PRELIMINARY PREPRINT VERSION: DO NOT CITE
The AAAI Digital Library will contain the published
version some time after the conference.

Language Modelling via Learning to Rank

Arvid Frydenlund [1,2], **Gagandeep Singh** [3], **Frank Rudzicz** [1,2,4]
[1] Department of Computer Science, University of Toronto
[2] Vector Institute for Artificial Intelligence
[3] Nuance Communications Inc.
[4] Unity Health Toronto
arvie@cs.toronto.edu
gagandeep.singh1@nuance.com
frank@cs.toronto.edu

Abstract

We consider language modelling (LM) as a multi-label structured prediction task by re-framing training from solely predicting a single ground-truth word to ranking a set of words which could continue a given context. To avoid annotating top-k ranks, we generate them using pre-trained LMs: GPT-2, BERT, and Born-Again models. This leads to a rank-based form of knowledge distillation (KD). We also develop a method using N-grams to create a non-probabilistic teacher which generates the ranks without the need of a pre-trained LM. We confirm the hypotheses that we can treat LMing as a ranking task and that we can do so without the use of a pre-trained LM. We show that rank-based KD generally improves perplexity (PPL) — often with statistical significance — when compared to Kullback–Leibler-based KD. Surprisingly, given the simplicity of the method, the N-grams act as competitive teachers and achieve similar performance as using either BERT or a Born-Again model as teachers. GPT-2 always acts as the best teacher, though, and using it and a Transformer-XL student on Wiki-02, rank-based KD reduces a cross-entropy baseline from 65.27 to 55.94 and against a KL-based KD of 56.70.

1 Introduction and motivation

More often than not, there are many ways to say the same thing. For example, '*the cat sat on the mat*' is semantically equivalent to '*the cat sat on the rug*', in most contexts. This basic fact about language is ignored when training language models, where we try to maximize the probability of predicting a single ground-truth word for a given context and penalize all other words regardless of any potential semantic equivalence. In particular, a word-level language model (LM) defines a distribution over T discrete tokens, $x_1, \ldots, x_T$, as

$$\begin{aligned} p(x_1, \ldots, x_T \,|\, x_0) &= \prod_{t=1}^{T} p(x_t \,|\, x_{<t}) \\ &= \prod_{t=1}^{T} \frac{e^{w_t^{g_t}}}{\sum_{j=1}^{|V|} e^{w_t^j}} = \prod_{t=1}^{T} \frac{e^{w_t^{g_t}}}{Z_t}, \end{aligned} \tag{1}$$

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

where we use the softmax function to model a multinomial distribution over a set vocabulary V, $w_t \in \mathbb{R}^{|V|}$ are the logits produced by the model at step t, j indexes the score for the j^{th} word-type, and g_t is the ground-truth (GT) index at time t. Such models are trained by minimizing the per-word cross-entropy (CE or $H(y, p)$) against the ground-truth text, $y = y_1, \ldots, y_T \in \mathbb{R}^{|V|}$ which are one-hot representations of $x_1, \ldots, x_T$,

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|V|} y_t^i \log \frac{e^{w_t^i}}{Z_t} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|V|} y_t^i (\log Z_t - w_t^i). \tag{2}$$

During training, we assume that y is a one-hot distribution since we do not have access to the true data distribution. This assumption is obviously incorrect, considering that humans are able to form a valid continuation of most contexts with multiple potential words. Call these potential words the *branching set* of a given context. Instead, we may want to give partial credit for predicting words in the branching set, like '*rug*' in instances when the GT is '*mat*' (and vice versa) based on the semantic similarity between the words and contexts. One method for achieving this is via *label smoothing*.

1.1 Label smoothing and knowledge distillation

Label smoothing (LS) is a method which modifies each y_t to be soft-targets, instead of one-hot, by redistributing probability from the GT label to other labels (Szegedy et al. 2016). It is commonly implemented using a hand-crafted function, such as setting the GT probability to 0.9 and spreading the remaining 0.1 uniformly across the other labels (Pereyra et al. 2017). LS is a simple method which has different motivations depending on the problem it is meant to solve. One problem is noisy labels (Lukasik et al. 2020a). Here one views the GTs as truly one-hot but possibly incorrect due to errors in annotation. A second problem is to regularize over-confident models in situations where we desire that model's probabilities should not spike to a single prediction (Pereyra et al. 2017; Müller, Kornblith, and Hinton 2019). Here, one again views the GTs as truly one-hot but they do not want the model to make predictions too close to the one-hot distribution due to how

the model is to be used. However, in our case, we use label smoothing to make the single-target GTs multi-target, under the view that the true GTs are actually multi-target but that we lack access to this information.

Given that the multiple-targets should change given the context, we desire the smoothing values to depend on the context. One can have a data-dependent or semantically-aware LS by using an auxiliary pre-trained model to form label scores, such as using cosine scores from FastText (Elbayad, Besacier, and Verbeek 2018; Li et al. 2019). This can be seen as a form of knowledge distillation (KD), which is when a pre-trained teacher model is used to train a student model by minimizing the cross entropy, $H(p_{\text{KD}}, p)$, between the teacher's distribution, p_{KD} and the student's, p (Hinton, Vinyals, and Dean 2015; Wang and Yoon 2021). Here, the smoothing scores are derived from the teacher's distribution and thus can be viewed as parametric form of semantic LS (Yuan et al. 2020; Tang et al. 2020).

