ICLR 2021 Spotlight

# PMI-MASKING: PRINCIPLED MASKING OF CORRELATED SPANS

**Yoav Levine*** **Barak Lenz** **Opher Lieber** **Omri Abend**

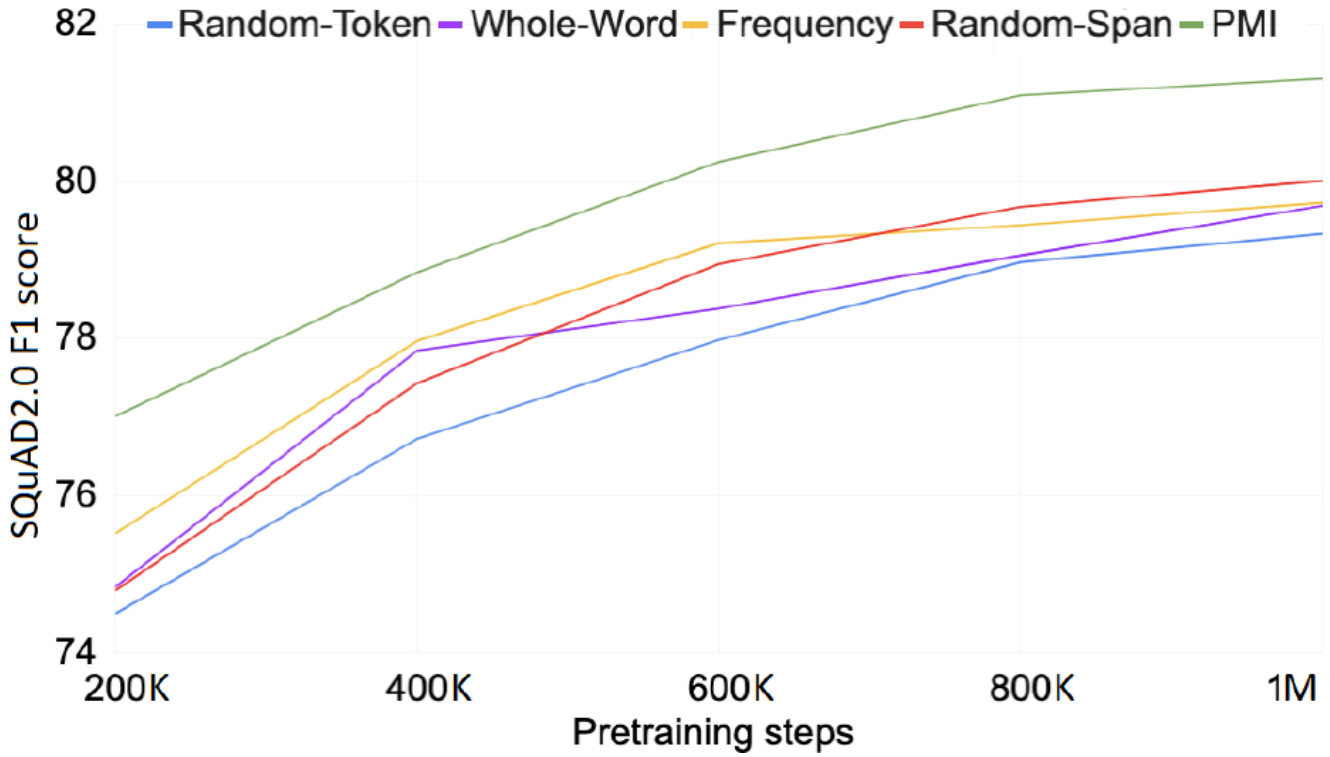
**Kevin Leyton-Brown** **Moshe Tennenholtz** **Yoav Shoham**

AI21 Labs, Tel Aviv, Israel

紹介者：笹野遼平（名大）

# 論文の概要

- Masked Language Modelの Masking Strategyについて
- PMIが大きい$n$-gramを まとめてmaskingする PMI-MASKINGを提案
- 既存手法と同等の精度 に半分の時間で到達、 最終的な精度も向上



