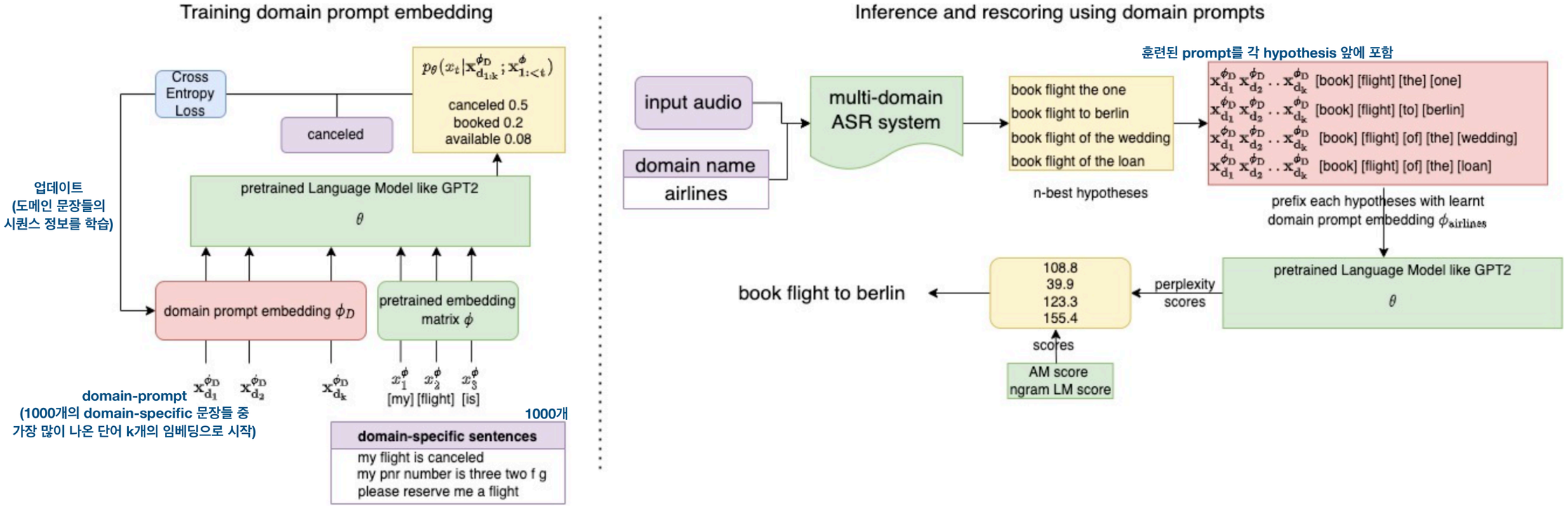
Language Modeling and Lexical Modeling for ASR

1. Domain Prompts: towards memory and compute efficient domain adaptation of ASR systems

- ABSTRACT
  - domain-prompts : 도메인에 특화된 임베딩을 토큰 임베딩과 함께 사용해 transformer-based LM의 성능을 높인다.
  - 메모리 효율적 : LM 파라미터 개수의 약 0.02%만 이용해 훈련됨
  - domain 연관 문장 1000개로 domain prompt embedding 학습 뒤, ASR scoring 시 7-13% WER 감소
  - fully fine-tuned 모델에 비해 유사하거나 더 나은 성능을 보임

- METHOLOGY
  - GPT3에서 번역할 문장 앞에 프롬프트 붙이는 방식(ex.이 문장 좀 번역해줘)과 유사하게 도메인 임베딩을 함께 input으로 활용

1. Domain Prompts: towards memory and compute efficient domain adaptation of ASR systems



1. Domain Prompts: towards memory and compute efficient domain adaptation of ASR systems

- SETUP
  - LM : gpt2, gpt2-medium / AM : 12000시간으로 훈련
  - GPU : Tesla V100 8장 (mint 등과 동일)
  - low-data setting (1k), large-data setting (50k) 구분해서 각각 훈련