One can perform KD either from a weaker language representation model such as FastText or a strong language model such as GPT-2. However, this raises the question – *who teaches the teacher*? That is, in order to train our LM we need to assume the existence of an already established model, which seems to defeat the purpose. Instead, we set our own desideratum that we do not want to use a pre-trained neural LM as a teacher.

We hypothesize that N-gram statistics are a rich source of semantic information which can be exploited to train neural LMs, i.e., introducing global N-gram statistics to our otherwise local training procedure (Neubig and Dyer 2016; Zhao et al. 2017; Yang, Wang, and Downey 2019). The naïve approach would be to pre-train an N-gram LM to use as our teacher; however, this will not work because the N-gram model will be a weak teacher. N-gram LMs perform worse than neural language models (Mikolov 2012; Jozefowicz et al. 2016). Because of this, if we just try to match the teacher's distribution, as is done in KD, we will end up with a higher perplexity model than had we forgone the KD from the N-gram model. To overcome this issue, we need to examine how label smoothing and N-gram LMs work.

1.2 Ranking in lieu of label smoothing

We can break semantic label smoothing into two problems: a) how to choose which words should get partial credit in a given context, and b) how much partial credit they should receive. To solve the first problem, we could use the common 'distributional hypothesis' from linguistics to give partial credit to words which share similar contexts (Harris 1954; Mikolov et al. 2013). That is, words which have some shared context should get some partial credit. The distributional hypothesis underlies N-gram language models as well, which employ smoothing algorithms and back-off methods to weigh higher-order N-grams more than lower order ones (Goodman 2001). To solve the problem of quantifying credit assignment, it seems we need a probabilistic (even if un-normalized) model to specify the smoothing values. However, as reasoned above, if we naïvely use N-gram LMs for this, we will wind up trying to match the distribution of a weaker teacher model. One solution to this would be to modify the N-gram LM teacher itself.

N-gram LM algorithms were developed with the goal of the model being a valid LM for use in applications. One criterion of such a model is that it produces a non-zero probability. This is achieved by including the unigram distribution. Another criterion is that the N-gram LM generalizes well to unseen contexts. This is achieved by well-developed smoothing algorithms. If we only intend to use the N-gram model as a teacher, we may ease or remove these criteria. For example, we might posit the converse of the distributional hypothesis and say that we will restrict credit from words which do not share contexts. To achieve this, we could forego backing-off to the unigram distribution and smoothing into unseen contexts. One of the main motivations of neural LMs over N-grams is their ability to generalize due their distributed representations (Bengio et al. 2003). This implies that we could forego applying any kind of smoothing algorithm to the N-gram teacher and let the neural student learn to generalize to unseen contexts. If we follow this approach, then we would need to make modifications to existing N-gram LM algorithms to determine how to combine the various N-gram probabilities into a (sparse) probabilistic model. Instead, we propose a more radical approach in order to avoid this.

If we decompose the problem of specifying label weights, we are specifying not only how to weight each cost term but also, implicitly, an ordering of the weighted terms. Thus, given label weights, we can convert the problem of LS to a ranking problem by specifying that the most probable label is the most relevant rank and the second most probable is the second most relevant rank, etc. **The main contribution of this paper is therefore the idea that we may not need to specify specific label weights, so long as we can specify an *order* to the labels.** In particular, we hypothesize that the ordering of labels contains sufficient semantic information to be used for our desired partial credit. To specify the ordering, we again employ the distributional hypothesis and consider that increasing the shared context between a word and the ground-truth word increases the relevance of that word in the ordering. Note that the original GT word will always retain the most relevant rank, as it will always share the most context with itself. Since ordinal information is less strict than interval information, we may overcome the issue of having a weak teacher since we are no longer trying to directly match a weaker distribution.

This approach obviates the need for a modified probabilistic N-gram LM since we just need to consider the size of a shared N-gram context to determine how much credit to give to a set of words which can follow. We describe how to construct artificial ranking GTs from N-gram statistics in Section 2.1 and how to use these in training the neural LM in Section 2.2.

Adams	River	is	a	tributary	to	the	Thompson	and	Fraser
Moro	passes	Lumber	the		,	Jim		sub	Sam
Shebelle	Lake	<unk>	located		of	Harry			<unk>
Colorado		to	thought		the				Kansas
Missouri		run	north		in				Platte
Payette		flows	298		valleys				Missouri
<unk>		begins			took				Yellowstone
Back		(			by				Mickey
Red		valley			paymen-				Mississippi
Yellowst-		Provincial			is				Cheyenne
Grand		sockeye							Columbia
Mississ-		area							Ohio
		,							Osage
		and							Kissimmee

Figure 1: Branching sets created for the ground-truth '*Adams River is a tributary to the Thompson and Fraser Rivers ...*' using the ground-truths (O_{gt}) and four orders (O_5, O_4, O_3, O_2) from the Wiki02 training partition. The branching sets for '*tributary*' and '*Thompson*' are only the ground-truths since all other orders have been pruned.

