# Efficient Methods for Natural Language Processing: A Survey

This is just the beginning of something big.

# Abstract

**Recent work in NLP’s efficient methods.**

**What is Efficiency?**
**→ “relation btwn resources and output”**
**→ 같은 성능을 더 적은 자원으로 낼 수 있다면 더 효율적이라고 할 수 있다.**
**→ SOTA 이상의 성능을 내는 것도 efficiency를 개선하는 것의 일종이다.**

# Efficiency

오늘 다룰 테마들을 살펴보자면 다음과 같다.

- Data
- Model Design
- Pre-training
- Fine-tuning
- Inference and Compression
- Hardware Utilization
- Evaluating Efficiency
- Model Selection

# Data

**Filtering: 학습 전후로 중복·노이즈·편향을 제거하여 유효 정보 밀도를 높이는 정제 기법**

**Active Learning: 수집·라벨링 단계에서 가치 높은 샘플만 선택해 라벨 비용을 줄이는 전략**

**Curriculum Learning: 샘플 제시 순서를 쉬운 것 → 어려운 것으로 설계해, 목표 성능에 도달하는 학습 스텝 수를 줄이는 방법**

- **사전학습과 미세조정 모두에서 수렴을 빠르게 하고 연산 낭비를 줄임.**

# Model Design

**Improving Attention in Transformers:**

- 문제: 트랜스포머의 self-attention은 시퀀스 길이의 제곱에 비례하는 비용을 가진다. 따라서 긴 문서에서 비효율이 있음.
- 해법:
  - 글로벌·로컬 분리: 중요한 토큰만 전역으로 보고 나머지는 국소로 봄 → 가까운 선형 비용
  - 희소/버킷팅/패턴화: 일부 연결만 계산
  - 커널 트릭/저랭크: 근사로 선형형(attention) 흉내
  - 메모리 재사용/압축, IO-aware 커널: 실제 하드웨어 병목 완화

# Model Design

## 2.2 Sparse Modeling

- 필요한 부분만 계산하기. 대표적으로 MoE(다음 슬라이드에서 알아보자)
- 비슷한 품질을 적은 연산/에너지로 달성.
  주의: 학습 안정화, 라우팅, 구현 복잡도가 어려운 포인트.

## 2.3 Parameter Efficiency (파라미터 효율)

- 파라미터 공유, 저랭크, 부분 업데이트 등으로 모델 크기/학습 가능 파라미터를 줄이는 설계.
- 왜 효율적? 메모리/시간 절약, 배포도 쉬움.

# Model Design

**MoE: Mixture of Experts**
**큰 모델을 여러 개의 전문가(서브네트워크)로 쪼개 놓고, 입력마다 필요한 몇 개만 골라 계산하는 방식**

1. **게이트(gating) 네트워크가 입력 토큰을 보고 “어느 전문가가 잘 풀 것 같은지” 점수(로짓)를 매김**
2. **보통 Top-k(예: k=1 또는 2) 전문가만 선택해 그 전문가 FFN(서브네트)만 실행**
3. **선택된 전문가들의 출력을 가중합해서 다음 레이어로 넘김**

**즉, 전체 파라미터 수는 매우 크지만, 한 토큰이 실제로 거치는 파라미터 수(활성 파라미터)는 작아서 연산을 아낄 수 있음**

# Model Design

## 2.4 Retrieval-Augmented Models (검색 결합 모델)

- 모델이 외부 지식베이스를 검색해 필요한 정보를 가져와서 사용(파라미터만으로 기억하지 않음).
- 파라미터를 크게 키우지 않고도 지식 보강. 데이터스토어 최적화로 지연도 관리 가능.

## 2.5 Model Design Considerations (설계 시 고려사항)

- 핵심 포인트: 초장문 처리, 포지셔널 인코딩 선택, 희소/선형 주의의 일반화 거동은 아직 연구 중.

