Low rank softmax can have unargmaxable classes in theory but rarely in practice

Anonymous ACL submission

Abstract

Classifiers in natural language processing (NLP) often have a large number of output classes. For example, neural language models (LMs) and machine translation (MT) models both predict tokens from a vocabulary of thousands. The softmax output layer of these models typically receives as input a dense feature representation, which has much lower dimensionality than the output. In theory, the result is some words may be impossible to predict via argmax, irrespective of input features, and empirically, this has been shown to happen in small language models (Demeter et al., 2020). In this paper we ask whether it can happen in practical large language models and translation models. To do so, we develop algorithms to detect such *unargmaxable* tokens in public models. We find that that 13 out of 150 models do indeed have such tokens; however, they are very infrequent and unlikely to impact model quality. We release our algorithms and code to the public.[1]

1 Introduction

Probabilistic classifiers with a large number of output classes are commonplace in NLP. For example, the vocabulary size of contemporary LMs and MT models varies from tens to hundreds of thousands (Liu et al., 2020). Recent advances in modelling such large vocabularies have mostly been made by improving neural network feature encoders (Devlin et al., 2019; Liu et al., 2019; Conneau et al., 2020). But irrespective of the encoder's usefulness, projecting lower dimensional features to higher dimensional outputs constrains expressivity, with consequences that are not well understood.

In this work we elaborate on the consequences that arise when the number of output classes $|C|$ is greater than the dimensionality d of the classification layer inputs. For example, MT models often have subword vocabularies of size $|C| \approx 30000$, but have $d \approx 1024$. These models are low rank and thus less expressive (Yang et al., 2018; Ganea et al., 2019); more importantly, they cannot represent some outputs. Demeter et al. (2020) recently highlighted that this weakness[2] occurs in softmax LMs, showing that, in theory, some tokens can never be assigned the highest probability for any input, a phenomenon they call **Stolen Probability**. Figure 1 illustrates how it occurs.

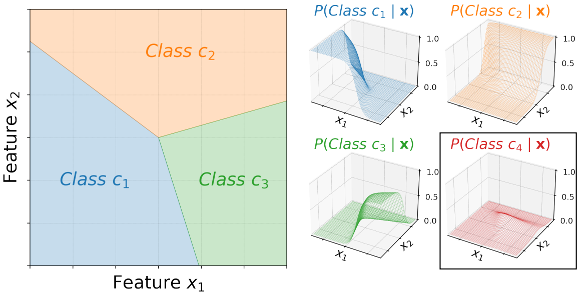


Figure 1: Illustration of *Stolen Probability*: Class c_4 can never be predicted using argmax for this softmax classifier with $|C| = 4$ classes in $d = 2$ dimensions. On the left, each input point $\mathbf{x}$ is colored according to the class assigned the largest probability; note that while c_1, c_2 and c_3 surface as regions, c_4 does not. On the right we similarly show that there is no direction in the input space for which c_4 has the largest probability.

While Demeter et al. (2020) highlighted the theoretical problem and showed that it occurs in small LMs, they were unable to test larger LMs. In this paper we ask: *Does Stolen Probability arise in large models used in practice?* To answer this question, we develop algorithms to identify unargmaxable tokens. We tested 7 LMs and 143 MT models. Out of those, only 13 of the MT models exhibit Stolen Probability, and even for those cases the tokens are all noisy and infrequent. We conclude that most practictioners do not need to worry about Stolen Probability, and we provide new tools so

[1] Code available at REDACTED

[2] This problem was highlighted by Cover (1967) and has an interesting history of independent discovery (Smith, 2014).

that they can confirm this on their own models.

Our contributions are the following:

- We explain how Stolen Probability can arise as a consequence of a rank constrained softmax layer.
- We extend the work in (Demeter et al., 2020) with algorithms that provide an exact answer rather than an approximate one while also including the softmax bias term in the analysis.
- We verify a large number of commonly used publicly available language and translation models for Stolen Probability.
- We release our algorithm so that others can inspect their models.

2 The Softmax Bottleneck and Stolen Probability

2.1 Softmax Bottleneck

Neural network layers with more outputs than inputs impose low rank constraints.[3] Such constraints commonly exist as **bottlenecks** in hidden neural network layers, e.g. eutoencoders (Hinton and Zemel, 1994) and projection heads in multi-head transformers (Vaswani et al., 2017) among others. While bottlenecks make a model less expressive by restricting the functions it can represent, they are desirable both computationally and as a form of regularisation that can improve modelling.

In contrast, herein we focus on the undesirable properties of a softmax output layer with a low rank parametrisation, also known as a **Softmax Bottleneck** (Yang et al., 2018). The crucial difference is that a Softmax Bottleneck is usually not followed by a non-linear transformation, and as such the rank constraint limits expressivity in a very rigid way by restricting outputs to a subspace.[4] This constraint was shown to hurt LM perplexity (Yang et al., 2018) and non-linear augmentations have been proposed as improvements (Yang et al., 2018; Kanai et al., 2018; Ganea et al., 2019). Ganea et al. (2019, Theorem 2) further elaborated on the loss of expressivity due to the Softmax Bottleneck by showing that the minimum cross entropy loss that can be achieved by a rank constrained softmax is greater or equal to that obtained by a softmax with increased rank. In this work we discretise the output space of softmax and quantify the loss in expressivity more tangibly by thinking in terms of unrealisable class rankings. From this interpretable perspective we will see that a fixed number of rankings is not realisable and Stolen Probability can arise as a consequence.

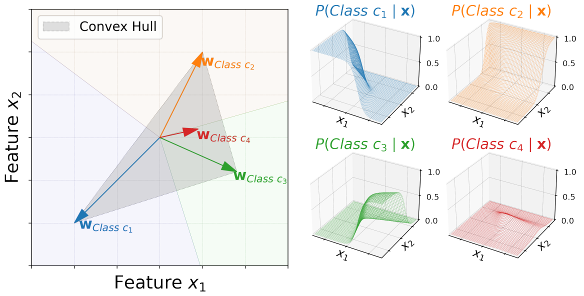


Figure 2: Illustration of the culprit softmax weights for Stolen Probability in Figure 1. On the left each vector is a row of the softmax weights $\mathbf{W} \in \mathbb{R}^{4\times 2}$. c_4 is interior to the convex hull, the triangle formed by c_1, c_2 and c_3.

2.2 Stolen Probability

Demeter et al. (2020) analyse what happens if a class weight vector of a softmax layer is interior to the convex hull of all other class weight vectors. They show that the interior class probability is bounded above by the probability of at least one class on the convex hull, making it unargmaxable (see Figure 2 and Cover, 1967, Figure 1). However, in their analysis they did not address softmax layers that include a bias term. We address this limitation in Section 3, thus enabling us to search for Stolen Probability in any released model.

To detect whether Stolen Probability arises in models without a bias term, the authors introduce an approximate algorithm that asserts whether a weight vector is internal to the convex hull. It is approximate since their method had a precision approaching 100% but 68% recall when compared to an exact algorithm (Qhull Barber et al., 1996) on the first 10 dimensions of a softmax LM. In Section 3.3 we introduce an exact algorithm to detect unargmaxable tokens with certainty.

The authors use their approximate algorithm to show that AWD-LSTM LMs (Merity et al., 2018) “steal” probability from candidate bounded words when contrasted to the probabilities assigned by a smoothed n-gram LM. However, they find that as they increase the dimensionality d of the softmax weights to 200, the effect of Stolen Probability begins to dissipate. This raises the question of whether Stolen Probability is of importance for

[3] A layer can also be made low rank if any weight vectors are made collinear, but we do not consider this case here.

[4] A linear subspace if no bias term is present and an affine subspace otherwise.

neural models used in practice which also have larger softmax weight dimensionality. In this paper we address this question for MT models of dimensionality $d \in [256, 512, 1024]$. We choose MT models since they have more practical use cases than (generative) LMs: if Stolen Probability exists in an MT model, then the affected tokens can never be produced when using greedy decoding. In our experiments we find that Stolen Probability arises in limited cases, which however are not of grave importance.

3 Detecting Stolen Probability

In order to quantify whether Stolen Probability arises in released LMs and MT models, we first need to introduce tractable algorithms for detecting it. In this section we explain how Stolen Probability can arise when we have a Softmax Bottleneck. Then, we introduce a fast approximate algorithm and a slow exact algorithm which we combine to detect vocabulary tokens that cannot be predicted.

3.1 Definitions

3.1.1 Softmax

A softmax layer gives us the probability assigned to a target class c_t for an input feature vector $\mathbf{x} \in \mathbb{R}^d$ as follows:

$$\begin{aligned} P(C = c_t \mid \mathbf{x}) &= \frac{e^{\mathbf{w}_{c_t}^\top \mathbf{x} + b_{c_t}}}{\sum_i e^{\mathbf{w}_{c_i}^\top \mathbf{x} + b_{c_i}}} \\ &= \mathrm{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})_{c_t} \end{aligned}$$

where $\mathbf{W} \in \mathbb{R}^{|C| \times d}$ are the class weight vectors stacked row by row, and $\mathbf{b} \in \mathbb{R}^{|C|}$ is the bias term. The above are used to compute the **logits** $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$. In what follows, we will refer to the feature activations $\mathbf{x}$ in $\mathbb{R}^d$ as the **input space** and the logits $\mathbf{y}$ in $\mathbb{R}^{|C|}$ as the **output space** of the softmax layer.

3.1.2 Discretising the output space into Permutations

As we saw in Figure 1, there are certain arrangements of softmax weights for which a target class c_t cannot be surfaced as the argmax. To understand this phenomenon, it will be helpful to discretise the outputs to a finer granularity: rankings. In order for a classifier to predict a class c_t it must rank c_t above all other classes by assigning it the largest probability. From this perspective, a classifier assigns each input $\mathbf{x}$ a permutation $\boldsymbol{\pi}$ that ranks the class indices in increasing order of probability.

$$\boldsymbol{\pi} : P(c_{\boldsymbol{\pi}_{[1]}} \mid \mathbf{x}) < P(c_{\boldsymbol{\pi}_{[2]}} \mid \mathbf{x}) < \ldots < P(c_{\boldsymbol{\pi}_{[|C|]}} \mid \mathbf{x})$$

As an example, if we have 4 classes and obtain probabilities $P(C \mid \mathbf{x}) = \begin{bmatrix} .2 & .4 & .1 & .3 \end{bmatrix}^\top$ we assign $\mathbf{x}$ the permutation $\boldsymbol{\pi}_{3142}$, since $P(c_3 \mid \mathbf{x}) < P(c_1 \mid \mathbf{x}) < P(c_4 \mid \mathbf{x}) < P(c_2 \mid \mathbf{x})$. We can readily obtain the coarser argmax decision (c_2) by reading off the last index of the permutation.