# 背景

# 背景１: Masked Language Model (MLN)

- BERTに代表されるMasked Language Model (MLN)では与えられた文字列の一部のtokenをmaskし、それを予測することで学習
  - オリジナルのBERTでは無作為に選ばれた15%のsub-word tokenが対象
  - 80%は[mask]、10%は別token、10%は元のまま(以降もこの設定で実験)
- 既知の問題点
  - 繋がりの強いtokenがmaskされた場合、局所的な手掛かりを過度に利用 (cf. Whole-Word Masking)

    e.g., 以下では[mask]=“igen”を前後のtokenだけから予測している可能性

    *we use the eigenvector corresponding to its largest e-[mask]-val-ue*
  - sub-wordの場合に顕著だけどcollocationでも同様の問題があるはず

# 予備実験: sub-wordを独立にmaskする弊害の検証

- tokenごとに独立にmaskする手法(Random-Token Masking)とword単位でtokenをmaskする手法(Whole-Word Masking)を比較
- 語彙サイズを30K→10K→2Kと小さくしていき精度を比較
  - 語彙サイズが小さくなると語彙に含まれるwordが減少
    ⇒ sub-wordに分割されるwordが増加
    ⇒ sub-word予測に局所的な手掛かりが過度に利用されているならば
    Whole-Word Maskingの効果が顕著に現れるはず

| SQuAD2.0 development set F1 scores | 1.08 **tokens per word** (30K vocabulary) | 1.22 **tokens per word** (10K vocabulary) | 2.06 **tokens per word** (2K vocabulary) |
|---|---|---|---|
| Random-Token Masking | 79.3 | 77.8 | 72.8 |
| Whole-Word Masking | 79.7 | 79.5 | 77.6 |

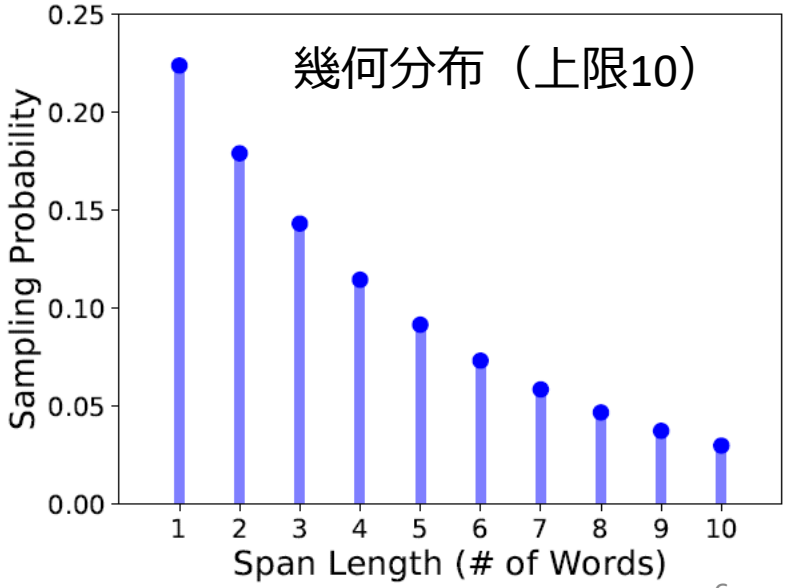
顕著な差

# 背景２：既存のMasking Strategy

- Random-Token Masking (Devling+’19a)
  - オリジナルのBERTの実装、独立に無作為にmaskするtokenを選定
- Whole-Word Masking (Develin+’19b)
  - sub-wordの予測が簡単すぎるのでword単位でまとめてmasking

inter-word

- Knowledge Masking (Sun+’19)
  - いわゆるBaidu版ERNIE (外部知識を利用)
  - Entity-levelおよびphrase-levelでmasking
- Random-Span Masking (Joshi+’20)
  - 直接の先行研究、Randomなspan単位でmasking
  1. 幾何分布（右図）に従いスパン長をサンプリング
  2. 開始位置を無作為に選定しスパン長だけmasking
  3. 全体の15%がmaskingされるまで繰り返し