2 Methods

2.1 N-gram branching set construction

Let a *branching set* (BS), b_t, be a set of words which can continue a given context $c_t = (x_1, \ldots, x_{t-1})$ at step t. Let $c_t(n)$ be a slice of the context which only includes the previous $n-1$ words (assuming $t \geq n$). Intuitively, the construction of the BS is simple. We are going to consider a series of past N-gram contexts, starting with the largest context until the smallest, and any word that we find in the training set that continues those contexts will be in the BS. Then, in order to derive artificial rank GTs from the BS, we can give a *weak* ordering to the words in the BS by specifying that words in the BS with a larger context should have a higher rank relevance than those with a smaller context. If, for example, we consider bigrams and trigrams then any word that continues the trigram context will also continue the bigram context but words which only continue the bigram context but not the trigram context will be less relevant. Unigrams are excluded since they use no context.

More formally, we construct a tuple of sets referred to as *orders*, $O = (O_{gt}, O_M, \ldots, O_2)$ which will be merged to form the BS. $O_{gt} = \{x_t\}$, only contains the original GT. The other orders are made up of all words which continue a specific context length, such that we construct each of the orders as $O_m = \{x | x \in V \setminus O_{>m}, C(c_t(m), x) \geq 1\}$, where $m \in \{M, \ldots, 2\}$ and $C(c_t(m), x)$ is the count for how many times x appears in the context $c_t(m)$. Note, we exclude words already included in previous orders. We also use a cut-off, q, so that we only consider sufficiently small orderings $|O_m| < q$, since large BSs will not be informative; e.g., all words which follow '*the*' or '*is a*'. The BS is constructed as an ordered tuple where $b_t = (x \,|\, x \in O_i, O_i \in O)$. The artificial GT ranks are then specified as the weak rank ordering of the BS, which is weak since each O_m may contain multiple unordered words.

We can further resolve the rank within each order by using future context in addition to past context. This assumes that, had we access to longer past contexts, the ranks would have resolved the same way as using future information. Note that we can introduce future information here because the teacher model does not need to form a valid probability distribution since it only defines targets for the student model. Let $c_t(n_p, n_f)$ be the context defined by using $n_p - 1$ words in the past and $n_f - 1$ words in the future. We prioritize past context over future context, since that is what the student model will be able to observe as input. This is done by specifying the orders by a reverse orthographical sorting of past and future context lengths, i.e., $c_t(3,4) \succ c_t(3,3) \succ c_t(2,4)$. This does not completely resolve the weak ordering, though, since two words may share both the same past and future contexts. Figure 1 shows an example BS and Figure 2 shows how it was constructed using future information.

2.2 Plackett-Luce rank loss

Let us assume that the BS, b_t, is strongly ordered by some preferred (partial) rank over $k \leq |V|$ words in V for step t and that the GT word has the first rank. Let y_t be the set of words in V and say that $b_t(m)$ indexes the element of rank m according to b_t, i.e, b_t defines a rank permutation. If we assume that y_t and w_t are sorted by rank, we can drop the indexing in the following section.

Given strongly ordered top-k ground-truths, we train our LM by learning to rank with the top-k Plackett-Luce (PL) rank loss (Plackett 1975; Cao et al. 2007; Xia, Liu, and Li 2009). The PL loss models the data as a Plackett-Luce distribution, where for any time step t,

Adams	River	is	a	tributary	to	the	Thompson	and	Fraser	Rivers
			a	tributary	,	the	Thompson	and	Sam	
			a	tributary	of	the		and	<unk>	Rivers
				tributary	the			and	Kansas	Rivers
				tributary	in			and	Platte	Rivers
				tributary	valleys			and	Missouri	Rivers
				tributary	took			and	Yellowstone	Rivers

Figure 2: BS construction for the example in Figure 1 using four orders, 2p-1f, 2p, 1p-1f, 1p and the GTs. Where '2p-1f' indicates an order which matches two past words and one future word. We show the branching sets for 'to' and 'Fraser', centred in dash lines, and the context that selected those words in solid lines. The table has been truncated.

$$p_t(y_t^1 \succ \cdots \succ y_t^k) = \prod_{i=1}^{k} p_t(y_t^i \,|\, y_t^1, \ldots, y_t^{i-1}) = \prod_{i=1}^{k} \frac{e^{w_t^i}}{\sum_{j=i}^{|V|} e^{w_t^j}}. \quad (3)$$

The PL distribution represents a generative process of sampling a softmax distribution without replacement, were we re-normalize the distribution by excluding previous samples at every ranking step. The loss is then defined as the negative log-likelihood of equation 3,

$$\sum_{i=1}^{k} \log \sum_{j=i}^{|V|} e^{w_t^j} - w_t^i = \sum_{i=1}^{k} \log \left(Z_t - \sum_{j<i} e^{w_t^j} \right) - w_t^i. \quad (4)$$

While there are many surrogate ranking losses, we choose PL since, in the case when $k = 1$, the PL distribution collapses to the softmax distribution and the loss to the corresponding CE loss. That is, we revert back to traditional language modelling. This is a beneficial property since a) we might encounter contexts during training without available ranking data, i.e., only the original GT word is in the branching set, b) we only consider the probability of the original GT during evaluation, and c) we do not wish to radically change the distribution being modelled so that PPL is comparable to previous work. The PL loss can also be efficiently implemented for GPUs and does not add significant computation over a regular softmax (see Appendix C).