3.2 How can Stolen Probability arise?

Stolen probability arises for class c_t when all permutations that rank c_t above the rest cannot be realised due to rank constraints. We explain how by combining the following two observations.

Observation 1. ***We can discretise $\mathbb{R}^{|C|}$ into regions corresponding to permutations by segmenting the space with hyperplanes.***

The hyperplanes that partition space into regions $\mathcal{R}_{\boldsymbol{\pi}}$ corresponding to permutations are a well known structure in Combinatorics, the **Braid Hyperplane Arrangement** [5] (Stanley, 2004). The Braid Arrangement for 3 and 4 classes is illustrated in rows 1 and 2 of Figure 3 respectively.

In order to be able to rank the classes according to permutation $\mathcal{R}_{\boldsymbol{\pi}}$, our network needs to be able to map an input $\mathbf{x}$ to region $\mathcal{R}_{\boldsymbol{\pi}}$ in the output space. However, this is not always possible when we have a Softmax Bottleneck as we elaborate below.

Observation 2. ***When we have rank constraints only a subspace of $\mathbb{R}^{|C|}$ is feasible.***

Case i) $\mathrm{softmax}(\mathbf{W}\mathbf{x})$. By calculating $\mathbf{y} = \mathbf{W}\mathbf{x}$, the class logits $\mathbf{y}$ are a linear combination of d columns of $\mathbf{W}$. Therefore, when $d < |C|$ we can only represent a d-dimensional subspace of $\mathbb{R}^{|C|}$ at best. This feasible subspace is illustrated as a grey plane in the middle column of Figure 3.

Case ii) $\mathrm{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$. If we also have a bias term $\mathbf{b}$ the model can choose how to offset the subspace. When the bias term $\mathbf{b}$ is not in the column space of $\mathbf{W}$ the zero vector $\mathbf{0}$ is no longer a feasible $\mathbf{y}$ and instead of a linear subspace we have an affine subspace. See Figure 7 in the Appendix for an illustration comparing the two cases.

Corollary 1. *A softmax classifier parametrised by* $\mathbf{W}$ *and* $\mathbf{b}$ *can rank classes in the order of permutation* $\boldsymbol{\pi}$ *iff the affine subspace spanned by* $\mathbf{W}$

[5] See Appendix B for more details on hyperplane arrangements and the Braid Arrangement specifically.

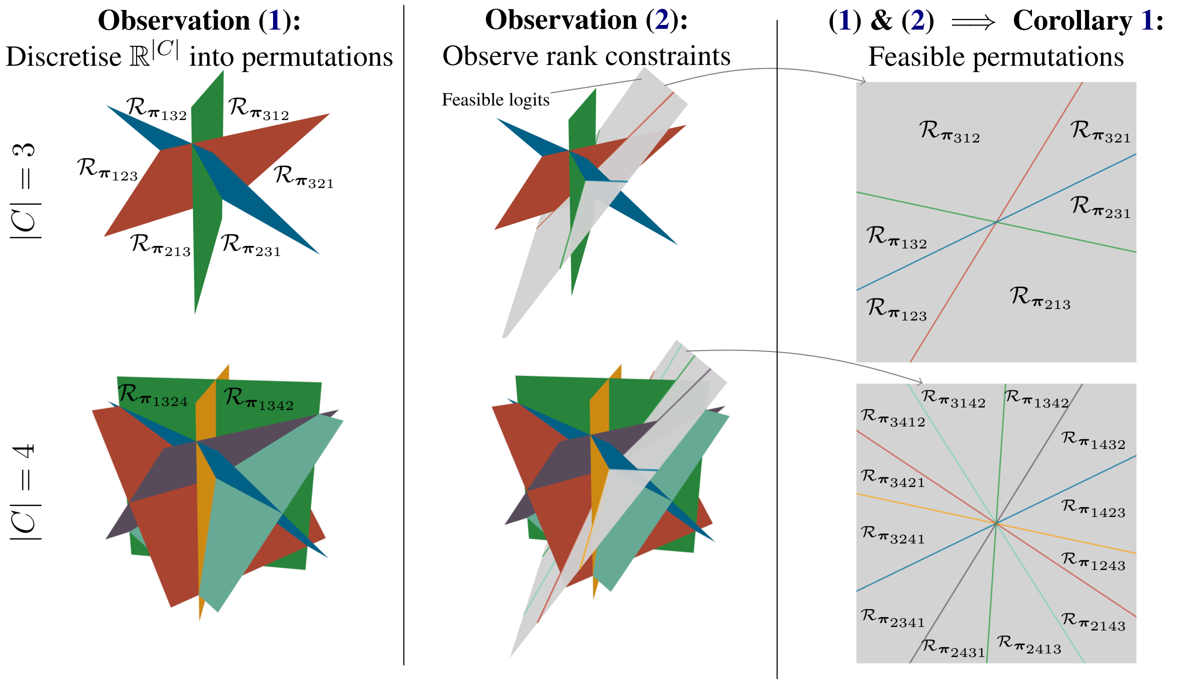

Observation (1):
Discretise $\mathbb{R}^{|C|}$ into permutations
Observation (2):
Observe rank constraints
(1) & (2) $\Longrightarrow$ Corollary 1:
Feasible permutations
$|C| = 3$
$|C| = 4$
Feasible logits
$\mathcal{R}_{\pi_{132}}$
$\mathcal{R}_{\pi_{312}}$
$\mathcal{R}_{\pi_{123}}$
$\mathcal{R}_{\pi_{321}}$
$\mathcal{R}_{\pi_{213}}$
$\mathcal{R}_{\pi_{231}}$
$\mathcal{R}_{\pi_{1324}}$
$\mathcal{R}_{\pi_{1342}}$
$\mathcal{R}_{\pi_{3142}}$
$\mathcal{R}_{\pi_{3412}}$
$\mathcal{R}_{\pi_{1432}}$
$\mathcal{R}_{\pi_{3421}}$
$\mathcal{R}_{\pi_{1423}}$
$\mathcal{R}_{\pi_{3241}}$
$\mathcal{R}_{\pi_{1243}}$
$\mathcal{R}_{\pi_{2341}}$
$\mathcal{R}_{\pi_{2143}}$
$\mathcal{R}_{\pi_{2431}}$
$\mathcal{R}_{\pi_{2413}}$

Figure 3: Illustration of Corollary 1 (3^{rd} column) as a result of Observation 1 (1^{st} column) and Observation 2 (2^{nd} column) for $\mathrm{softmax}(\mathbf{Wx})$, $\mathbf{W} \in \mathbb{R}^{|C|\times d}$, $d = 2$. Planes truncated for ease of visualisation. **Top row**: In the left column we see the Braid Arrangement for 3 classes partitioning the output space into 6 regions that correspond to permutations: class rankings in increasing order of probability. In the middle column we see that because $d = 2$ we can only map $\mathbf{x}$ to the feasible logits, a plane (grey) defined by $\mathbf{W}$. Therefore, in the right column we see that we can only represent permutations that correspond to the regions we can intersect with this plane. For $|C| = 3$ we can still represent all 6 rankings of 3 classes since any plane in general position will intersect all 6 regions. **Bottom row**: The Braid Arrangement for 4 classes. Since $d < |C| - 1$ the plane can only intersect 12 regions so only 12/24 permutations are feasible. As an example, we see that the plane intersects region $\mathcal{R}_{\pi_{1342}}$ but not $\mathcal{R}_{\pi_{1324}}$ and hence π_{1342} is feasible while π_{1324} is not. In fact, the orientation of the plane is such that none of the 6 $\mathcal{R}_{\pi_{*4}}$ regions are intersected, so as in Figures 1 and 2 c_4 cannot be ranked above c_1, c_2 and c_3 and Stolen Probability arises.

and* b *intersects region* R_π *of the Braid Arrangement [6]. When $d < |C| - 1$ there are regions that cannot be intersected [7]. The feasible permutations in our example correspond to the sections formed on the grey plane illustrated in the rightmost column of Figure 3. Note that for $|C| = 4$ only 12 out of 24 regions can be intersected.

As we make the Softmax Bottleneck narrower by reducing the dimension d of the softmax inputs, more permutations become infeasible (Good and Tideman, 1977; Kamiya and Takemura, 2005). Importantly, if we choose $|C|$ and d and whether to use a bias term, changing the values of the softmax weights changes the set of feasible permutations but not the cardinality of the set (Cover, 1967; Smith, 2014). See Appendix C for more details.

Corollary 2. ***Stolen Probability occurs for class*** c_t ***when any permutation that would rank class*** c_t ***above all other classes is infeasible.***

3.2.1 Effect of softmax bias term

Without a bias term the regions corresponding to permutations are unbounded (see the rightmost column of Figure 3). As such, imposing any range restrictions on the softmax layer inputs $\mathbf{x}$ does not change the feasible regions as long as the restriction includes the origin. However, when we introduce a bias term we also get bounded regions (see Figure 7 in the Appendix that contrasts the two situations). Therefore, in this case the scale of the inputs to the softmax layer also matters. If the inputs do not have a large enough range, there will be regions that exist but cannot be reached by the feature encoder.

3.3 Exact algorithm

Given a softmax layer parametrised by $\mathbf{W}$ and $\mathbf{b}$, is there a class c_t that has Stolen Probability? First we describe a slow but exact algorithm.

An exact algorithm will either prove class c_t has

[6] This insight of slicing the Braid Arrangement was introduced in Kamiya et al. (2011).

[7] When $d = C - 1$ we can still intersect all regions, because the Braid Arrangement always has rank $|C| - 1$ (all its normal vectors are perpendicular to the $\mathbf{1}$ vector).

no Stolen Probability by returning a feasible point $\mathbf{x} : \operatorname{argmax}(\mathbf{W}\mathbf{x} + \mathbf{b}) = c_t$ or it will prove c_t is bounded by verifying no such point exists.