# 提案手法: PMI-MASKING

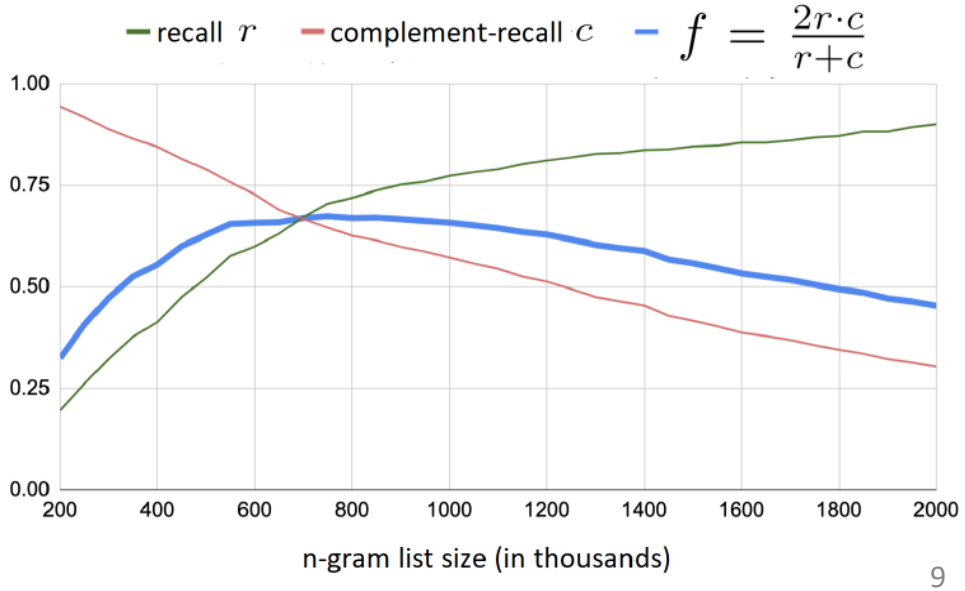
# 準備1: $n$-gramに対応したPMI

- 一般的なPointwise Mutual Information (PMI)はbigramに対し定義
  - $\mathrm{PMI}(w_1 \mathrm{w}_2) = \log \frac{p(\mathrm{w}_1 \mathrm{w}_2)}{p(w_1)p(w_2)}$
- これを単純に拡張すると下記になるが部分列のPMIが大きいと全体のPMIも大きくなってしまう（cf. “*Kuala Lumpur is*”）
  - $\mathrm{Naive-PMI}_n(w_1 \dots w_n) = \log \frac{p(w_1 \dots w_n)}{\Pi_{j=1}^{n} p(w_j)}$
- 全体で１つのまとまりとなる$n$-gramのPMIが大きくなるように定義
  - $\mathrm{PMI}_n(w_1 \dots w_n) = \min_{\sigma \in \mathrm{seg}(w_1 \dots w_n)} \log \frac{p(w_1 \dots w_n)}{\Pi_{s \in \sigma} p(s)}$

※ $\mathrm{seg}(w_1 \dots w_n)$は“$w_1 \dots w_n$”の連続する任意の文字列への分割の集合（分割なしは除く）

# 準備2: masking vocabularyの決定

- コーパスに10回以上出現した2-gram～5-gramが候補
- $\mathrm{PMI}_n$はtoken長の影響が大きいのでtoken長ごとに順位付けし、合わせて800Kの語彙サイズとなるように選択
  - 訓練用コーパスのおよそ半分のtokenがそれらの語彙に含まれることに
- 800Kという数字は予備実験で決定
  - 10 wordsを対象に長さ$n \in \{2,3,4,5\}$、それぞれ25個の$n$-gramをタグ付け
  - F値が高くなる語彙サイズに決定



# PMI-MASKING

- masking vocabularyに含まれる$n$-gramを１つのunitとして扱う
  - 入れ子があった場合は大きい方を採用
  - その以外で重複するentryがあった場合は無作為に片方を採用
  - それ以外のtokenは独立にunitとして扱う
- 入力されたtoken列をunitに分けた後はunit単位でmaskingを行う
  - BERTと同様に全体の15%がmaskingの対象
  - 80%は[mask]、10%は別token、10%は元のまま