When combined with a method for constructing artificial ranking ground-truths, we can view the PL loss as a form of self-supervision where the first ranking term is our original loss and all higher ranking terms are auxiliary losses. However, unlike most auxiliary losses used in self-supervised learning, the extra PL terms are a natural extension or generalization of the original loss.

The PL loss can fail to capture the inductive bias that it is more important to get the more-preferred ranks correct over the lesser-preferred ranks (Lan et al. 2014). This is relevant since the original GT is the first rank and we will have less confidence as we descend the ranks, since the amount of semantic information in a bigram is generally less than in a trigram, and so on. Lan et al. (2014) introduced discount weights on the PL terms which decreased exponentially, but, since their exact method did not work for our task, we use a *stepped* function where a static $\eta < 1$ weight is given to the top rank and the remaining $1-\eta$ is partitioned such that there is an equal difference between the weights for consecutive ranks. In practice, η acts similar to the temperature smoothing hyper-parameter used in KD.

The PL loss does not allow for weak orderings. This is a problem since the words within each individual order, O_i, have no ordering; i.e., if $O_i = \{o_1, \ldots, o_S\}$, then the S items are of an unknown ordering. This creates a partitioned preference where all the partitions define a ranking, $O_i \succ O_{i+1}$, but the ranking within each ordering is unknown. One way of handling this would be to consider the marginal distribution across all permutations for each ordering. However, the likelihood would require calculating a factorial number of terms (Ma et al. 2021). Instead, we consider a lack of known preference as an inductive bias that words in the weak orders are equally preferred. We enforce this by modifying the PL loss such that we revert back to regular softmax-cross entropy within the weak order for rankings $i, \ldots, i+S$ as $\sum_{s=i}^{i+S} \log Z_{t,i} - w_t^s$, where $Z_{t,i} = Z_t - \sum_{j<i} e^{w_t^j}$, which will optimize to a uniform distribution within the ordering. This is a novel modification to our knowledge.

3 Related work

Previous work combined N-gram information with neural LMs. Neubig and Dyer (2016); Bakhtin et al. (2018) used dynamic mixtures of N-grams and neural LMs. Neubig and Dyer (2016) found that their models performed worse than regularly trained neural LMs except on low-frequency words. Noraset, Demeter, and Downey (2018) used soft constraints for text generation with an LSTM LM. They tried two sets of constraints: those to reduce repeated words during generation and those to match Kneser-Ney bigram probabilities during generation. Their method estimates the marginal probability of the LSTM LM given some conditioning con-

straint and regularizes it using a KL-divergence against the constraint distribution. These marginal probabilities are based on a pool of generated text that is periodically updated during training. This can be seen as a form of LS against the actual generative distribution instead of the GT distribution. They showed they could better match the distribution of repeated tokens and bigram statistics against a baseline on PTB. Yang, Wang, and Downey (2019) built on this by making computation of the marginals more tractable. They regularized the LM's whole vocabulary against trigram statistics, which makes their methods very similar to KD against a trigram N-gram model. They also proposed a method for selecting useful N-grams based on an information criterion. This is important, as they need to do a single forward pass for every N-gram context they regularize. So selecting which contexts they use significantly decreases the computational cost. They trained an LSTM LM and a trigram N-gram LM. On Wiki02, their method achieved 69 PPL given a baseline of 76.

Various works implemented semantic label smoothing using pre-trained probabilistic models (Elbayad, Besacier, and Verbeek 2018; Hahn and Choi 2019; Li et al. 2019; Ghoshal et al. 2020; Liu, Shen, and Lapata 2021; Lukasik et al. 2020b). These can use various pre-trained models such as embedding models like Fasttext (Joulin et al. 2017), language representation models like BERT (Devlin et al. 2019) or language models like GPT2 (Radford et al. 2019). Lukasik et al. (2020b) smoothed over related sequences which were found using a pre-trained model and then refined using BLEU as a selection criterion. Wang et al. (2020) introduced graduated label smoothing which uses a set of binned label-smoothing values and the confidence of a pre-trained model to determine the bin to which each token belongs, improving BLEU and calibration for NMT.

Li et al. (2019) used an auxiliary KL-divergence based loss on a semantically-aware or data-dependent Gaussian prior which is parameterized using cosine similarity from Fasttext embeddings. They applied this over a range of conditional language modelling tasks including NMT, text summarization, and story telling. For story telling, their method lowered test PPL by $\approx$ 2-3 points. In general, LS can be framed as an entropy regularizer and many prior distributions can be used (Pereyra et al. 2017; Meister, Salesky, and Cotterell 2020).