1. Domain Prompts: towards memory and compute efficient domain adaptation of ASR systems

- RESULT

	domain adaptation methods	base model	# additional domain-specific params	Low data setting (1k sentences) WER Relative % ↑				Large data setting (50k sentences) WER Relative % ↑			
				airlines	fastfood	healthcare	insurance	airlines	fastfood	healthcare	insurance
	no rescoring	-	0	-	-	-	-	-	-	-	-
4	no adaptation	gpt2-medium	0	3.6	3.4	3.4	4.8	4.9	3.4	0.8	8.2
16	domain-prompts ($k = 200$, vocab init)	gpt2	0.16M	6.1	9.3	7.3	5.9	8.5	9.2	4.0	10.3
17	domain-prompts ($k = 50$, vocab init)	gpt2-medium	0.05M	**8.1**	**13.1**	**7.7**	**8.1**	**11.0**	11.1	5.7	**12.4**
18	full-fine-tuning	gpt2	117M	6.6	9.8	6.8	6.4	8.5	12.9	4.8	12.3
19	full-fine-tuning	gpt2-medium	345M	7.1	11.2	7.2	7.0	8.5	**16.2**	**7.2**	**12.4**

- WERR 기준, large-data setting에서의 두 도메인을 제외하고는 모두 가장 큰 WER 감소폭을 보임
- 즉 새로운(=데이터가 충분하지 않은) 도메인에서는 domain prompts를 활용하는 것이 좋은 성능을 보임

1. Domain Prompts: towards memory and compute efficient domain adaptation of ASR systems

- 감상
  - LM을 도메인별로 파인튜닝하는 방법에 비해 효율적인 것 같다.
  - WER 대신 WERR을 기재한 이유가 있을지 궁금하다.

2. A BERT-based Language Modeling Framework

- ABSTRACT
  - BERT를 활용한 독자적인 LM (BLM) 생성
  - 세 가지 데이터셋 기준으로 성능 향상을 보임
  - AM을 통과한 n-best hypothesis를 re-ranking하는 태스크에도 좋은 성능을 보임

- FRAMEWORK
  - 목표: 주어진 시퀀스를 보고 가장 유력한 다음 단어를 예측하는 LM
  - cross-layer BLM (xBLM), BERT-based cache models (cBLM, gcBLM), 여러 LM들을 혼합하는 방법론 (dynamic adaptation) 제공

2. A BERT-based Language Modeling Framework

- Cross-layer BLM (xBLM)
  - 목표: 주어진 시퀀스를 보고 가장 유력한 다음 단어를 예측하는 LM
  - 기존 BERT-based LM은 문장에 대한 정보들이 transformer layer별로 다르게 분포되어 있음 (보통 마지막 layer 사용)
  - Cross-layer BLM : 모든 층을 이용, 증류된 정보를 LM에 plug-in

$$P_{xBLM}(w_i|W_{<i}) = \text{softmax}\left(\text{FFN}(\tilde{h}_{xBLM})\right),$$
$$\tilde{h}_{xBLM} = \sum_{k=1}^{K} \alpha^k h_{[CLS]}^k,$$

weighted sum of all [CLS] representations

- BERT-based Cache models
  - 기존 cache model : '비교적 최근 사용된 단어는 또 나올 가능성이 높다'는 점을 반영한 확률 모델
  - cBLM : 단어 단계에서 확률을 계산하지 않고, BERT가 pre-trained 모델임을 활용해 미리 embedding matrix를 생성한 뒤 가중치를 계산
  - gcBLM : cache를 문장 내에서가 아닌, text data 전체를 기준으로도 계산. 다만 매우 큰 저장공간이 필요하다.

beach | I,went,to,the

$$P_{cBLM}(w_i|W_{<i}) = \text{softmax}\left(\overline{\text{FFN}}(\tilde{h}_{cBLM}^{w_i})\right),$$
$$\tilde{h}_{cBLM}^{w_i} = \sum_{t=2}^{i-1} \alpha^{W_{<t}} h_{[CLS]}^{K,W_{<t}},$$

weighted sum of {CLS(I), CLS(I,went), CLS(I,went,to), CLS(I,went,to,the)}

- Neural-based Dynamic Adaptation
  - 여러 LM들을 self-attention, feed-forward NN 활용해 혼합하는 방법

$$[\alpha_{NNLM}; \alpha_{LSTMLM}; \alpha_{xBLM}; \alpha_{cBLM}; \alpha_{gcBLM}] =$$
$$\text{softmax}\left(\text{FFN}\left(\text{selfatt}([\tilde{h}_{NNLM}^{w_i}; \tilde{h}_{LSTMLM}^{w_i}; \tilde{h}_{xBLM}^{w_i}; \tilde{h}_{cBLM}^{w_i}; \tilde{h}_{gcBLM}^{w_i}])\right)\right).$$

