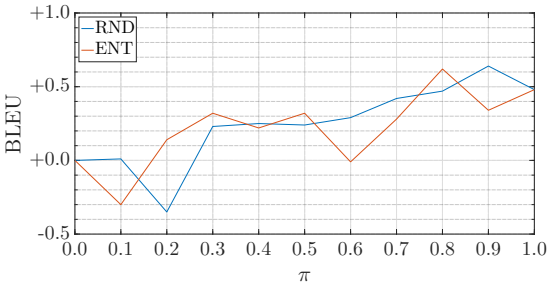


Figure 3: Smoothing with RND versus ENT on `de-en`. m is set to 0.1. The development and test perplexities of the helper LM are 53.8 and 46.5.

For this purpose, we begin by considering models smoothed with an LM helper. The helper LM is trained on target sentences from the corresponding parallel data till convergence. Figure 3 shows a comparison between RND and ENT, varying the percentage of smoothed tokens π and using the absolute performance improvements in BLEU as the vertical axis. Since the two methods only affect the order in which tokens are selected, they should yield the exact same results when all tokens are selected. This can be clearly seen from the figure and serves as a sanity check for the correctness of

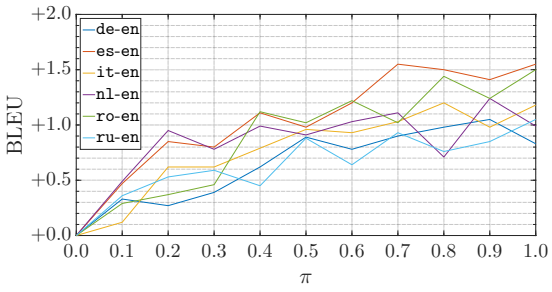
[2] Assuming r only depends on n, v and not q_v. In the latter case, one needs to solve the optimization problem ignoring the outer summation and reusing the Lagrange multiplier.

[3] For an intuitive understanding, consider the case when two sentence pairs have the exact same context up to a certain target position but the next tokens are different (e.g. "Danke ." in German being translated to "Thank you ." and "Thank you very much ." in English, the period in the first translation and "very" in the second translation have the same context.)

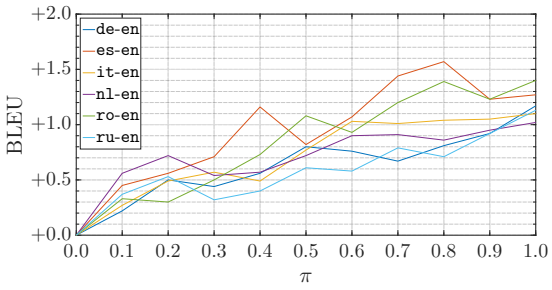
dataset		IWSLT						WMT
language pair		de-en	es-en	it-en	nl-en	ro-en	ru-en	en-de
number of sentence pairs	train	160K	169K	167K	154K	168K	153K	4.50M
	valid	7.3K	7.7K	7.6K	7.0K	7.6K	7.0K	3.0K
	test	6.8K	5.6K	6.6K	5.4K	5.6K	5.5K	3.0K

Table 1: Data statistics of IWSLT and WMT datasets.

the implementation. The RND and ENT curves follow a similar trend, increasing with the number of smoothed tokens. From the curves, neither selection method is consistently better than the other, indicating that the entropy-based selection heuristics is probably an oversimplification considering the stochasticity introduced when altering the number of smoothed tokens. We continue to examine the uphill trend seen in Figure 3 in other cases.



(a) uniform as r_v, $m = 0.1$



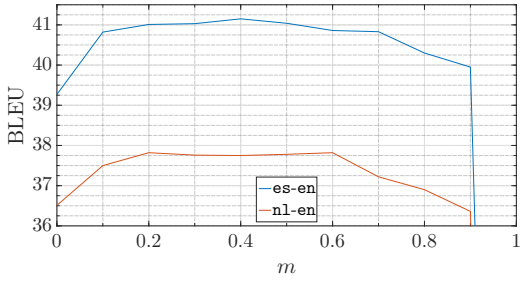
(b) unigram as r_v, $m = 0.1$

Figure 4: Smoothing different percentages of tokens.