To check if a region exists that ranks c_t above all others, we need to find an input $\mathbf{x} \in \mathbb{R}^d$ that satisfies the following constraints:

$$P(c_i \mid \mathbf{x}) < P(c_t \mid \mathbf{x}), \quad \forall i : 1 \leq i \leq |C|, i \neq t$$

Each of the above constraints is equivalent to restricting $\mathbf{x}$ to a halfspace (see Appendix A). Hence, to enforce all above inequalities $\mathbf{x}$ is restricted to an intersection of halfspaces.

$$\begin{gathered} (\mathbf{w}_{c_i} - \mathbf{w}_{c_t})^\top \mathbf{x} + (b_{c_i} - b_{c_t}) < 0 \\ \forall i : \quad 1 \leq i \leq |C|, \quad i \neq t \end{gathered} \tag{1}$$

If the intersection of halfspaces is empty, there is no $\mathbf{x}$ for which class c_t can be ranked above all others - and hence Stolen Probability occurs. Finding a point in an intersection of halfspaces can be solved via linear programming, albeit we found the algorithm to be slow in practice for $d > 100$.

3.3.1 Chebyshev Center linear programme

The Chebyshev center of a polytope (Boyd et al., 2004, p. 417) is the center of the largest ball of radius r that can be embedded within the polytope. We can find the Chebyshev center $\mathbf{x}_c$ and the radius r with the following linear programme.

$$\begin{aligned} \text{maximise} \quad & r \\ \text{subject to} \quad & \mathbf{w}_i^\top \mathbf{x}_c + r\|\mathbf{w}_i\|_2 \leq b_i, \quad {\scriptstyle 1 \leq i \leq |C|-1} \\ & \mathbf{x}_c \leq 100 \\ & \mathbf{x}_c \geq -100 \\ & r > 0 \end{aligned}$$

Where $\mathbf{w}_i = \mathbf{w}_{c_i} - \mathbf{w}_{c_t}$ and $b_i = b_{c_i} - b_{c_t}$, $\forall i : c_i \neq c_t$. We further constrain $\mathbf{x}_c$ to guarantee the feasible set is bounded, since the Chebyshev center is not defined otherwise. This constraint also captures the fact that neural network activations cannot be arbitrarily large.

If the above linear programme is feasible, we know that class c_t is unbounded and we also get a lower bound on the volume of the region for which it is solvable by inspecting r. On the other hand, if the linear programme is infeasible, c_t is bounded in probability.

3.4 Approximate algorithm

The exact algorithm was too slow to run for the whole vocabulary. In order to avoid running the exact algorithm for every single vocabulary item, we developed an incomplete algorithm (Kautz et al., 2009) with a one-sided error, which can quickly rule out most tokens, leaving only a small number to be checked by the exact algorithm. It proves that c_t is **unbounded** by finding an input $\mathbf{x}$ for which c_t has the largest activation. Unlike the exact algorithm, if no solution exists it cannot prove that the token is **bounded**. Hence we terminate our search after a predetermined number of steps. However, not finding a solution does not necessarily mean that this token is bounded. We denote any tokens not found to be bounded by the approximate algorithm as **approx bounded** and we run the exact algorithm on them. An illustration of the way we combine the exact and approximate algorithms to decide whether class c_t is bounded in probability can be found in Figure 4.

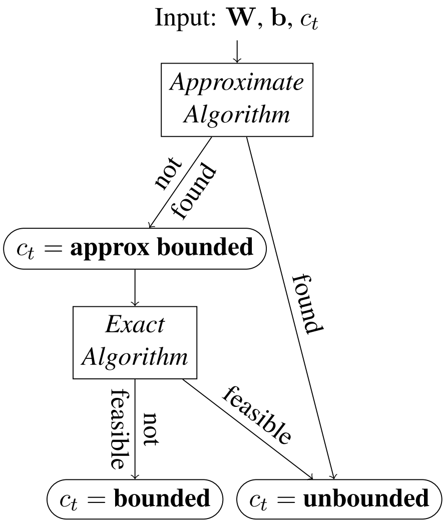


Figure 4: Algorithm to detect Stolen Probability for class c_t. We first run the approximate algorithm, which quickly proves most vocabulary tokens are unbounded. If it fails to find a solution in N steps, we rely on the exact algorithm to either find a solution or prove there is no solution and the token is bounded.

3.4.1 Braid Reflect

The idea behind this approximate algorithm is to use the Braid Hyperplane Arrangement as a map to guide us towards a point $\mathbf{x}$ for which c_t has the largest activation. To show that class c_t is not bounded, it suffices to find an input $\mathbf{x}$ for which the largest probability is assigned to c_t. Empirically we found this to be easy for most classes.

We begin by interpreting the actual weight vector as the candidate input $\mathbf{x} = \mathbf{W}_{c_t:}^\top$. We do so since the dot product of two vectors is larger when

```
Algorithm 1: Braid reflection step
  Data: Class index $c_t$, $\mathbf{x} \in \mathbb{R}^d$,
        $\mathbf{W} \in \mathbb{R}^{|C| \times d}$, $\mathbf{b} \in \mathbb{R}^{|C|}$
1 $c_i = \text{argmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$
2 $\mathbf{w} = (\mathbf{W}_{c_t:} - \mathbf{W}_{c_i:})^\top$
3 $b = \mathbf{b}_{c_t} - \mathbf{b}_{c_i}$
4 $\mathbf{w}' = \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
5 $d = \mathbf{w}'^\top \mathbf{x}$
6 $\mathbf{x} = \mathbf{x} - 2(d + \frac{b}{\|\mathbf{w}\|_2})\mathbf{w}'$
```

Figure 5: Move $\mathbf{x}$ to region where $P(c_t) > P(c_i)$.

the two vectors point in the same direction[8]. While the magnitude of the vectors affects the dot product, we found the above initialisation worked well empirically. When c_t is not the argmax for $\mathbf{x}$ and c_i is instead, Relation 1 for c_i and c_t will have the wrong sign. The sign of this relation defines which side of the Braid hyperplane for c_i and c_t we are on. To correct the sign, we construct the normal vector and offset (Lines 2, 3 in Figure 5) of the Braid hyperplane, compute the distance of $\mathbf{x}$ from it (Line 5), and reflect $\mathbf{x}$ across it (Line 6). We repeat the above operation until we get c_t to be the argmax or we give up after N steps.

4 Experiments

Do publicly released LMs and MT models have classes that are bounded in probability? In this section we use the combined algorithm introduced in Figure 4 to search models for Stolen Probability.

We test 7 LMs and 143 MT models. We find that Stolen Probability only occurs in 13 MT models, but this mostly affects infrequent and noisy vocabulary tokens. We therefore do not expect Stolen Probability to affect translation quality per se.

We also find that nearly all vocabulary tokens of LMs and student MT models can be verified with less than 10 steps of the approximate algorithm. In contrast, other MT models need thousands of steps and also rely on the exact algorithm. In this sense, models that need few steps of the approximate algorithm are easy to verify: the search problem for their arrangement of softmax weights is easier.

Throughout the following experiments we assumed the softmax inputs were bounded in magnitude for all dimensions $-100 \leq x_i \leq 100$. As we mentioned in Subsection 3.2.1, if we have a softmax bias term, there are bounded regions. If the bounded regions are large, even though the outputs are not theoretically bounded, they are practically bounded since neural network feature encoders cannot produce arbitrarily large activations and some regions may be unreachable[9]. For the approximate algorithm, we search for a solution with a patience of $N = 2500$ steps and resort to the exact algorithm if the approximate method fails or returns a point outside the aforementioned bounds. We use Gurobi (Gurobi Optimization, 2021) as the linear programme solver. The experiments took 3 days to run on an AMD 3900X 12-core CPU using 10 threads and 64Gb of RAM.

4.1 Language Models (0/7 bounded)

We checked 7 widely used Language Models for Stolen Probability. While some of these models such as BERT (Devlin et al., 2019) are not directly used for generation, a recent trend is to use these large LMs as prompt models (Liu et al., 2021) for few shot learning. A prompt model obviates the need for a separate classifier by rephrasing a classification task as slot filling given a task specific template. Prompt approaches commonly choose the answer for the slot by argmaxing the softmax distribution obtained by a LM. Hence we verify that there are no answers that are unargmaxable.

BERT, RoBERTa (Liu et al., 2019), XLM-RoBERTa (Conneau et al., 2020) and GPT2 (Radford et al., 2019) did not exhibit any bounded tokens and can be assessed without resorting to the exact algorithm (see Table 4 in the Appendix). Moreover, the LMs were very easy to verify with the approximate algorithm requiring less than 1.2 steps per token on average.

4.2 Machine Translation (13/143 bounded)

model source	Helsinki	FAIR	Edinburgh	Bergamot
bounded	13/32	0/4	0/82	0/25
dataset	OPUS	WMT'19	WMT'17	multiple[10]
architecture	Transf	Transf	LSTM	Transf
input dim	512	1024	500,512	256,512,1024
softmax bias	✓	✗	✓	✓
tied embeds	enc+dec+out	dec+out	dec+out	enc+dec+out

Table 1: Results for the MT models we verified.

We first focus on models which we found to have Stolen Probability (**bounded**) and then briefly de-

[8] $\mathbf{a}^\top \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos\theta$ is maximised for $\theta = 0$

[9] The validity of our assumption is only relevant for models we find to be bounded. We therefore verified that $-100 \leq \mathbf{x} \leq 100$ holds for two of them, see Appendix G.

[10] https://github.com/browsermt/students

scribe models that were not. A summary of the results and characteristics of the models we checked can be seen in Table 1. More detailed results can be found in Tables 5, 6, 7 and 8 in the Appendix.

Helsinki NLP OPUS (13/32 bounded). The 32 models we use for this subset of experiments are MT models released through Hugging Face (Wolf et al., 2020). We use models introduced in Tiedemann and Thottingal (2020). These models are trained on subsets of OPUS. All models are transformer models trained using Marian (Junczys-Dowmunt et al., 2018). They include a bias term and have tied encoder, decoder and output embeddings of dimensionality 512.