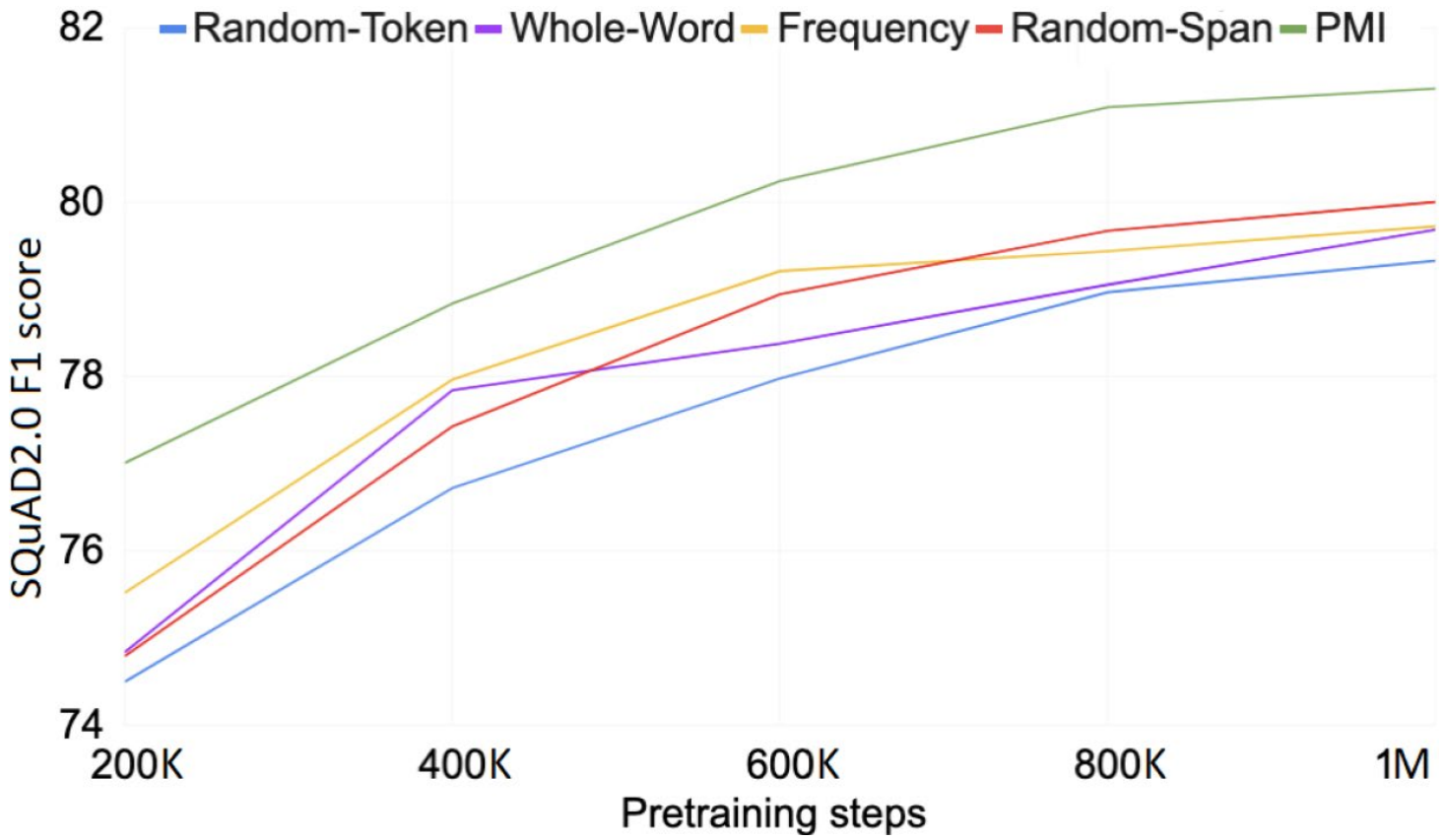
# 実験

# 評価対象

- SQuAD (Stanford Question Answering Dataset) 2.0
  - 質問応答のベンチマーク（正答がソース文に存在するとは限らない）
- RACE (ReAding Comprehension dataset from Examinations)
  - 中国の中学/高校生向けの英語の試験問題から作成されたデータセット
  - ４つの選択肢から１つの正しい答えを選択する形式
- GLUE (General Language Understanding Evaluation)
  - 複数のタスクで構成される言語理解ベンチマーク（8タスクで評価）
- Single-Token perplexity (右表)
  - １つのtokenだけmaskした場合のperplexity
  - 下流タスクの性能の目安にはならない

| | |
|---|---|
| Random-Token Masking | **2.96** |
| Random-Span Masking | 4.30 |
| Naive-PMI-Masking | 7.35 |
| PMI-Masking | 21.85 |