# Pre-training

**Optimization Objective (학습 목표 설계)**

- **CLM/MLM/RTD/시퀀스 복원 등 무엇을 예측하게 할지 정하기**
- **마스킹/노이즈 설계가 좋으면 같은 데이터로도 더 빨리/깊게 학습함**

**Pre-training Considerations (사전학습 유의점)**

- **성능 향상은 데이터 품질/모델 설계/목표에서 크게 온다.**
  **자원(메모리·연산)과 데이터 이슈를 현실적으로 관리해야 함.**

# Fine-Tuning

**Parameter-Efficient Fine-Tuning, PEFT**

- 일부만 새로 학습(Adapters, Prefix/Prompt-tuning, LoRA 등).
- 저장·시간 절약, 성능이 대체로 잘 나옴.
  어떤 레이어/랭크를 쓰느냐에 따라 품질·지연이 달라짐.

**Multi-Task & Zero-Shot**

- 여러 과제를 함께 학습해 범용성을 키우거나, 프롬프트만으로 새 과제에 사용.
- 매 과제마다 풀튜닝을 하지 않아도 되어 비용 절감.

# Fine-Tuning

**Prompting (프롬프트)**

- 과제를 텍스트로 잘 표현해 모델을 안내.
  주의: 좋은 프롬프트 탐색이 또 하나의 비용이 될 수 있어 자동화 연구가 많음.

**Fine-tuning Considerations (미세조정 유의점)**

- 요지: 초대형 모델은 풀튜닝이 비싸니 PEFT/프롬프트/멀티태스크 같은 가벼운 방법을 적극 활용.

# Inference&Compression

**Pruning (가지치기)**

- 무엇? 덜 중요한 가중치/헤드/레이어를 제거.
- 왜 효율적? 모델을 가볍게 해서 지연·메모리 감소.
  주의: 비구조 프루닝은 정확도 보존이 좋지만 실속도 이득이 적을 수 있음 (하드웨어에 안 맞음). 구조 프루닝은 반대.

**Knowledge Distillation (지식 증류)**

- 교사 → 학생 모델로 지식 전수.
- 작은 모델로도 성능 유지/개선.
  주의: 학생 튜닝 비용, 일부 정보 손실 가능.

# Inference&Compression

## Quantization (양자화)

- 고정밀 수치 → 저정밀 표현(예: FP32→INT8/4).
- 왜 효율적? 메모리·대역폭 크게 절약, 속도↑.
  주의: 정확도 하락 방지를 위해 혼합정밀/교정/증류 등을 함께 씀.

## Inference Considerations (추론 유의점)

- 실제 서비스 맥락(지연/처리량/비용)에 맞춰 빔서치/병렬화/얼리-엑싯/동적계산 등을 조합.

# Evaluating Efficiency

**Measures (지표)**

- 정확도뿐 아니라 시간, FLOPs, 메모리, 전력/탄소를 함께 봄.
- 상황마다 목표가 다르니 파레토 관점(여러 축의 균형)이 필요.

**Open Challenges (열린 과제)**

- 지표 충돌: 예를 들어, 희소성은 FLOPs를 줄여도 인덱싱 비용으로 메모리가 늘 수 있음.
- 공정성/견고성: 프루닝·증류가 편향에 영향 줄 수 있어 정확도 너머 평가가 필요

# Model Selection

## Hyperparameter Search (하이퍼파라미터 탐색)

- BO/ASHA 같은 효율 탐색 기법.
- 적은 시도로 좋은 설정을 찾음. 다만 초기화/데이터 순서 영향도 존재.

## 8.2 Hyperparameter Transfer (전이)

- 작은 모델이나 다른 과제에서 찾은 좋은 설정을 전이.
- 큰 모델/새 과제에서 탐색 비용 절감.

## 8.3 Considerations (유의점)

- 요지: 공정 비교를 위해 튜닝 예산을 고정하고, 보고 관행을 투명하게.