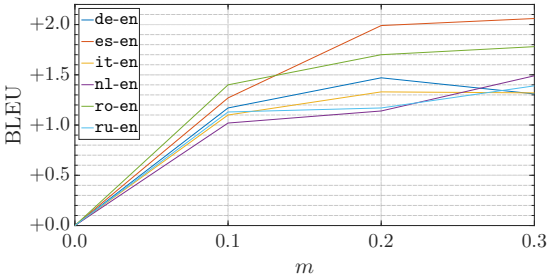
Figure 4 reveals the relationship between absolute BLEU improvements and π, when smoothing with uniform or unigram (RND) distributions. While for each language pair the actual changes in BLEU differ, it is clear to conclude that, the more tokens smoothed, the better the performance. This conclusion is rather universal and holds true for the majority of our experiment settings (varying m and r). From here on, we smooth all tokens, i.e. $|\mathcal{B}| = N$, by default.

4.2 Probability Mass

Our next goal is to find good values of m.



(a) uniform as r_v



(b) unigram as r_v

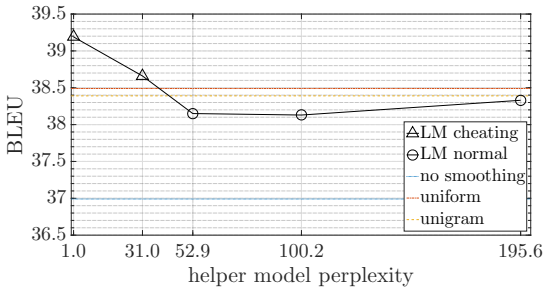
Figure 5: Discounting different probability masses.

The discounted probability mass m is a tunable hyperparameter that is set to 0.1 in the original Transformer (Vaswani et al., 2017) paper. We vary this parameter in the case of uniform smoothing and unigram smoothing, and plot the results in Figure 5. As shown in Figure 5a, the BLEU score immediately improves at $m = 0.1$, then plateaus when $m \in [0.3, 0.6]$, slowly decreases when $m \in [0.7, 0.9]$ and quickly drops to zero when m approaches one. When $m = 1$, the model is optimized towards a uniform distribution and completely ignores the training data. Because perplexity can be thought of as the effective vocabulary size of a model, we examine the perplexities when $m = 1$ for both language pairs. As expected, the development perplexities are around 10K, which is in the same order of magnitude as the corresponding vocabulary sizes. Another interesting observation is that the BLEU scores only drop when

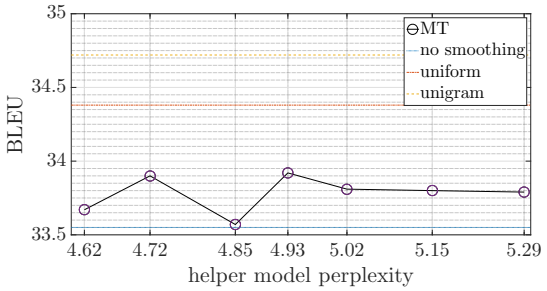
m gets close to one and the model produces acceptable translations elsewhere. This indicates that NN models trained with gradient optimizers are very good at picking out the effective training signals even when they are buried in much stronger noise signals (the uniform smoothing priors in the case of Figure 5a). This could be further related to multi-task learning (Ruder, 2017), where the system performances are also related to the regularization weights of the auxiliary losses. For unigram, we vary m in $\{0.1, 0.2, 0.3\}$. As seen in Figure 5b, while smoothing with $m = 0.1$ gives a large improvement over no smoothing, setting $m = 0.3$ further boosts the performance, consistently for all six IWSLT language pairs.

4.3 Prior Distribution

Furthermore, we explore the use of LM and MT posteriors as prior distributions for smoothing.



(a) `ro-en`, LM posterior as r_v, $m = 0.1$



(b) `de-en`, MT posterior as r_v, $m = 0.1$

Figure 6: Smoothing with LM and MT posteriors.