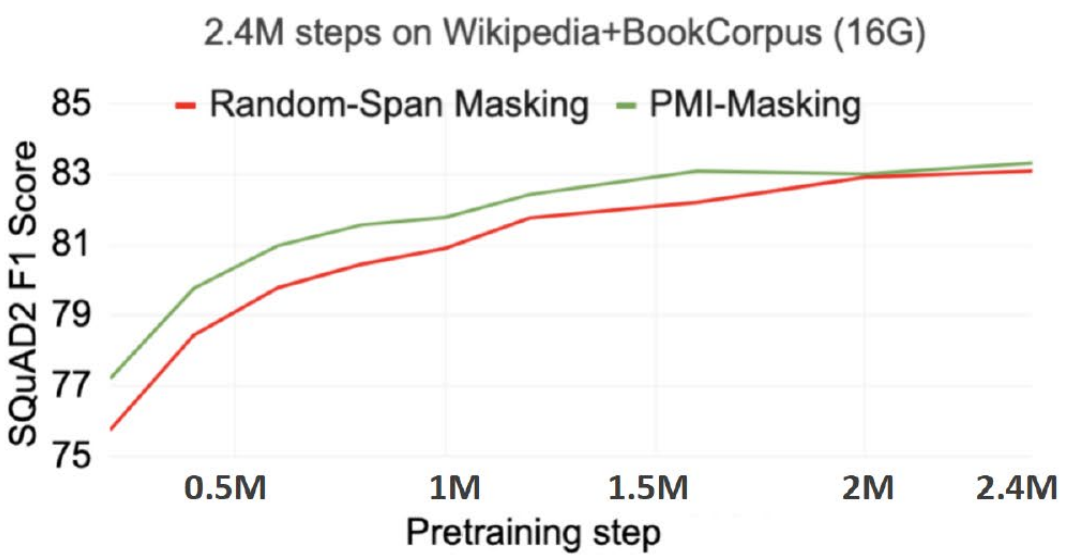
# 事前学習のStep数ごとの比較

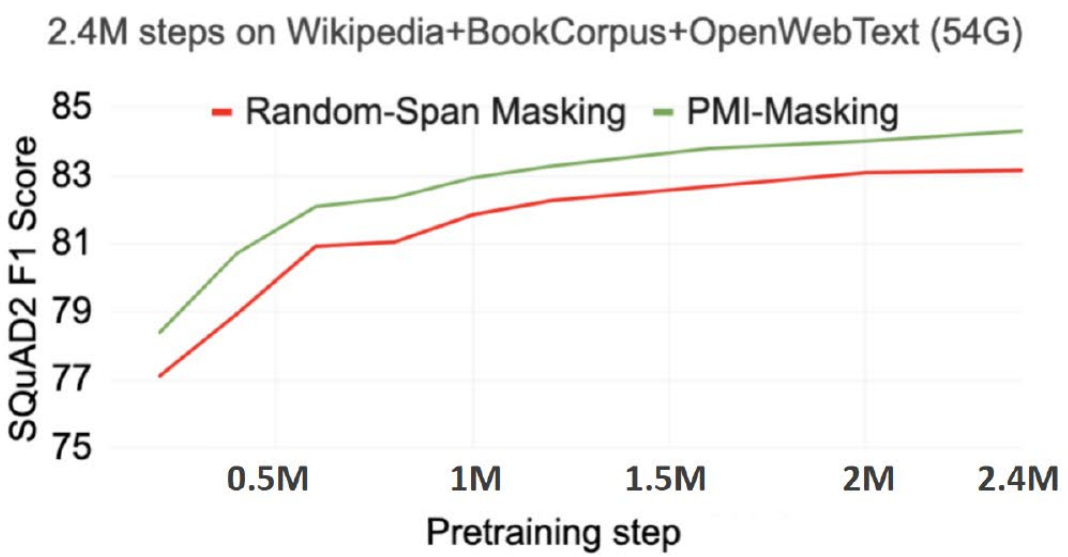



Wikipedia+BookCorpus dataset with batch size 256

# Step数をさらに増やした場合の比較

- Wikipedia+BookCorpus (16GB)で学習した場合、最終的な性能はRandom-Span Maskingと同じだったもののその性能に早く到達（下図左）
- Wikipedia+BookCorpus+OpenWebText (54GB)で学習した場合さらに性能向上