Ghoshal et al. (2020) proposed a method which jointly trains a model with a set of learned smoothing parameters. Given C classes, one could implement semantic LS by learning a similarity $C \times C$ matrix where each row provides smoothing scores for a given target class. Instead, the authors approximated this with a $C \times k$ matrix with $k \lll C$. While this is a form of semantic label smoothing, it is limited in that it is based only on the target class and not the full context. They applied it to semantic parsing and question answering.

Welleck et al. (2019) introduced an unlikelihood loss which penalizes words which should have low probability. We make use of the concept of a branching set which is the words that can follow a given context. The complementary set of the branching set could be used to select negative samples for the unlikelihood loss.

Born-again distillation or self-distillation is where the student and teacher have the same specification (Furlanello et al. 2018; Hahn and Choi 2019; Yuan et al. 2020). Furlanello et al. (2018) applied born-again KD to LMs and showed that a student LSTM LM on PTB could outperform the same teacher model. Yang et al. (2019) used a top-k auxiliary loss and trained image classification models with a series of self-distillations. Tang et al. (2020) developed a top-k KD using the teacher's probability for the top-k ranks and distributing the remaining probability uniformly.

Reddi et al. (2021) recently introduced the use of top-k rank loss functions for rank-based KD using PL and pairwise hinge losses, outperforming traditional KD on a variety of traditional ranking tasks. Other forms of structured KD can also be applied to other NLP sequence tasks (Wang et al. 2021).

4 Experiments

We use the word-level Penn Treebank (PTB) and WikiText-02 (Wiki02) datasets and use ADW-LSTM (LSTM) and Transformer-XL (T-XL) students (Taylor, Marcus, and Santorini 2003; Merity et al. 2017; Merity, Keskar, and Socher 2018; Dai et al. 2019). Our LSTM uses 24.2M parameters on PTB and 33.5M on Wiki02, and our Transformer-XL uses 35.8M on Wiki02.

We propose two hypotheses: that we can re-frame the problem of label-smoothing as a form of rank-based knowledge distillation from a teacher which *only* provides rank ground-truths and that we can derive the ranks directly from a non-probabilistic N-gram teacher. In order to decouple these two hypotheses, we first use three different types of pre-trained LMs to evaluate the PL loss when used for KD then, provided that works, we apply the PL loss with an N-gram teacher model.

We choose GPT-2, BERT, and Born Again (BA) models as teachers for different reasons. GPT-2 and BERT were chosen under the assumption that these large LMs will produce better ranks than the N-grams. We tried both since the former is an auto-regressive LM which only conditions on past context and thus matches our student models, while the latter is an auto-encoding language representation model using both past and future contexts, allowing us to test if we can distil future information. BA models are also considered as they will not present data-leakage problems, unlike BERT and GPT-2 which were pre-trained on a large amount of auxiliary data. The BA models are selected as the highest performing CE baseline models.

GPT-2 and BERT use sub-word vocabularies instead of the standard word-level ones for PTB and Wiki02. We convert them by summing the sub-word hidden states to get whole-words and fine-tune them using CE.

Table 1 shows the validation performance of the teacher models. The performance of these models is better than the baseline CE models, which warrants their

Data	Model	(P-)PPL	A $\subseteq$ 1	A $\subseteq$ 2	A $\subseteq$ 3	A $\subseteq$ 5	A $\subseteq$ 10
PTB	BERT 24-layer	*1.205	0.975	0.986	0.989	0.991	0.993
	GTP-2 774M	22.810	0.437	0.525	0.573	0.630	0.701
	BA-LSTM	60.724	0.302	0.394	0.446	0.511	0.593
Wiki02	BERT 24-layer	*1.423	0.930	0.979	0.983	0.985	0.987
	GTP-2 774M	25.277	0.419	0.521	0.575	0.635	0.706
	BA-LSTM	68.437	0.296	0.394	0.448	0.512	0.592
	BA-T-XL	67.468	0.301	0.398	0.452	0.517	0.598

Table 1: Perplexity (PPL) and accuracy in the top-k (A $\subseteq k$) for word-level fined-tuned probabilistic teacher models BERT, GPT-2, Born-Again (BA) on the PTB and Wiki02 validation sets. *We report a *pseudo*-PPL for BERT.

use as teacher models. BERT out-performs GPT-2 due to using surrounding context when predicting a word where GPT-2 is limited to only past context[1]. We provide two other sanity checks; first, we provide examples of the top-k predictions for BERT, GPT-2 and the N-grams in Appendix E and, second, we plot frequency statistics for BERT and GPT-2 in Appendix D.

The N-grams used a set of contexts from 5 to 1 past tokens concurrently with 4 to 0 future tokens and a pruning cut-off of 10, i.e. 5p-4f, 5p-3f, ..., 1p-1f, 1p.