Stolen Probability, if present, will affect generation in the target language. We therefore restrict our analysis to the target language vocabulary. To facilitate this, we inspect translation models for which the source and target languages have different scripts. We explore 32 models with source and target pairs amongst Arabic (ar), Hebrew (he), English (en), German (de), French(fr), Spanish (es), Finnish (fi), Polish (pl), Greek (el), Russian (ru), Bulgarian (bg), Korean (ko) and Japanese (ja). We rely on the script to disambiguate between source and target language and discard irrelevant tokens from other languages. We also ignore vocabulary tokens containing digits and punctuation.

In Figure 6 we can see the number of Byte Pair Encoding (BPE) (Sennrich et al., 2016) tokens that were bounded for these models, sorted in decreasing order. As can be seen, Stolen Probability does not occur for any tokens for 19/32 language pairs. For the remaining 13 languages, while there can be quite a few bounded tokens, most would not be expected to affect translation quality.

Out of the set of 427 unique bounded BPE tokens, 307/476 are single character subword tokens and only 2 are word stem BPE segments: *erecti* (bg-en) and Предварительны (en-ru) which means "preliminary" in Russian. The rest include the <*unk*> token and what seem to be noisy subword unicode tokens such as κΚ́Κ́κ́, ὶῖῖ and ἄὐῇ.

On closer inspection of the SentencePiece tokeniser we found that both Предварительны and *erecti* come up as tokenisation alternatives that make them rare and irregular. We found that the Предварительны token was rare since it is capitalised and only occurs once, while another occurrence was caused by a BPE segmentation corner case due to Unicode token variation of Предварительны-е. Other mentions having Предварительны as a substring were split differently. In a similar vein, we found that the *erecti* token occurred due to BPE corner cases for *erecti-0-n*, *erecti-lis-)*, *erecti-l*, *erecti-*. and erecti-cle many of which are misspellings or rare word forms from clinical text. As such, the impact of these tokens being bounded is small since there are alternative ones the MT model can prefer over them which could even correct spelling mistakes.

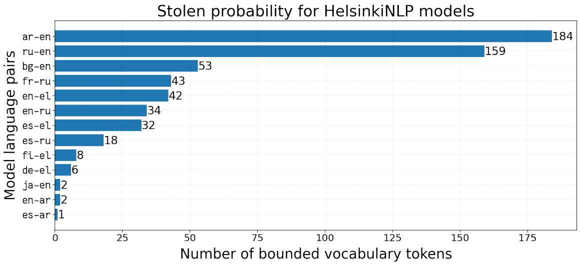


Figure 6: 13/32 HelsinkiNLP models have vocabulary tokens that cannot be predicted using greedy decoding.

FAIR WMT'19 (0/4 bounded). We checked 4 FAIR models (en-ru, ru-en, en-de, de-en) submitted to WMT'19 (Ng et al., 2019). These transformer models have softmax weights of dimensionality 1024 and no softmax bias term.

None of the FAIR models were found to have Stolen Probability, but for some tokens we had to rely on the exact algorithm to show this.

Edinburgh WMT'17 (0/82 bounded). These WMT'17 submissions (Sennrich et al., 2017) were ensembles of left-to-right trained models (l2r) and right-to-left trained models (r2l). These were LSTMs trained with Nematus using softmax weight dimensionality 500 or 512 and softmax weights tied with the decoder input embeddings. The models include a bias term.

None of the models have Stolen Probability. However, we found that models that comprise an ensemble varied a lot in how easy it was to show that the vocabulary was unbounded, despite them differing solely in the random seed used for weight initialisation. As an example, zh-en.l2r(1) had 8 tokens that needed to be verified with the exact algorithm, zh-en.l2r(2) had 3 and zh-en.l2r(3) had 366. This highlights that random initialisation alone is enough to lead to very different arrangements of softmax weights.

Bergamot (0/25 bounded). The Bergamot project[11] model repository contains both large

[11] https://browser.mt

transformer-base and transformer-big teacher models, as well as small knowledge distilled (Kim and Rush, 2016) student models. Student models have $d = 256$ (tiny) or $d = 512$ (base), while teacher models have $d = 1024$. Interestingly, we find that it is easier to show that student models are unbounded when compared to teacher models, despite student models having softmax weights $1/2$ or $1/4$ the dimensions of the teacher model.

5 Discussion

We conclude from our experiments that *Stolen Probability is possible, but it rarely occurs in practice* for tokens that would lead to irrecoverable errors in the MT models we checked. It is challenging to make exact claims about why Stolen Probability occurs because the models we tested varied in so many ways. However, we observed some general trends which we outline below.

5.1 Infrequent tokens are the victims

The most general observation is that the tokens that are more likely to be bounded or are hard to prove to be unbounded are the infrequent ones. This can be seen in Figures 12 and 13 in the Appendix, where the x-axis contains the vocabulary of the models sorted left to right by increasing frequency. Each dot represents the number of steps needed to check whether a token is bounded or not, and as can be seen the values to the right are generally much higher than those to the left.

5.2 Some models are easier to verify

We found that the LMs and student MT model vocabularies can be shown to be unbounded with one step of the approximate algorithm on average. On the other hand, for Helsinki NLP and FAIR MT models more than 10 steps were needed.

To put the above observations into context, we also check the behaviour of our algorithms on randomly initialised parameters. If we initialise a softmax layer of $|C| = 10000$ classes using a uniform distribution $U(-1, 1)$ we do not expect Stolen Probability to occur after $d = 30$ (see Figure 11 in the Appendix). Moreover, any randomly initialised parameters can be checked using the approximate algorithm with fewer steps as we increase d.

From this perspective it is therefore surprising that student models were easier to show to be unbounded than the teacher models, despite the softmax weight dimensionality of the student models being much lower (256 for tiny, versus 1024 for teacher). This shows that effective neural MT models do not need to be hard to check, but nevertheless neural models trained on the original data can sometimes converge to such an arrangement of weights.

6 Related Work

Other works have observed limitations of the softmax layer when modelling infrequent classes for image classification (Kang et al., 2020) and rare words for MT (Nguyen and Chiang, 2018; Raunak et al., 2020). They show that normalising the magnitude of the softmax weight vectors improves predictions for infrequent classes. However, the motivation for weight normalisation is guided empirically. From the perspective of this work, weight normalisation provably prevents Stolen Probability from arising when a softmax layer has no bias term. For more details, see Section D in the Appendix.

7 Conclusions and Future Work

In this work we discretised the outputs of softmax and showed how dimensionality constraints shrink the set of feasible class rankings and can lead to some classes being impossible to predict using argmax. In our experiments we demonstrated that while neural MT models can have vocabulary tokens that are bounded in probability, this does not occur often in our experiments. Moreover, for the models we tested we would not expect discernible differences in translation quality because the bounded tokens are noisy and infrequent. We release an algorithm for detecting whether some classes are bounded in probability with the hope that this will be helpful to the wider community working on a plethora of different models where the observed phenomena may vary.

In future work we aim to investigate any learnability consequences more closely. As we saw, when using an approximate search algorithm, some models are much harder to show to be bounded than others. Since gradient descent algorithms are also iterative search algorithms seeking optimal parameters, we hypothesise that it will be challenging to train neural network encoders to map activations to regions of the input space that a search algorithm cannot find easily. Hence, while Stolen Probability may not be present because of constraints imposed by the softmax parameters of the last layer, it may practically be present because of difficulties encountered by the encoder.

References

C. Barber, D. Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22:469–483.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Thomas M. Cover. 1967. The number of linearly inducible orderings of points in d-space*. *Siam Journal on Applied Mathematics*, 15:434–439.

David Demeter, Gregory Kimmel, and Doug Downey. 2020. Stolen probability: A structural weakness of neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2191–2197, Online. Association for Computational Linguistics.

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.

Octavian Ganea, Sylvain Gelly, Gary Bécigneul, and Aliaksei Severyn. 2019. Breaking the softmax bottleneck via learnable monotonic pointwise non-linearities. In *ICML*, pages 2073–2082.

I.J Good and T.N Tideman. 1977. Stirling numbers and a geometric ,structure from voting theory. *Journal of Combinatorial Theory, Series A*, 23(1):34–45.

Gurobi Optimization. 2021. Gurobi Optimizer Reference Manual.

Sibylle Hess, Wouter Duivesteijn, and Decebal Constantin Mocanu. 2020. Softmax-based classification is k-means clustering: Formal proof, consequences for adversarial attacks, and improvement through centroid based tailoring. *ArXiv*, abs/2001.01987.

Geoffrey E Hinton and Richard Zemel. 1994. Autoencoders, minimum description length and helmholtz free energy. In *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Hidehiko Kamiya and Akimichi Takemura. 2005. Characterization of rankings generated by linear discriminant analysis. *Journal of multivariate analysis*, 92(2):343–358.

Hidehiko Kamiya, Akimichi Takemura, and Hiroaki Terao. 2011. Ranking patterns of unfolding models of codimension one. *Advances in Applied Mathematics*, 47(2):379–400.

Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. 2018. Sigsoftmax: Reanalysis of the softmax bottleneck. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. 2020. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*.

Henry A. Kautz, Ashish Sabharwal, and Bart Selman. 2009. Incomplete algorithms. In *Handbook of Satisfiability*.

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.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv*, abs/2107.13586.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

Toan Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 334–343, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Vikas Raunak, Siddharth Dalmia, Vivek Gupta, and Florian Metze. 2020. On long-tailed phenomena in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3088–3095, Online. Association for Computational Linguistics.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark. 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.

Warren D. Smith. 2014. D-dimensional orderings and stirling numbers. [Online; accessed 05-November-2021].

Richard P. Stanley. 2004. An introduction to hyperplane arrangements. In *Lecture notes, IAS/Park City Mathematics Institute*.

Jörg Tiedemann. 2020. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.

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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

A Halfspace interpretation

As promised, here is the derivation showing that if $P(c_i \mid \mathbf{x}) < P(c_j \mid \mathbf{x})$ then $\mathbf{x}$ is constrained to a halfspace.