# 事前学習完了後の精度比較(dev/test)

| **BERT Base with different maskings** | **SQuAD2.0** EM | F1 | **RACE** Acc. | **GLUE** Avg |
|---|---|---|---|---|
| *1M training steps on* WIKIPEDIA+BOOKCORPUS*(16G):* | | | | |
| Random-Token Masking | 76.4/– | 79.6/– | 67.8/66.2 | 83.1/– |
| Random-Span Masking | 77.1/– | 80.3/– | 68.6/66.9 | 83/– |
| Naive-PMI-Masking | 78.2/– | **81.3**/– | 69.7/67.8 | **84.1**/– |
| PMI-Masking | **78.5**/– | **81.4**/– | **70.1/68.4** | **84.1**/– |
| *2.4M training steps on* WIKIPEDIA+BOOKCORPUS*(16G)* | | | | |
| Random-Span Masking | 79.7/80.0 | 82.7/82.8 | 71.9/69.5 | **84.8**/79.7 |
| Naive-PMI-Masking | **80.3**/80.2 | **83.2**/83.2 | 71.7/69.8 | 84.5/80.0 |
| PMI-Masking | **80.2/80.9** | **83.3/ 83.6** | **72.3/70.9** | 84.7/**80.3** |
| *2.4M training steps on* WIKIPEDIA+BOOKCORPUS+OPENWEBTEXT*(54G):* | | | | |
| Random-Span Masking | 80.1/80.4 | 83.2/83.3 | 74.0/72.2 | 85.1/80.1 |
| Naive-PMI-Masking | 80.4/80.0 | 83.3/83.0 | 73.9/71.4 | 85.6/80.3 |
| PMI-Masking | **80.9/82.0** | **83.9/84.9** | **74.8/73.2** | **86.0/80.8** |

# published Base-sized modelとの比較

| **PMI vs Prior BASE MLMs** | **Corpus size** | **Batch × Steps = Examples** | **RACE** dev/test |
|---|---|---|---|
| *PMI vs n-grams in vocabulary* | | | |
| AMBERT (Zhang & Li, 2020) | 47G | 1024 × 0.5M = 512G | 68.9†/66.8† |
| PMI-Masking | 16G | 256 × 1M = 256M | **70.1/68.4** |
| *PMI vs Random-Span Masking* | | | |
| SpanBERT$_{\text{BASE}}$ (Joshi et al., 2020) | 16G | 256 × 2.4M = 614.4M | 70.5/68.7 |
| PMI-Masking | 16G | 256 × 2.4M = 614.4M | **72.3/70.9** |
| *PMI vs Random-Token Masking with 3X more data and 6X more training examples* | | | |
| RoBERTa$_{\text{BASE}}$ (Liu et al., 2019) | 160G | 8K × 0.5M = 4G | **74.9**/73 |
| PMI-Masking | 54G | 256 × 2.4M = 614.4M | **74.8/73.2** |

# まとめと感想

- Masked Language Modelにおける新しいMasking Strategyとして、PMIに基づく手法（PMI-MASKING）を提案した論文
  - PMIが大きくなる$n$-gramをunitとしてまとめてmasking
- 下流タスクに対し既存手法と同等の精度に半分の時間で到達、最終的な性能も向上
  - 学習データが大きい場合に顕著
  - Perplexityは既存手法より大きくなる
- シンプルなアイデアをしっかり検証
  汎用性はかなり高そう

$$\mathrm{PMI}_n(w_1 \dots w_n) = \min_{\sigma \in \mathrm{seg}(w_1 \dots w_n)} \log \frac{p(w_1 \dots w_n)}{\Pi_{s \in \sigma} p(s)}$$