We compare a CE baseline to four other losses. The first is a top-k KL-based KD. This has been modified in three ways from traditional KD. First, we only use the top-k ranks, as did Tang et al. (2020), although we forgo placing a uniform distribution over all words not in the top-k. They reported a test PPL of 60.85 using top-k KD on PTB with AWD-LSTM using a smaller student with 9.1M parameters. Second, we post-process the predicted ranks by floating the original GTs to the top rank while keeping the top-k probabilities the same order i.e. make the GTs the most probable. Initial experimentation did not show a significant change for KL-based KD and we believed this was an important modification for rank-based KD. Third, we cycle the interpolation value between the CE and KD terms, similar to Clark et al. (2019); Jafari et al. (2021). For PL, we can forego discounting (PL), use the teacher's probabilities (PL-t), or use the stepped function (PL-s). Note that the KL also uses the teacher's probabilities. See Appendix A for further experimental details.

5 Discussion

The results in Table 2 show we can treat language modelling as a structured prediction task using a rank-based KD and that this can be done without the need of a probabilistic teacher. We discuss four main results.

First, given the proper form of discounting, either form of KD improves PPL over the CE baseline.

Second, some form of rank-based KD often significantly outperforms KL-based KD. This is true of 6/9 test-set groups, with only a single experiment showing the reverse (this is a slight conceit, since we are making post-hoc comparisons against a set of PL experiments instead of selecting the best form of discounting during the validation phase). Since, KL-based KD contains both the label order and weight information, it acts as a strong baseline. We believe that the reason the rank-based KD often does better is because matching the teacher's ranks is a less strict criterion than matching its probabilities. It may also be a better criterion when evaluating PPL, since PL may allow for the top ranks to all increase their probability together, so as long as the GT is in the top ranks, then PPL will decrease. Importantly, these results also indicate that performance of the N-grams will be due to their inherent value as a teacher and will not be hindered by the loss function. The results of the PL and PL-s experiments support the claim that the ordering of the labels provides sufficient semantic information to perform KD and that we can forego label weights (with a caveat addressed later).

Third, the N-grams act as a competitive teacher, with only GPT-2 consistently outperforming them. It seems plausible that the N-grams would surpass the BA models, since both types of KD will restrict how much the BA students can deviate from the baseline CE teacher. With BA and BERT, and LSTM students, the raw PL loss often does worse than the CE baseline. This is indicative of the PL loss failing when it is easier to optimize the lesser-preference ranks over the GT.

That the N-grams outperform BERT speaks more to BERT failing to act as a good teacher rather than the quality of the N-gram models. For BERT, it was important to use the teacher probabilities (see the KL and PL-t vs PL and PL-s experiments). We believe that reliance on the teacher's probabilities is indicative of poor ranks where the teacher's sharp probability discounts the lower ranks. A qualitative analysis of the produced ranks shows that the future information can produce ranks that do not generalize well to a student evaluated on a past-context-only task. For example, BERT can see when a sentence ends and so produces ranks which do not try to continue the sentence where GPT-2 and the N-grams will. Thus, the major caveat mentioned above is that label ordering is sufficient, provided those ranks are informative for the student's task. The

[1]These PPL results are not directly comparable as BERT does not form a valid distribution over sequences, hence using P-PPL. Also, they use different auxiliary training data.

		PTB				Wiki02							
		LSTM Student				LSTM Student				T-XL Student			
		Validation		Test		Validation		Test		Validation		Test	
Tchr	Loss	PPL	SEM	PPL	SEM	PPL	SEM	PPL	SEM	PPL	SEM	PPL	SEM
	CE	61.44	0.014	59.11	0.015	69.59	0.018	66.43	0.015	68.66	0.021	65.27	0.020
GPT-2	KL	57.43	0.012	55.46	0.010	65.41	0.008	62.79	0.007	59.43	0.009	56.70	0.008
	PL-t	57.28	0.014	55.17	0.013	65.47	0.015	62.69	0.014	59.32	0.007	56.58	0.007
	PL	58.16	0.012	56.25	0.012	65.45	0.011	62.74	0.011	**58.61**	0.007	**55.94**	0.007
	PL-s	57.63	0.008	55.67	0.007	65.34	0.008	**62.59**	0.008	59.48	0.008	56.76	0.007
BERT	KL	59.86	0.018	57.73	0.017	68.63	0.010	65.51	0.009	67.79	0.010	64.16	0.009
	PL-t	**59.38**	0.009	**57.20**	0.009	**68.04**	0.010	**64.99**	0.009	67.66	0.012	64.18	0.012
	PL	62.23	0.011	60.32	0.011	71.98	0.009	68.68	0.008	67.78	0.009	64.28	0.010
	PL-s	61.13	0.010	59.11	0.010	68.55	0.007	65.53	0.008	67.72	0.008	64.26	0.008
BA	KL	**60.14**	0.011	**57.63**	0.010	68.90	0.014	65.75	0.012	67.46	0.014	64.19	0.012
	PL-t	60.80	0.014	58.15	0.014	69.14	0.009	65.74	0.009	**66.96**	0.012	**63.46**	0.011
	PL	63.36	0.017	60.54	0.013	69.63	0.007	66.16	0.008	**66.93**	0.017	**63.58**	0.014
	PL-s	60.91	0.010	58.22	0.012	**67.96**	0.007	**64.76**	0.006	**67.10**	0.012	**63.64**	0.011
N	PL-s	59.66	0.008	57.16	0.009	67.25	0.006	64.44	0.006	66.59	0.015	63.54	0.013

Table 2: Average PPL (↓) with standard error (SEM) ($n = 30$) using an LSTM and T-XL student models and GPT-2, BERT, Born again (BA) and Ngram (N) teachers (Tchr). Bolded PL experiments exceed the in-group KL baseline with $p < .001$ using a two-tailed t-test. Bolded KL experiments exceed the best in-group PL experiment with significance. Underlined experiments are those which perform better than the N-gram model with significance.