We train systems using Transformer LMs and MT models of different qualities for label smoothing, as in Figure 6. To obtain very good LMs, we train them with test data and mark the cheating LMs in Figure 6a. We additionally plot the BLEU scores of models with no smoothing, smoothed with uniform and unigram, as horizontal lines to compare the absolute performances. Intuitively, the curve should follow a downhill trend, meaning that the worse the helper model performs, the worse the model smoothed with it performs. This is loosely the case for LM, with cheating LMs giving better performances than uniform and unigram, and normal LMs lacking behind. As for MT, improvement over the no smoothing case is seen in Figure 6b. However, neither the downhill trend nor the competence over other priors in terms of BLEU, is seen. This suggests that the model is probably not utilizing the information in the soft distribution effectively. Related to knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016), a trainable teacher (the helper model in our case) might be further beneficial (Bi et al., 2019; Wang et al., 2018).

One important thing to mention is that, while neither LM nor MT outperforms uniform or unigram in terms of test BLEU score in our experiments, we see significant drops in development set perplexities when smoothing with LM or MT. This signals a mismatch between training and testing, and suggests that smoothing with LM or MT indeed works well for the optimization criterion, but not as much for the final metric, the calculation of which involves beam search and scoring of the discrete tokens.

4.4 Final Results

Finally, we report BLEU scores of our best systems across all language pairs in Table 2. While applying uniform label smoothing significantly improves over the baselines, by using a good recipe, an additional improvement of around +0.5 BLEU is obtained across all language pairs. For the hyperparameters, we find that smoothing **all tokens** by $\mathbf{m = 0.3}$ with **a unigram prior** is a good recipe, consistently giving one of the best BLEU scores.

5 Analyzing the Mismatch

As discussed in Section 4.3, models smoothed with LMs or MT model posteriors yield very good development set perplexities but no big improvements in terms of test BLEU scores. Here, we further investigate this phenomenon in terms of search and scoring.

5.1 Search

We first plot the test BLEU scores with respect to the beam size used during search. In Figure 7, we see that the dashed curves for "no smoothing", "uniform" and "unigram" initially increase and then plateau, which is an expected shape (see Figure 8

dataset	IWSLT						WMT
language pair	de-en	es-en	it-en	nl-en	ro-en	ru-en	en-de
no label smoothing	33.6	39.3	31.2	36.5	37.0	22.3	28.0
Vaswani et al. (2017)	34.4	40.8	32.4	37.5	38.5	23.4	28.4
our best recipe	**35.0**	**41.5**	**32.8**	**38.0**	**39.0**	**23.9**	**29.0**

Table 2: BLEU scores can be significantly improved with good label smoothing recipes. The first row of numbers corresponds to using only the cross entropy criterion for training. The second row of numbers corresponds to the Transformer baselines. The last row contains scores obtained with our best hyperparameters.

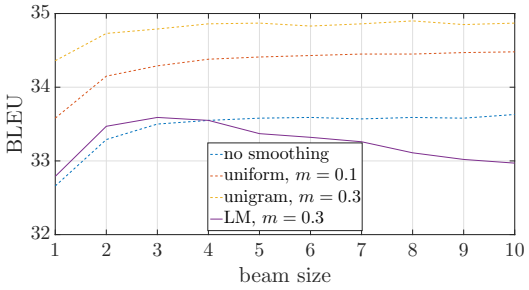
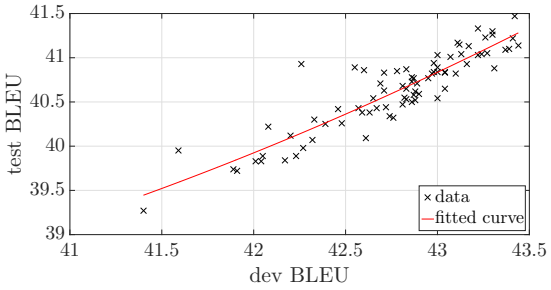
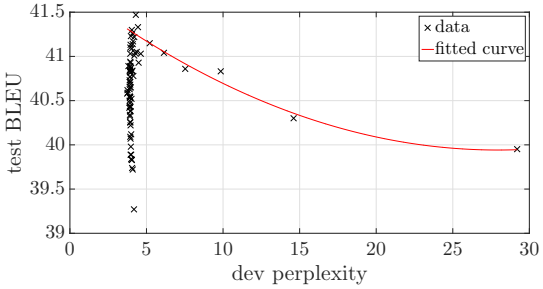


Figure 7: BLEU versus beam size on de-en.