We have:

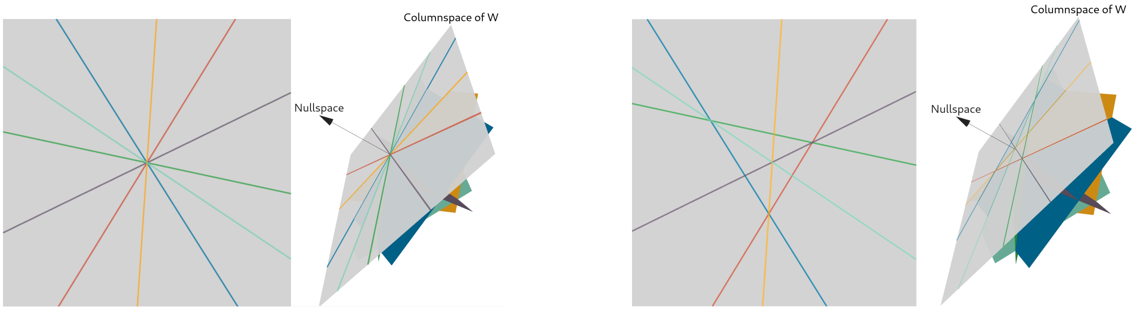
$$\begin{aligned}
P(c_i \mid \mathbf{x}) < P(c_j \mid \mathbf{x}) &\iff \\
\frac{e^{\mathbf{w}_{c_i}^\top \mathbf{x} + b_{c_i}}}{\sum_{i'} e^{\mathbf{w}_{c_{i'}}^\top \mathbf{x} + b_{c_{i'}}}} < \frac{e^{\mathbf{w}_{c_j}^\top \mathbf{x} + b_{c_j}}}{\sum_{i'} e^{\mathbf{w}_{c_{i'}}^\top \mathbf{x} + b_{c_{i'}}}} &\iff \\
e^{\mathbf{w}_{c_i}^\top \mathbf{x} + b_{c_i}} < e^{\mathbf{w}_{c_j}^\top \mathbf{x} + b_{c_j}} &\iff \\
\frac{e^{\mathbf{w}_{c_i}^\top \mathbf{x} + b_{c_i}}}{e^{\mathbf{w}_{c_j}^\top \mathbf{x} + b_{c_j}}} < 1 &\iff \\
e^{(\mathbf{w}_{c_i} - \mathbf{w}_{c_j})^\top \mathbf{x} + (b_{c_i} - b_{c_j})} < e^0 &\iff \\
(\mathbf{w}_{c_i} - \mathbf{w}_{c_j})^\top \mathbf{x} + (b_{c_i} - b_{c_j}) < 0 &
\end{aligned}$$

$\mathbf{x}$ is therefore constrained to a halfspace defined by normal vector $\mathbf{w}_{c_i} - \mathbf{w}_{c_j}$ and offset by $b_{c_i} - b_{c_j}$. This linear form defined by the normal vector and offset is the "shadow" in the input dimension of our friend, the Braid Arrangement, as we will make clear in the next section (see Derivation 2).

B Hyperplane Arrangements

Excellent resources to learn more about hyperplane arrangements are Stanley (2004) and Federico Ardila's lectures on polytopes (see Lecture 34 onwards). We give a brief introduction below.

A *hyperplane* in a vector space $\mathbb{R}^d$ is an affine subspace of dimension $d-1$. The hyperplane $\mathcal{H}$



(a) Input space $\mathbf{b} = \mathbf{0}$ (b) Output space $\mathbf{b} = \mathbf{0}$ (c) Input space $\mathbf{b} \neq \mathbf{0}$ (d) Output space $\mathbf{b} \neq \mathbf{0}$

Figure 7: Effect of bias term $\mathbf{b}$ on feasible permutations of $\mathrm{softmax}(\mathbf{Wx} + \mathbf{b})$, $\mathbf{W} \in \mathbb{R}^{|C| \times d}$, $d = 2$, $|C| = 4$. Having a bias term offsets the grey plane and allows it to not pass through the origin. This increases the number of regions by creating bounded regions seen in subfigures c and d. Each region intersected by the grey 2D plane corresponds to a feasible permutation. We therefore obtain $18/24$ feasible permutations if we include a bias term, compared to $12/24$ without one.

has one degree of freedom removed by specifying a constraint: a normal vector $\mathbf{w} \in \mathbb{R}^d$ to which it is perpendicular. The hyperplane may also be offset by b in that direction $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} = b\}$.

A *real hyperplane arrangement* $\mathcal{A}$ is defined as a set of n hyperplanes in $\mathbb{R}^d$, $\mathcal{A} = \{\mathcal{H}_1, \mathcal{H}_2 \dots \mathcal{H}_n\}$. The *regions* $\mathcal{R}$ defined by a hyperplane arrangement $\mathcal{A}$ are the connected components X of Euclidean space $\mathbb{R}^d$ left when we remove the hyperplanes $\mathcal{A}$, namely $X = \mathbb{R}^d - \bigcup_{\mathcal{H} \in \mathcal{A}} \mathcal{H}$. As an example, subfigure (a) in Figure 7 has 12 regions while subfigure (c) has 18 regions.

B.1 Braid Arrangement

The Braid Arrangement $\mathcal{B}_n$ is a hyperplane arrangement that partitions space into $n!$ regions corresponding to permutations. It can be constructed in $\mathbb{R}^n$ from the standard basis, the columns of the identity matrix $(\mathbf{e}_1, \mathbf{e}_2 \dots \mathbf{e}_n)$, $\mathbf{e}_i \in \mathbb{R}^n$, by taking all $\binom{n}{2}$ pairs of differences between them, each difference defining the normal vector of a hyperplane $\mathcal{H}_{i,j}$ of the Braid Arrangement.

$$\mathcal{B}_n = \{\mathcal{H}_{i,j} \quad \forall i, j : 1 \leq i < j \leq n\},$$

$$\mathcal{H}_{i,j} = \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{e}_i - \mathbf{e}_j)^\top \mathbf{x} = 0\}$$

The Braid Arrangement for $n = 3$ and $n = 4$ can be seen in Figure 3. It has $\binom{n}{2}$ hyperplanes, one per pair of dimensions in $\mathbb{R}^n$. Hence there are 3 hyperplanes for $|C| = 3$ and 6 hyperplanes for $|C| = 4$. As an example, when we have 4 classes the normal vector for $\mathcal{H}_{1,3}$ is $\mathbf{w}_{1,3} = \begin{bmatrix} 1 & 0 & -1 & 0 \end{bmatrix}^\top$. As can be verified by taking the dot product $\mathbf{w}_{i,j}^\top \mathbf{x}$, the result is positive if $x_i > x_j$ and negative if vice versa. Therefore, each hyperplane bisects space into two regions one for each possible ranking of the pair of coordinates.

To see how the hyperplanes intersect to give us a region $\mathcal{R}_{\boldsymbol{\pi}}$, we express a permutation (total order) over $|C|$ classes, such as that in Relation 3.1.2, using a chain of $|C| - 1$ pairwise inequalities.

$$P(c_{\pi_i} \mid \mathbf{x}) < P(c_{\pi_{i+1}} \mid \mathbf{x}), \quad 1 \leq i \leq |C| - 1$$

Each above constraint is equivalent to choosing a side of a braid hyperplane. By imposing all constraints, we obtain a region $\mathcal{R}_{\boldsymbol{\pi}}$ as the intersection of $|C| - 1$ halfspaces. There is therefore bijection between permutations and regions of the Braid Arrangement $\boldsymbol{\pi} \leftrightarrow \mathcal{R}_{\boldsymbol{\pi}}$.

B.2 Restricting the Braid Arrangement to lower dimensions

In the softmax classification layer of a neural network we often compute the output space activations $\mathbf{y} \in \mathbb{R}^n$ by applying a final affine layer to the softmax input space $\mathbf{x} \in \mathbb{R}^d$.

$$\mathbf{y} = \mathbf{Wx} + \mathbf{b}, \quad \mathbf{W} \in \mathbb{R}^{n \times d}, \mathbf{b} \in \mathbb{R}^n$$

What do the Braid Arrangement hyperplanes look like in the input dimension d? Let us start from the output space $\mathbb{R}^n$ and work backwards towards the input space $\mathbb{R}^d$.

$$\begin{aligned} y_i < y_j \implies & (\mathbf{e}_i - \mathbf{e}_j)^\top \mathbf{y} < 0 \\ & \mathbf{e}_i^\top \mathbf{y} - \mathbf{e}_j^\top \mathbf{y} < 0 \\ & \mathbf{e}_i^\top (\mathbf{Wx} + \mathbf{b}) - \mathbf{e}_j^\top (\mathbf{Wx} + \mathbf{b}) < 0 \\ & \mathbf{w}_i^\top \mathbf{x} + b_i - \mathbf{w}_j^\top \mathbf{x} - b_j < 0 \\ & (\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} + (b_i - b_j) < 0 \end{aligned} \tag{2}$$

We therefore see that if $d < n$ we can think of how the Braid Arrangement classifies outputs into permutations from two equivalent perspectives:

- In the output space $\mathbb{R}^n$ not all $\mathbf{y}$ are feasible, we can only classify an input $\mathbf{x}$ as a permutation $\boldsymbol{\pi}$ if the affine layer can map $\mathbf{x}$ to $\mathcal{R}_{\boldsymbol{\pi}}$. This can be seen in subfigures b and d of Figure 7 where the feasible outputs are a plane that intersects the Braid Arrangement.

- In the input space $\mathbb{R}^d$ all $\mathbf{x}$ are feasible but we only see the projection of the Braid Arrangement in this lower dimension. This can be seen in subfigures a and c of Figure 7.

The above gives us a recipe for building the Braid Arrangement in the input space when the outputs are an affine function of the inputs. This construction is illustrated in Figure 8, albeit without the bias term.

C Number of Regions (Feasible Permutations) of the restricted Braid Arrangement

The number of feasible permutations is invariant to specific choices of $\mathbf{W}$ and $\mathbf{b}$ (Cover, 1967; Smith, 2014) and only depends on the dimensionality of the softmax inputs d, the number of classes $|C|$ and whether we specify a bias term $\mathbf{b}$ not in the columnspace of $\mathbf{W}$. Namely, the cardinality of the set of feasible permutations does not change, but the members of the set do - they depend on the specific values in $\mathbf{W}$ and $\mathbf{b}$. There exists a recurrence formula to obtain the number of feasible permutations for a particular $|C|$ and d (Good and Tideman, 1977; Kamiya and Takemura, 2005). See our code and the relations in (Smith, 2014) for more details.