2. A BERT-based Language Modeling Framework

- EXPERIMENT
  - 단어별 log perplexity 및 re-ranking 위한 WER 계산
  - Penn Treebank, Wikitext2, Tedium Release2로 테스트
  - (for re-ranking task) ASR : kaldi tdnn-lstm 레시피로 훈련, 100-best hypothesis 추려냄

- RESULT
  - NN + LSTM + xBLM + cBLM + gcBLM 혼합 모델이 좋은 성능을 보임
    - BERT-Large-CAS, GPT-2 등 거대 LM에 비해 perplexity는 조금 높지만, parameter-efficient하다는 점을 강조 (190M vs 395M, 1542M)
    - n-best re-ranking task : 기존 모델 (trigram, NN, LSTM 기반 LM)에 비해 가장 낮은 WER 기록

2. A BERT-based Language Modeling Framework

Table 1: *The perplexity and word error rates with respect to different models, settings and datasets.*

		Perplexity						**Word Error Rate**	
		Penn Treebank		Wikitext 2		Tedlium Release 2		Tedlium Release 2	
	Model	Dev.	Test	Dev.	Test	Dev.	Test	Dev.	Test
	Trigram	229.86	220.93	274.15	253.32	213.32	195.11	8.12	8.23
	Trigram + LSTMLM	100.18	98.23	130.63	127.51	134.81	137.28	7.66	7.72
	Trigram + LSTMLM + xBLM + cBLM + gcBLM	53.46	52.81	52.03	51.58	56.00	56.61	6.49	6.69
	NNLM	182.86	174.64	207.30	210.66	169.12	175.08	8.02	8.12
	NNLM ◦ LSTMLM	82.36	79.61	107.15	105.28	110.26	113.01	7.31	7.45
	NNLM ◦ LSTMLM ◦ xBLM	48.97	48.64	33.99	33.51	55.16	55.45	6.46	6.61
190M	NNLM ◦ LSTMLM ◦ xBLM ◦ cBLM ◦ gcBLM	**47.56**	**46.87**	**31.41**	**31.01**	**53.05**	**53.77**	**6.38**	**6.51**
	NNLM ◦ BLM	56.84	56.46	54.76	53.84	63.53	64.04	6.75	6.95
	NNLM ◦ xBLM	53.11	52.78	37.86	37.03	57.00	57.57	6.58	6.77
	NNLM ◦ cBLM	65.99	65.64	68.89	68.01	79.49	81.83	7.10	7.22
	NNLM ◦ gcBLM	66.31	65.16	72.74	71.89	81.51	83.91	7.10	7.22
395M	BERT-Large-CAS [49]	**36.14**	**31.34**	**37.79**	34.11				
1,542M	GPT-2 [21]		35.76		**18.34**				

2. A BERT-based Language Modeling Framework

- 감상

- 음성인식 모델에서 LM의 역할 및 BERT의 활용 방안에 대해 잘 정리된 논문이었다.
- BERT-based LM을 기존 LM과 함께 사용했을 때 perplexity, WER 모두 눈에 띄게 감소한 것이 인상적이었다.
- 제안된 cache model, 특히 저장공간을 많이 요구하는 gcBLM 등을 사용하지 않아도 성능에 큰 차이는 없는 것 같아 “NNLM。LSTMLM。xBLM” 혼합 모델을 사용하면 실용적일 것 같다.

3. Sentence-Select: Large-Scale Language Model Data Selection for Rare-Word Speech Recognition

ABSTRACT

- LM 차원에서 rare-word recognition 성능을 저해하는 원인 3가지를 정의하고 해결책 제시

💡 내용을 입력하세요

1. heavy-headedness
    - ‘weather’와 같이 인기 많은 단어가 큰 용량을 차지하는 현상
    - soft log function 통해 전반적 분포 유지하면서 방지
    - f1 = fc log(1 + f0/fc)
2. rare-word 빈도가 적다
    - 음성 데이터에 잘 나오지 않는 단어들을 찾기 위해 필터링
3. domain mismatch
    - perplexity-based contrastive data selection

- sentence-select 없이 훈련된 LM에 비해 WER이 상대적으로 (최대 24%) 감소함 (on rare-word sentences)