(a) Dev BLEU is a good proxy for test BLEU.



(b) Dev perplexity is a bad proxy for test BLEU.

Figure 8: Relationships between test BLEU and dev metrics. 79 converged es-en models with different label smoothing hyperparameters are scattered.

in Zhou et al. (2019)). However, the solid curve for LM drops quickly as beam size increases (see Stahlberg and Byrne (2019) for more insight). A possible explanation is that models smoothed with LMs generate search spaces that are richer in probability variations and more diversified, compared to e.g. uniform label smoothing. As search becomes stronger, hypotheses that have higher probabilities, but not necessarily closer to the true targets, are found. This suggests that the mismatch in development set perplexity and test BLEU is a complex phenomenon and calls for more analysis.

5.2 Scoring

We further examine test BLEU with respect to development (dev) BLEU and dev perplexity. As shown in Figure 8a, test BLEU is nicely correlated with dev BLEU, indicating that there is no mismatch between dev and test in the dataset itself. However, as in Figure 8b, although test BLEU increases with a decreasing dev perplexity, in regions of low dev perplexities, there exist many systems with very different test performances ranging from 39.3 BLEU to 41.5 BLEU. Despite perplexity being directly related to the cross entropy training criterion, this is an example where it fails to be a good proxy for the final BLEU metric. Against this mismatch between training and testing, either a more BLEU-related dev score or a more perplexity-related test metric needs to be considered.

6 Conclusion

In this work, we investigate label smoothing in neural machine translation. Considering important aspects in label smoothing: token selection, probability mass and prior distribution, we introduce a generalized formula and derive theoretical solutions to the training problem. Examining the effect of various hyperparameter choices, practically we show that with a good label smoothing recipe, one can obtain consistent improvements over strong baselines. Delving into search and scoring, we finally emphasize the mismatch between training and testing, and motivate future research. Reassuring that label smoothing brings concrete improvements and considering that it only operates at the output side of the model, our next step is to explore similar smoothing ideas at the input side.

Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG.

References

J. Ainam, K. Qin, G. Liu, and G. Luo. 2019. Sparse label smoothing regularization for person re-identification. *IEEE Access*, 7:27899–27910.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 932–938. MIT Press.

tianchi Bi, hao xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Multi-agent learning for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 856–865, Hong Kong, China. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Kenneth W. Church and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54.

Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. 1996. On the lambertw function. *Advances in Computational mathematics*, 5(1):329–359.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017a. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Fred Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 35:400–401.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, Detroit, Michigan, USA.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When does label smoothing help? *CoRR*, abs/1906.02629.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *ArXiv*, abs/1706.05098.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. 2016. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc.

Holger Schwenk. 2007. Continuous space language models. *Comput. Speech Lang.*, 21(3):492–518.

Holger Schwenk, Daniel Dechelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730, Sydney, Australia. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China. Association for Computational Linguistics.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, Portland, OR, USA.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014a. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014b. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Jinzhuo Wang, Wenmin Wang, and Wen Gao. 2018. Beyond knowledge distillation: Collaborative learning for bidirectional model assistance. *IEEE Access*, 6:39490–39500.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019. Improving back-translation with uncertainty-based confidence estimation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 791–802, Hong Kong, China. Association for Computational Linguistics.

Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. 2016. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*, pages 4753–4762.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.

Zhiming Zhou, Han Cai, Shu Rong, Yuxuan Song, Kan Ren, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Activation maximization generative adversarial nets. In *ICLR*.

A Derivation of Optimal Solutions

Ignoring the outer summation in tokens and dropping the dependencies on n for simplicity, the optimization problem in Equation 8 can be solved analytically and the optimization problem in Equation 10 can be solved numerically.