C.1 Softmax with no bias term

The number of feasible permutations as a function of $|C|$ and d when we have a softmax with no bias term can be seen in Table 2. When $d \geq |C| - 1$ all permutations corresponding to ways of ranking $|C|$ classes are feasible (table cells with $d = |C| - 1$ are highlighted in bold). However, as we make the Softmax Bottleneck narrower, we can represent less permutations, as can be seen from the numbers reported below the diagonal.

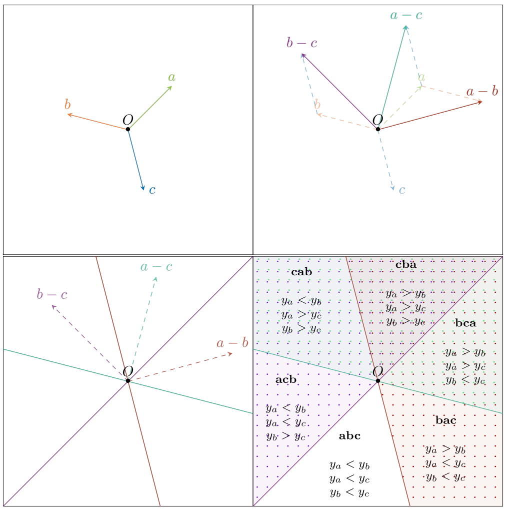


Figure 8: Constructing the Braid Arrangement in the input space for $|C| = 3$ classes and $d = 2$. *Top left*: The softmax weights $\mathbf{W} \in \mathbb{R}^{|C| \times d}$ for 3 classes, a, b, c. Each vector is a row of the weight matrix. *Top right*: We form the normal vectors for the braid hyperplanes by taking all pairs of differences between the basis vectors. *Bottom left*: The Braid hyperplanes are perpendicular to the normal vectors. Each hyperplane bisects space into two regions, one comprises the set of $\mathbf{x}$ for which class i has a larger activation that class j and the second vice versa. *Bottom right*: The hyperplanes partition space into $3! = 6$ regions corresponding to permutations. Each permutation contains the indices that sort the activations over classes in increasing order. Softmax decision boundaries are unions of two regions, e.g. regions **cba** and **bca** for class **a**.

C.2 Softmax with bias term

The number of feasible permutations as a function of $|C|$ and d when we have a softmax with a bias term is larger as can be seen in Table 3. As we saw in Figure 7, this is because a bias term can offset the representible linear subspace to an affine subspace which can intersect more regions of the Braid Arrangement.

		BOTTLENECK DIMENSIONALITY d									
		1	2	3	4	5	6	7	8	9	10
NUMBER CLASSES $\|C\|$	2	**2**	2	2	2	2	2	2	2	2	2
	3	2	**6**	6	6	6	6	6	6	6	6
	4	2	*12*	**24**	24	24	24	24	24	24	24
	5	2	20	72	**120**	120	120	120	120	120	120
	6	2	30	172	480	**720**	720	720	720	720	720
	7	2	42	352	1512	3600	**5040**	5040	5040	5040	5040
	8	2	56	646	3976	14184	30240	**40320**	40320	40320	40320
	9	2	72	1094	9144	45992	143712	282240	**362880**	362880	362880
	10	2	90	1742	18990	128288	557640	1575648	2903040	**3628800**	3628800

Table 2: Number of permutation regions defined by a bottlenecked softmax layer $Softmax(\mathbf{W}x)$ with no bias term. When $d \geq |C| - 1$ all permutations corresponding to ways of ranking $|C|$ classes are feasible. 12 in italics corresponds to the number of regions shown in the left subfigure of Figure 7. https://oeis.org/A071223.

		BOTTLENECK DIMENSIONALITY d									
		1	2	3	4	5	6	7	8	9	10
NUMBER CLASSES $\|C\|$	2	**2**	2	2	2	2	2	2	2	2	2
	3	4	**6**	6	6	6	6	6	6	6	6
	4	7	*18*	**24**	24	24	24	24	24	24	24
	5	11	46	96	**120**	120	120	120	120	120	120
	6	16	101	326	600	**720**	720	720	720	720	720
	7	22	197	932	2556	4320	**5040**	5040	5040	5040	5040
	8	29	351	2311	9080	22212	35280	**40320**	40320	40320	40320
	9	37	583	5119	27568	94852	212976	322560	**362880**	362880	362880
	10	46	916	10366	73639	342964	1066644	2239344	3265920	**3628800**	3628800

Table 3: Number of permutation regions defined by a bottlenecked softmax layer $Softmax(\mathbf{W}x + \mathbf{b})$. When $d \geq |C| - 1$ all permutations corresponding to ways of ranking $|C|$ classes are feasible. 18 in italics corresponds to the number of regions shown in the right subfigure of Figure 7.

D Preventing Stolen Probability

While Stolen Probability does not seem to occur in practice for MT models, there are solutions if we want to guarantee it cannot occur: we can force the decision boundaries of the softmax layer to create a Voronoi Tesselation in the output space.

A Voronoi Tesselation formed for a set of $|C|$ generating points partitions space into $|C|$ convex regions, one per point. A region created from a generating point contains any point in space for which the Euclidean distance to the generating point is smaller than the distance to all other generating points. Since all generating points form a region, this guarantees that all classes can be predicted.

The decision boundaries of softmax always form a Voronoi Tesselation when $\mathbf{W}$ is full rank (Hess et al., 2020, Theorem 1), as was shown by identifying the centroids of the tesselation as the softmax weights offset by a vector $\mathbf{u}$. When $\mathbf{W}$ is low rank, however, we will need to constrain the parameters of softmax in order to obtain a Voronoi Tesselation.

A summary of constraints we will show are needed are:

- If Softmax has no bias term, normalise the weight vectors [12]

$$\|\mathbf{w}_i\|_2 = const \quad 1 \leq i \leq C$$

- If Softmax has a bias term, then set the bias terms to:

$$b_i = -\frac{\|\mathbf{w}_i\|_2^2}{2} \quad 1 \leq i \leq C$$

Derivation: For any pair (i, j) of classes, we can construct an auxiliary pairwise classifier using the corresponding softmax weights $(\mathbf{w}_i, \mathbf{w}_j)$. This auxiliary binary classifier $\mathbf{w}' = \mathbf{w}_i - \mathbf{w}_j$ decides for each input $\mathbf{x}$ whether the activation for class i is greater than that for class j or not.

The decision boundary for this binary classifier will be the points $\mathbf{x}$ for which:

$$(\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} = 0$$

The region of $\mathbf{x}$ for which a multi-class classifier assigns class i as the argmax (a cone if no bias term) can be found using a combination of such binary classifiers:

For class i to be the argmax, require:

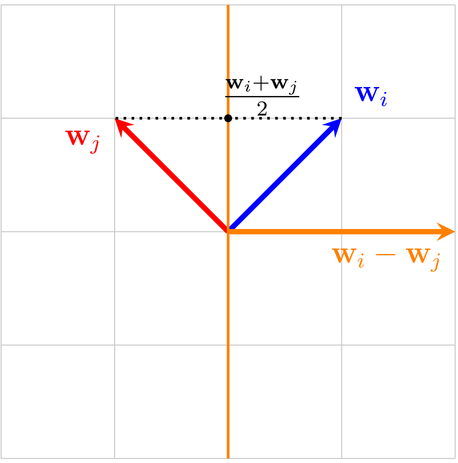

$\frac{\mathbf{w}_i+\mathbf{w}_j}{2}$
$\mathbf{w}_i$
$\mathbf{w}_j$
$\mathbf{w}_i - \mathbf{w}_j$

Figure 9: We can be certain that all classes are representable if we force each decision boundary formed by a braid vector to also be a perpendicular bisector of the line segment joining the two classes.

$$(\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} > 0$$
$$(\mathbf{w}_i - \mathbf{w}_k)^\top \mathbf{x} > 0$$
$$\dots$$
$$(\mathbf{w}_i - \mathbf{w}_n)^\top \mathbf{x} > 0$$

To obtain a Voronoi Tesselation, we want all the auxiliary classifier decision boundaries to be the perpendicular bisectors of the line segments joining their corresponding weight vectors $\mathbf{w}_i$ and $\mathbf{w}_j$. If we require this for all possible pairs of classes, the hyperplanes that define the boundaries of the softmax regions will also satisfy this property and classification will be equivalent to assigning an input to the class weight having the smallest Euclidean distance with it.

Therefore, let us force a point $\mathbf{x}$ of the decision boundary to be a point on the perpendicular bisector of the line segments connecting all pairs of class weights and see where this leads us. In the plot below, the orange line is both the perpendicular bisector of the line segment having $\mathbf{w}_i$ and $\mathbf{w}_j$ as endpoints, as well as the decision boundary for the auxiliary classifier since $\mathbf{w}' = \mathbf{w}_i - \mathbf{w}_j$ is the normal vector.

D.1 Softmax without a bias term

Let us start with the scenario where we have a softmax layer with no bias vector. We pick $\mathbf{x}$ to be the midpoint of the line segment joining $\mathbf{w}_i$ and $\mathbf{w}_j$. This way $\mathbf{x}$ will be both on the decision boundary as well as on the perpendicular bisector.

[12] This is also clear from Hess et al. (2020, Theorem 1)

$$
\begin{aligned}
(\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} = 0 &\implies \\
(\mathbf{w}_i - \mathbf{w}_j)^\top \left(\frac{\mathbf{w_i} + \mathbf{w_j}}{2}\right) = 0 &\implies \\
\frac{\mathbf{w}_i^\top \mathbf{w}_i + \mathbf{w}_i^\top \mathbf{w}_j - \mathbf{w}_j^\top \mathbf{w}_i - \mathbf{w}_j^\top \mathbf{w}_j}{2} = 0 &\implies \\
\frac{\|\mathbf{w}_i\|_2^2 - \|\mathbf{w}_j\|_2^2}{2} = 0 &\implies \\
\|\mathbf{w}_i\|_2 = \|\mathbf{w}_j\|_2 &
\end{aligned}
$$

The above would need to be true for all pairs (i, j). Therefore, we see that if the weight vectors for all classes are equal (or normalised to 1) we get a Voronoi tesselation with the weight vectors as centroids. Such a normalisation step has been carried out in the literature (Nguyen and Chiang, 2018; Raunak et al., 2020), but not justified from this point of view, to our knowledge.