N-grams are inherently sparse, since they only derive ranks when supported by the context. In the future, we would like to consider ways of dynamically selecting the top-k which may fix this discounting issue. This also continues the desire to model the branching set, which are differently sized for each context.

The fourth result is how well GPT-2 does when paired with the T-XL student, where we see a reduction of almost 10 PPL points. Compare this with the LSTM student which only sees a reduction of 4 points. Tang et al. (2020) suggested that certain failure cases in KD could be explained by a *capacity gap*, where the teacher's distribution is too complex to properly be distilled to the student model. We believe that the GPT-2 results are the inverse of this phenomenon, where the student falls into a good regime to match the teacher instead of out of the regime. This makes sense since, outside of the BA models, which may have limited value as teachers, GPT-2 and T-XL are the closest student-teacher pair in terms of specification. However, since the T-XL model only has 7% more parameters than the LSTM model, we believe this should be considered as a *capability gap* where the change in performance between the LSTM and T-XL is more likely due to actual model differences instead of raw capacity. Additionally, the T-XL model using PL and no discounting was competitive to the discounted forms for BERT and BA and was the best performing model for GPT-2. This is in contrast to the non-discounted PL using the LSTM student which often did worse than the CE baseline. This again speaks to a difference in capability between the two types of student models. We wish to explore this in the future.

Our results are limited to English and might not generalize to other languages. We believe that training a LM via ranking is language agnostic; however, our N-gram algorithm may not be as it assumes strong word-order information. Rank-based KD is a general idea which allows one to integrate rule-based systems into a student model. We could improve or generalize the N-grams by incorporating syntactic rules or enforcing co-references. Other uses may be to use rank-based KD to integrate an LM into a NMT system by generating target-side ranks without modifying the NMT model. It may be useful for black-box KD, where one only has access to the teacher's hard predictions (Wang 2021) or to augment small-data environments.

In this work, we offer a novel perspective on language modelling by considering it as a multi-label task and develop the methods necessary to train such a model. We connect the idea of LS to ranking via KD by showing that we can incorporate semantic similarity information from an ordered set of words into a LM without requiring associated probabilities. In our desire to forego using a pre-trained LM, we re-examined how N-gram statistics should be incorporated into a neural LM. We find that this can be done using a simple method which just considers N-gram contexts, and that this surpasses the CE baseline and is comparable to using BA and BERT as teachers. We also find that, for language modelling, a rank-based KD is often a superior method for KD.

Acknowledgements

We acknowledge the support and resources provided by the Province of Ontario, the companies sponsoring the Vector Institute, and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Bakhtin, A.; Szlam, A.; Ranzato, M.; and Grave, E. 2018. Lightweight adaptive mixture of neural and n-gram language models. *arXiv preprint arXiv:1804.07705*.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb): 1137–1155.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, 129–136.

Clark, K.; Luong, M.-T.; Khandelwal, U.; Manning, C. D.; and Le, Q. V. 2019. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5931–5937. Florence, Italy: Association for Computational Linguistics.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2978–2988. Florence, Italy: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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)*, 4171–4186.

Elbayad, M.; Besacier, L.; and Verbeek, J. 2018. Token-level and sequence-level loss smoothing for RNN language models. In *ACL-56th Annual Meeting of the Association for Computational Linguistics*, 2094–2103. ACL.

Furlanello, T.; Lipton, Z.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born Again Neural Networks. In *International Conference on Machine Learning*, 1607–1616.

Ghoshal, A.; Chen, X.; Gupta, S.; Zettlemoyer, L.; and Mehdad, Y. 2020. Learning Better Structured Representations Using Low-rank Adaptive Label Smoothing. In *International Conference on Learning Representations*.

Goodman, J. T. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4): 403–434.

Hahn, S.; and Choi, H. 2019. Self-Knowledge Distillation in Natural Language Processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, 423–430. Varna, Bulgaria: INCOMA Ltd.

Harris, Z. S. 1954. Distributional structure. *Word*, 10(2-3): 146–162.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *stat*, 1050: 9.

Jafari, A.; Rezagholizadeh, M.; Sharma, P.; and Ghodsi, A. 2021. Annealing Knowledge Distillation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2493–2504. Online: Association for Computational Linguistics.

Joulin, A.; Grave, É.; Bojanowski, P.; and Mikolov, T. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 427–431.

Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Lan, Y.; Zhu, Y.; Guo, J.; Niu, S.; and Cheng, X. 2014. Position-Aware ListMLE: A Sequential Learning Process for Ranking. In *UAI*, 449–458.

Li, Z.; Wang, R.; Chen, K.; Utiyama, M.; Sumita, E.; Zhang, Z.; and Zhao, H. 2019. Data-dependent gaussian prior objective for language generation. In *International Conference on Learning Representations*.

Liu, Y.; Shen, S.; and Lapata, M. 2021. Noisy Self-Knowledge Distillation for Text Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 692–703. Online: Association for Computational Linguistics.

Lukasik, M.; Bhojanapalli, S.; Menon, A.; and Kumar, S. 2020a. Does label smoothing mitigate label noise? In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 6448–6458. PMLR.

Lukasik, M.; Jain, H.; Menon, A.; Kim, S.; Bhojanapalli, S.; Yu, F.; and Kumar, S. 2020b. Semantic Label Smoothing for Sequence to Sequence Problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4992–4998. Online: Association for Computational Linguistics.

Ma, J.; Yi, X.; Tang, W.; Zhao, Z.; Hong, L.; Chi, E.; and Mei, Q. 2021. Learning-to-Rank with Partitioned Preference: Fast Estimation for the Plackett-Luce Model. In Banerjee, A.; and Fukumizu, K., eds., *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, 928–936. PMLR.

Meister, C.; Salesky, E.; and Cotterell, R. 2020. Generalized Entropy Regularization or: There's Nothing Special about Label Smoothing. In *Proceedings of the 58th*

Annual Meeting of the Association for Computational Linguistics, 6870–6886. Online: Association for Computational Linguistics.

Merity, S.; Keskar, N. S.; and Socher, R. 2018. An Analysis of Neural Language Modeling at Multiple Scales. *CoRR*, abs/1803.08240.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mikolov, T. 2012. *Statistical language models based on neural networks*. Ph.d. thesis, Brno University of Technology, Faculty of Information Technology.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26: 3111–3119.

Müller, R.; Kornblith, S.; and Hinton, G. E. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 4694–4703.

Neubig, G.; and Dyer, C. 2016. Generalizing and Hybridizing Count-based and Neural Language Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1163–1172.

Noraset, T.; Demeter, D.; and Downey, D. 2018. Controlling Global Statistics in Recurrent Neural Network Text Generation. In *AAAI*, 5333–5341.

Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, L.; and Hinton, G. E. 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*. OpenReview.net.

Plackett, R. L. 1975. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2): 193–202.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog 1.8*, 9.

Reddi, S.; Kumar Pasumarthi, R.; Menon, A.; Singh Rawat, A.; Yu, F.; Kim, S.; Veit, A.; and Kumar, S. 2021. RankDistil: Knowledge Distillation for Ranking. In Banerjee, A.; and Fukumizu, K., eds., *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, 2368–2376. PMLR.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Tang, J.; Shivanna, R.; Zhao, Z.; Lin, D.; Singh, A.; Chi, E. H.; and Jain, S. 2020. Understanding and Improving Knowledge Distillation. *arXiv preprint arXiv:2002.03532*.

Taylor, A.; Marcus, M.; and Santorini, B. 2003. The Penn treebank: An overview. In *Treebanks*, 5–22. Springer.

Wang, L.; and Yoon, K.-J. 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, S.; Tu, Z.; Shi, S.; and Liu, Y. 2020. On the Inference Calibration of Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3070–3079. Online: Association for Computational Linguistics.

Wang, X.; Jiang, Y.; Yan, Z.; Jia, Z.; Bach, N.; Wang, T.; Huang, Z.; Huang, F.; and Tu, K. 2021. Structural Knowledge Distillation: Tractably Distilling Information for Structured Predictor. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 550–564. Online: Association for Computational Linguistics.

Wang, Z. 2021. Zero-Shot Knowledge Distillation from a Decision-Based Black-Box Model. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 10675–10685. PMLR.

Welleck, S.; Kulikov, I.; Roller, S.; Dinan, E.; Cho, K.; and Weston, J. 2019. Neural Text Generation With Unlikelihood Training. In *International Conference on Learning Representations*.

Xia, F.; Liu, T.-Y.; and Li, H. 2009. Statistical consistency of top-k ranking. *Advances in Neural Information Processing Systems*, 22: 2098–2106.

Yang, C.; Xie, L.; Qiao, S.; and Yuille, A. L. 2019. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5628–5635.

Yang, Y.; Wang, J.-P.; and Downey, D. 2019. Using Large Corpus N-gram Statistics to Improve Recurrent Neural Language Models. 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)*, 3268–3273.

Yuan, L.; Tay, F. E.; Li, G.; Wang, T.; and Feng, J. 2020. Revisiting Knowledge Distillation via Label Smoothing Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3903–3911.

Zhao, Z.; Liu, T.; Li, S.; Li, B.; and Du, X. 2017. Ngram2vec: Learning Improved Word Representations from Ngram Co-occurrence Statistics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 244–253. Copenhagen, Denmark: Association for Computational Linguistics.