3. Sentence-Select: Large-Scale Language Model Data Selection for Rare-Word Speech Recognition

- METHOD

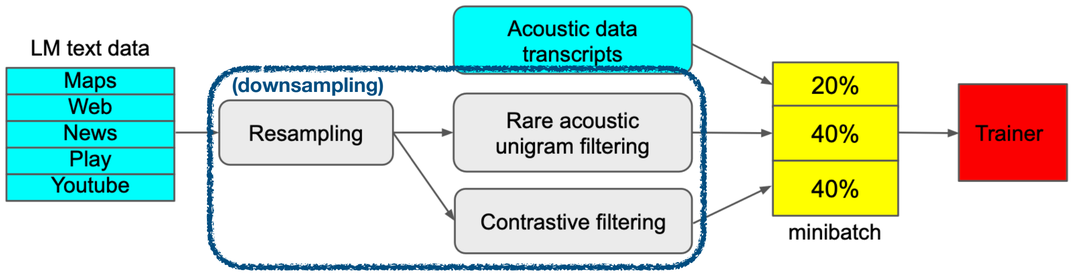


Figure 1: *Overall data selection pipeline.*

f0 : original frequency
fc : downsampled frequency (parameter)

$$f_1 = f_c \log(1 + f_0/f_c)$$

- Downsampling : ‘soft-log function’ 이용해 데이터 내 rare words - frequent words 비율 조절
- Rare-word filtering : 전체 데이터에서 threshold(15)보다 빈도수가 적은 unigram을 포함한 문장들만 걸러냄
- Contrastive data filtering : 앞서 필터링된 문장들 속 unigram은 도메인에 맞지 않을 확률이 높아, perplexity score > threshold인 경우 training data에서 제외

$$\mathcal{L}_{target}(x) - \mathcal{L}_{background}(x)$$

2. A BERT-based Language Modeling Framework

- SETTING
  - AM : E2E (streaming RNNT) / LM : Conformer LM
- Base model
  - E2E만
  - LM(raw text) + E2E
  - LM(중복제거) + E2E
- Test data의 두 가지 도메인
  - voice search (VS)
  - TAIL : 어렵거나 자주 쓰이지 않는 단어들로 구성된 문장 모음

2. A BERT-based Language Modeling Framework

- RESULT

  A. downsampling

  - log perplexity / WER 큰차이 x
  - 데이터 사이즈 감소 (4.1x)

  B. rare-word filtering

  - TAIL에서는 WER 줄어들지만 VS에서 더 높아짐

  C. rare-word filtering + contrastive data filtering

  - VS, TAIL 두 분야에서 모두 WER 소폭 감소

	Corpus size (B)	Log Perplexity VS	Log Perplexity TAIL	WER VS	WER TAIL
B0: No LM				5.6	32.81
B1: Raw	213.7	2.94	3.01	5.4	27.81
B2: Fully deduplicated	7.2	2.92	3.00	5.4	28.03
E1: SimplePower-2.08	61.6	2.93	2.93	5.4	27.31
E2: SimplePower-2.32	31.9	2.93	2.92	5.4	27.32
E3: SimplePower-2.84	18.4	2.90	2.90	5.4	27.39
E4: SoftLog-0	129.3	2.94	2.93	5.5	27.51
E5: SoftLog-0.5	108.0	2.92	2.92	5.5	27.40
E6: SoftLog-1	86.7	2.92	2.90	5.4	27.33
E7: SoftLog-2	51.9	2.91	**2.89**	5.4	**27.25**
E8: SoftLog-3	29.7	**2.89**	**2.89**	5.5	27.36

	Mixing ratio (D, A, R, C)	Corpus size (B)	WER VS	WER TAIL
B3: Downsampled	50, 50, 0, 0	52.19	5.5	27.48
E9: Rare	0, 50, 50, 0	3.19	5.7	25.87
E10: Contrastive	0, 50, 0, 50	0.72	5.3	29.52
E17: Mix-40/20/40	0, 40, 20, 40	3.91	5.4	27.03
E18: Mix-40/40/20	0, 40, 40, 20	3.91	5.4	26.77

2. A BERT-based Language Modeling Framework

- 감상

- 여러가지 방법으로 성능에 큰 영향 없이 데이터 규모를 압축할 수 있다는 점을 배웠다.
- 반대로, 성능에 큰 차이가 없으므로 데이터가 너무 방대해 압축해야 하는 경우가 아니라면 의미가 크지 않을 것 같다.