D.2 Softmax with a bias term

Alternatively, if we had a softmax layer with a bias term, we would have the following changes:

The auxiliary pairwise classifier decision boundaries would be:

$$(\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} + b_i - b_j = 0$$

Following the same steps as above we would arrive at:

$$
\begin{aligned}
(\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} + b_i - b_j = 0 &\implies \\
b_i - b_j = \frac{\|\mathbf{w}_j\|_2^2 - \|\mathbf{w}_i\|_2^2}{2} &
\end{aligned}
$$

We can obtain the above condition for all auxiliary classifier pairs by setting the bias term of each weight vector depending on the norm of the weight vector:

$$b_i = -\frac{\|\mathbf{w}_i\|_2^2}{2}$$

E Braid Reflect Approximate Algorithm

```
Algorithm 2: Braid reflect
Data: Class index $c_t$,
      $\mathbf{W} \in \mathbb{R}^{|C| \times d}$, $\mathbf{b} \in \mathbb{R}^{|C|}$
Result: Whether $c_t$ is bounded
bounded = true
patience = 2500
$\mathbf{x} = \mathbf{W}_{\mathbf{c_t}:}^\top$
while patience do
    $c_i$ = argmax($\mathbf{W}\mathbf{x} + \mathbf{b}$)
    if $c_i = c_t$ then
        bounded = false
        break
    else
        $\mathbf{w} = (\mathbf{W}_{c_t:} - \mathbf{W}_{c_i:})^\top$
        $b = \mathbf{b}_{c_t} - \mathbf{b}_{c_i}$
        $\mathbf{w}' = \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$
        $x_p = \mathbf{w}'^\top \mathbf{x}$
        $\mathbf{x} = \mathbf{x} - 2(x_p + \frac{b}{\|\mathbf{w}\|_2})\mathbf{w}'$
        patience = patience - 1
    end
end
```

Figure 10: Approximate algorithm to detect whether class c_t has Stolen Probability.

F Stolen probability search results

model	# approx bounded	# bounded
bert-base-cased	0	0
bert-base-uncased	0	0
roberta-base	0	0
roberta-large	0	0
xlm-roberta-base	0	0
xlm-roberta-large	0	0
gpt2	0	0

Table 4: Stolen Probability search results for Language Models. **approx bounded** is the number of tokens that the approximate algorithm failed to prove were unbounded. **bounded** is the number of bounded tokens according to the exact algorithm. No tokens were found to be bounded.

source	model	# approx bounded	# bounded
Helsinki NLP	opus-mt-ja-en	109	2
	opus-mt-ru-en	90	159
	opus-mt-bg-en	93	53
	opus-mt-ja-en(2)	14	0
	opus-mt-ar-en	40	184
	opus-mt-en-el	75	42
	opus-mt-de-el	115	6
	opus-mt-ar-el	41	0
	opus-mt-es-el	67	32
	opus-mt-fi-el	57	8
	opus-mt-ar-he	3	0
	opus-mt-de-he	4	0
	opus-mt-es-he	3	0
	opus-mt-fr-he	1	0
	opus-mt-fi-he	7	0
	opus-mt-ja-he	0	0
	opus-mt-en-ar	21	2
	opus-mt-el-ar	12	0
	opus-mt-es-ar	17	1
	opus-mt-fr-ar	17	0
	opus-mt-he-ar	7	0
	opus-mt-it-ar	8	0
	opus-mt-ja-ar	4	0
	opus-mt-pl-ar	52	0
	opus-mt-ru-ar	8	0
	opus-mt-en-ru	98	34
	opus-mt-es-ru	42	18
	opus-mt-fi-ru	1	0
	opus-mt-fr-ru	34	43
	opus-mt-he-ru	5	0
	opus-mt-ja-ru	13	0
	opus-mt-ko-ru	2	0

Table 5: Stolen Probability search results for Helsinki NLP OPUS models. **approx bounded** is the number of tokens that the approximate algorithm failed to prove were unbounded. **bounded** is the number of bounded tokens according to the exact algorithm. For 13/32 models some infrequent tokens were found to be bounded.

source	model	# approx bounded	# bounded
FAIR	facebook/wmt19-en-ru	5	0
	facebook/wmt19-ru-en	64	0
	facebook/wmt19-de-en	173	0
	facebook/wmt19-en-de	184	0

Table 6: Stolen Probability search results for FAIR WMT'19 models. **approx bounded** is the number of tokens that the approximate algorithm failed to prove were unbounded. **bounded** is the number of bounded tokens according to the exact algorithm. No tokens were found to be bounded.

source	model	# approx bounded	# bounded
Bergamot	cs-en.student.base	0	0
	es-en.teacher.bigx2(1)	0	0
	es-en.teacher.bigx2(2)	0	0
	en-es.teacher.bigx2(1)	0	0
	en-es.teacher.bigx2(2)	0	0
	et-en.teacher.bigx2(1)	2	0
	et-en.teacher.bigx2(2)	1	0
	en-et.teacher.bigx2(1)	1	0
	en-et.teacher.bigx2(2)	1	0
	nb-en.teacher.base	0	0
	nn-en.teacher.base	0	0
	is-en.teacher.base	0	0
	cs-en.student.base	0	0
	cs-en.student.tiny11	0	0
	en-cs.student.base	0	0
	en-cs.student.tiny11	0	0
	en-de.student.base	0	0
	en-de.student.tiny11	0	0
	es-en.student.tiny11	0	0
	en-es.student.tiny11	0	0
	et-en.student.tiny11	0	0
	en-et.student.tiny11	0	0
	is-en.student.tiny11	0	0
	nb-en.student.tiny11	0	0
	nn-en.student.tiny11	0	0

Table 7: Stolen Probability search results for Bergamot models. **approx bounded** is the number of tokens that the approximate algorithm failed to prove were unbounded. **bounded** is the number of bounded tokens according to the exact algorithm. No tokens were found to be bounded. Interestingly, student models were much easier to prove unbounded than teacher models, despite student model softmax weights being lower dimensional.

source	model	# approx bounded	# bounded
WMT'17 Edinburgh	en-cs.l2r(1-4)	≤ 2	0
	en-cs.r2l(1-4)	≤ 1	0
	cs-en.l2r(1-4)	≤ 2	0
	cs-en.r2l(1-4)	0	0
	en-de.l2r(1-4)	≤ 1	0
	en-de.r2l(1-4)	≤ 2	0
	de-en.l2r(1-4)	≤ 2	0
	de-en.r2l(1-4)	0	0
	en-ru.l2r(1-4)	0	0
	ru-en.l2r(1-4)	0	0
	ru-en.r2l(1-4)	0	0
	en-tr.l2r(1-4)	≤ 5	0
	en-tr.r2l(1-4)	≤ 4	0
	lv-en.l2r(1-4)	0	0
	lv-en.r2l(1-4)	≤ 1	0
	tr-en.l2r(1)	2	0
	tr-en.l2r(2)	8	0
	tr-en.l2r(3)	6	0
	tr-en.l2r(4)	2	0
	tr-en.r2l(1)	4	0
	tr-en.r2l(2)	0	0
	tr-en.r2l(3)	6	0
	tr-en.r2l(4)	4	0
	en-zh.l2r(1)	3	0
	en-zh.l2r(2)	3	0
	en-zh.l2r(3)	14	0
	en-zh.l2r(4)	1	0
	en-zh.r2l(1)	2	0
	en-zh.r2l(2)	0	0
	en-zh.r2l(3)	7	0
	en-zh.r2l(4)	7	0
	zh-en.l2r(1)	8	0
	zh-en.l2r(2)	3	0
	zh-en.l2r(3)	366	0
	zh-en.r2l(1-3)	≤ 3	0

Table 8: Stolen Probability search results for Edinburgh WMT'17 submission (ensemble) models. **approx bounded** is the number of tokens that the approximate algorithm failed to prove were unbounded. **bounded** is the number of bounded tokens according to the exact algorithm. **r2l** and **l2r** refer to training direction, with **l2r** denoting training left to right and **r2l** right to left. Models submitted were ensembles, hence there are more than one model per language pair and direction. When all models per language pair and direction had less than 5 counts, we summarise all models with a single row, e.g. (1-4).

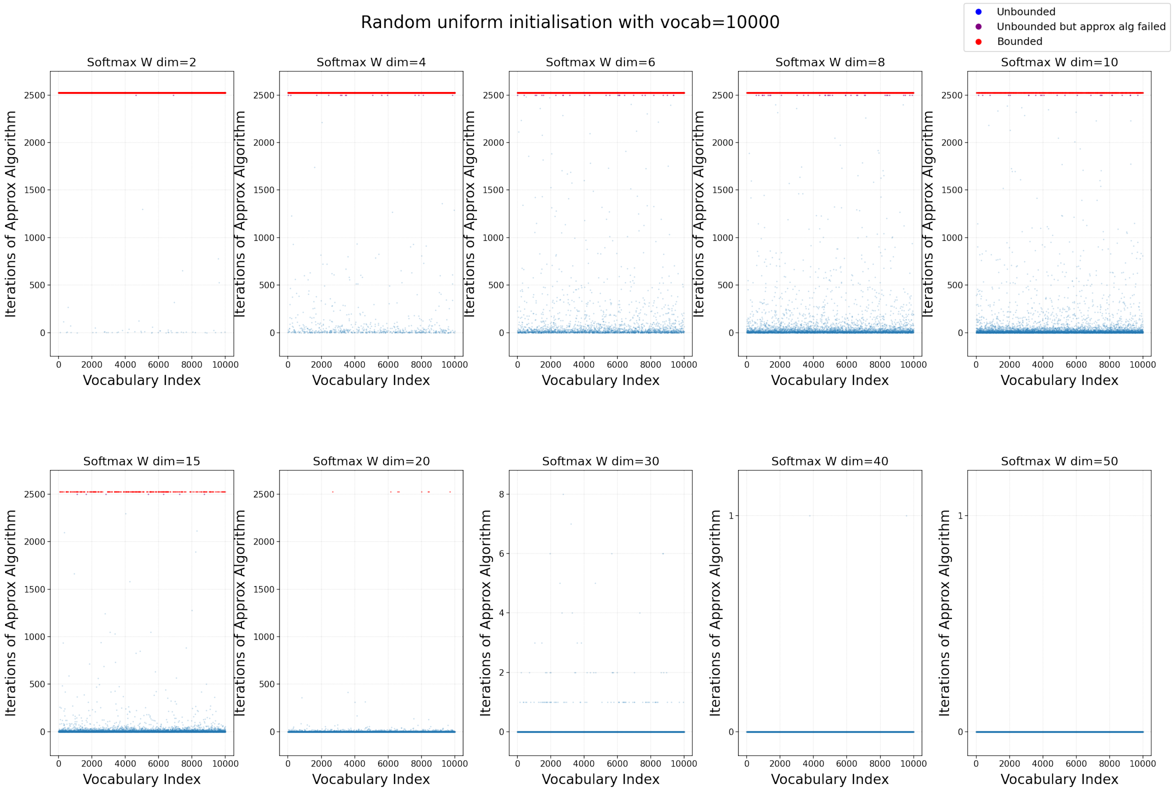


Figure 11: Illustration of softmax weight dimensionality affecting stolen probability when weights are randomly initialised for a vocabulary of 10000. The softmax weights and bias term are initialised using a uniform $U(-1,1)$ distribution. Stolen probability is unlikely to occur as we increase the dimensionality of the weight vectors. This can be seen in the subplots from top-left to bottom-right as we increase the dimensionality. Moreover, the braid reflect approximate algorithm fails less and needs less iterations to find an input that proves a token is unbounded. For example, for the bottom right two figures most tokens are shown to be unbounded with 1 or 0 iterations.