A.1 Minimizing L_n^{STD}

L_n^{STD} takes the form of $\sum_x p \log q$, where both p and q are probability distributions in x. The divergence inequality can be directly applied:

$$
\begin{aligned}
L_n^{\text{STD}} &= \sum_{v=1}^{V} -\left((1-m)p_v + m\frac{1}{V}\right)\log q_v \\
&\geq \sum_{v=1}^{V} -\left((1-m)p_v + m\frac{1}{V}\right) \quad \cdot \\
&\qquad \log\left((1-m)p_v + m\frac{1}{V}\right)
\end{aligned}
$$

Alternatively, one can use the Lagrange multiplier and calculate first order derivatives:

$$\begin{aligned}\mathcal{L}_n^{\mathrm{STD}}(q_v, \lambda) &= L_n^{\mathrm{STD}} + \lambda\left(\sum_v q_v - 1\right)\\ \frac{\partial \mathcal{L}_n^{\mathrm{STD}}}{\partial q_v} &= \frac{(1-m)p_v + m\frac{1}{V}}{q_v} + \lambda\\ \frac{\partial \mathcal{L}_n^{\mathrm{STD}}}{\partial \lambda} &= \sum_v q_v - 1\end{aligned}$$

Afterwards, set them to zero and solve for λ:

$$\begin{aligned}\frac{\partial \mathcal{L}_n^{\mathrm{STD}}}{\partial q_v} = 0 \Rightarrow q_v &= \frac{(1-m)p_v + m\frac{1}{V}}{-\lambda}\\ \frac{\partial \mathcal{L}_n^{\mathrm{STD}}}{\partial \lambda} = 0 \Rightarrow \lambda &= -1\end{aligned}$$

Plugging λ back in yield q_v, which should be further checked to see if it is a maxima or minima.

In both methods, the minimum is obtained when:

$$\tilde{q}_v^{\mathrm{STD}} = (1-m)p_v + m\frac{1}{V}$$

A.2 Minimizing L_n^{CFD}

Applying the Lagrange multiplier, the first order derivatives can be derived:

$$\begin{aligned}\mathcal{L}_n^{\mathrm{CFD}}(q_v, \lambda) &= L_n^{\mathrm{CFD}} + \lambda\left(\sum_v q_v - 1\right)\\ \frac{\partial \mathcal{L}_n^{\mathrm{CFD}}}{\partial q_v} &= \frac{-(p_v - m'q_v)}{q_v} + m'\log q_v + \lambda\\ \frac{\partial \mathcal{L}_n^{\mathrm{CFD}}}{\partial \lambda} &= \sum_v q_v - 1\end{aligned}$$

Note that setting $\frac{\partial \mathcal{L}_n^{\mathrm{CFD}}}{\partial q_v}$ to zero results in a transcendental equation in the form of:

$$Ax + bx\log x = C$$

where $A = (m' + \lambda)$, $B = m'$, $C = p_v$ and $x = q_v$.

Consider that the Lambert W function is the inverse function of:

$$f(W) = We^W$$

we can rewrite the transcendental equation until we reach a similar form:

$$\begin{aligned}Ax + Bx\log x &= C\\ \xrightarrow{t=\frac{1}{x}} \frac{A}{t} - \frac{B\log t}{t} &= C\\ A &= Ct + B\log t\\ \frac{A}{B} &= \frac{C}{B}t + \log t\\ \xrightarrow{u=\frac{C}{B}t} \frac{A}{B} &= u + \log u + \log\frac{B}{C}\\ u + \log u &= \frac{A}{B} + \log\frac{C}{B}\\ ue^u &= \frac{C}{B}e^{\frac{A}{B}}\\ \Rightarrow u &= W\left(\frac{C}{B}e^{\frac{A}{B}}\right)\end{aligned}$$

reversing the variable replacements:

$$\begin{aligned}u = \frac{C}{B}t &= \frac{C}{B}\cdot\frac{1}{x}\\ \Rightarrow x &= \frac{C}{B}\cdot\frac{1}{u}\\ &= \frac{C}{BW\left(\frac{C}{B}e^{\frac{A}{B}}\right)}\end{aligned}$$

Finally, plugging in A, B and C, we arrive at Equation 11:

$$\tilde{q}_v^{\mathrm{CFD}} = \frac{p_v}{m'W_0\left(\frac{p_v}{m'}e^{1+\frac{\lambda}{m'}}\right)}$$

When p is a one hot distribution and m' is given, one can use the constraint of q_v being a probability distribution to numerically solve for λ. Once λ is obtained, actual values of $\tilde{q}_v^{\mathrm{CFD}}$ can be calculated.