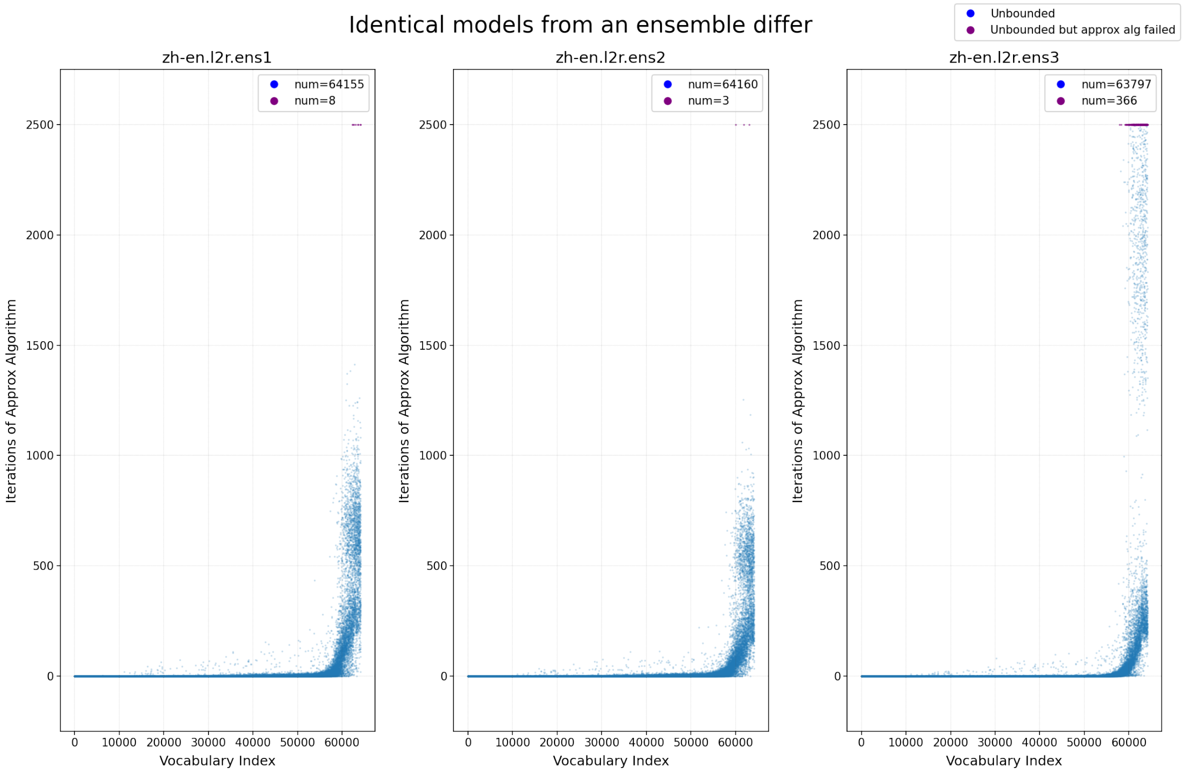
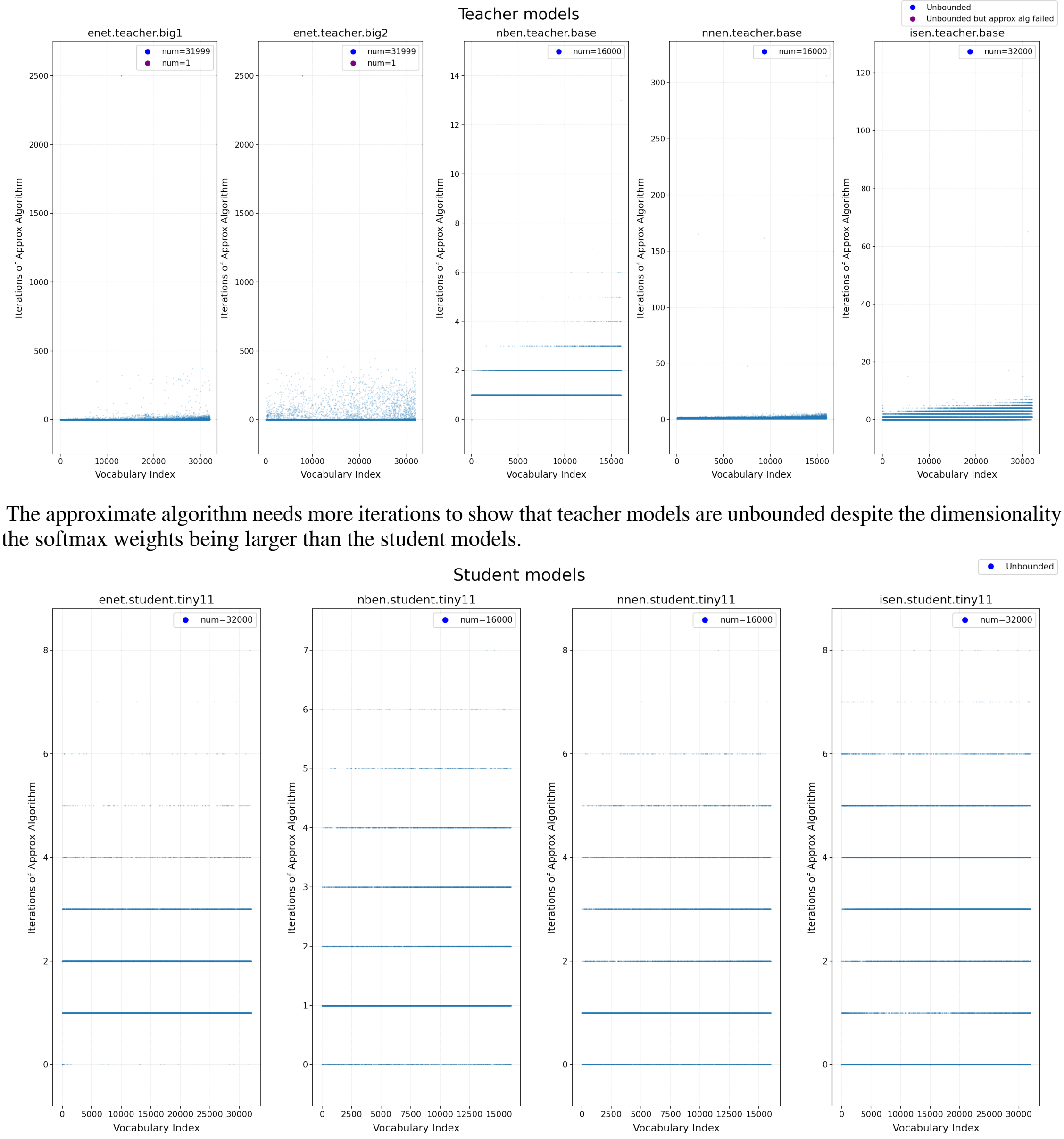


Figure 12: Models from an ensemble can differ a lot in how easy they are to scan for stolen probability despite their difference being solely the random seed used in initialisation. As can be seen, the right-most figure has 366 vocabulary tokens that are unbounded but the approximate algorithm fails to find a solution, compared to 8 and 3 for the other two models.



(a) The approximate algorithm needs more iterations to show that teacher models are unbounded despite the dimensionality of the softmax weights being larger than the student models.

(b) Student models are very easy to show to be unbounded.

Figure 13: Number of iterations of the approximate algorithm needed to show that a vocabulary token is unbounded.

G Activation range of softmax layer inputs

Neural network activations are bounded in magnitude in practice, since larger activations can lead to larger gradients and instability during training. In this work, we made the assumption that the softmax layer inputs $\mathbf{x}$ are bounded within a range for all dimensions: $-100 \leq \mathbf{x} \leq 100$. Below we provide some supporting empirical evidence that this assumption is reasonable.

We checked this assumption on 2 Helsinki NLP OPUS models for en-ru and bg-en, which were found to have tokens bounded in probability. We took 10 million sentence pairs from OPUS as released in Tiedemann (2020) for the corresponding language pairs and input them to the corresponding models, decoding using the gold translations. We then recorded the range of the minimum and maximum activation for the softmax layer inputs.

Since our assumption is that all 512 dimensions are bounded between -100 and 100, we focus on the range of the minimum and maximum activation for each output token across all dimensions. We therefore calculate a 99 percentile for the min and max activation per token across all dimensions as well as the overall min and max activations overall. The results can be seen in Table 9, from which we can see that for these two models our assumption holds for all activations produces for 10 million sentences and the percentiles show that more than 99% of the extreme values fall within the $[-50, 50]$ range.

model	min range	max range	min	max
bg-en	$[-37.5, -9.4]$	$[12.1, 40.3]$	-57.47	58.87
en-ru	$[-41.6, -9.9]$	$[10.9, 36.4]$	-95.4	94.4

Table 9: Range of activations for softmax inputs as calculated on 10 million sentence pairs from OPUS. Ranges are 99 percentiles and min and max are the largest activation across all dimensions for all sentences.