

# Relation Extraction

---

건국대학교 컴퓨터공학부 /  
KAIST 전산학부 (겸직)

김학수

# Relation Extraction (RE)

- 관계 추출
  - 문장 혹은 문서에 존재하는 개체들 간의 의미적 관계를 찾아내는 작업

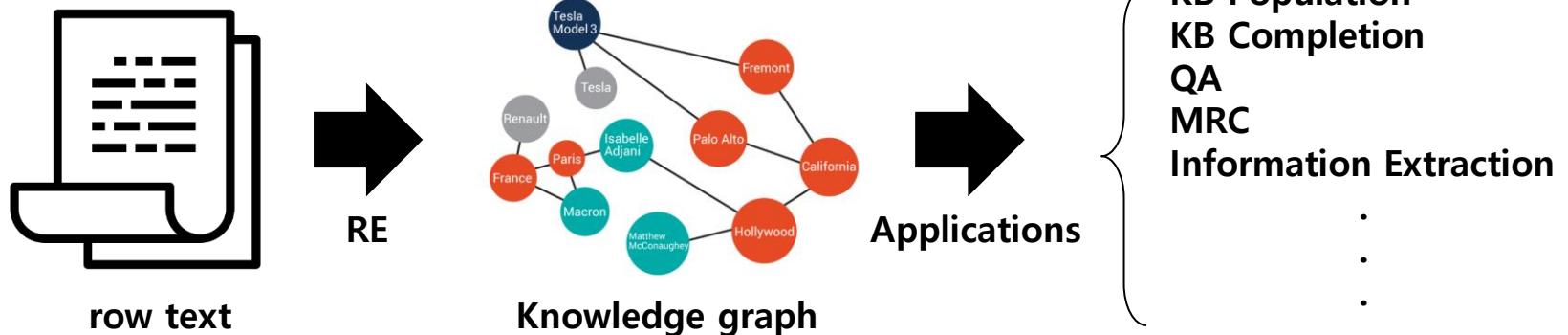
이건희는 대한민국 기업인 삼성의 회장이다. 또한, 삼성  
이재용 부회장의 아버지이다.

개체1	개체2	관계
이건희	삼성	Owner
삼성	대한민국	Located
이재용	삼성	Employee
이건희	이재용	Family



# From Relation To Knowledge

- 관계 추출 → 지식 추출
  - 개체 간 관계를 표현하는 트리플(triple) 생성
  - 트리플을 기반으로 지식 베이스 구축
  - 다양한 응용 태스크에서 활용 가능



# Technical Categories

---

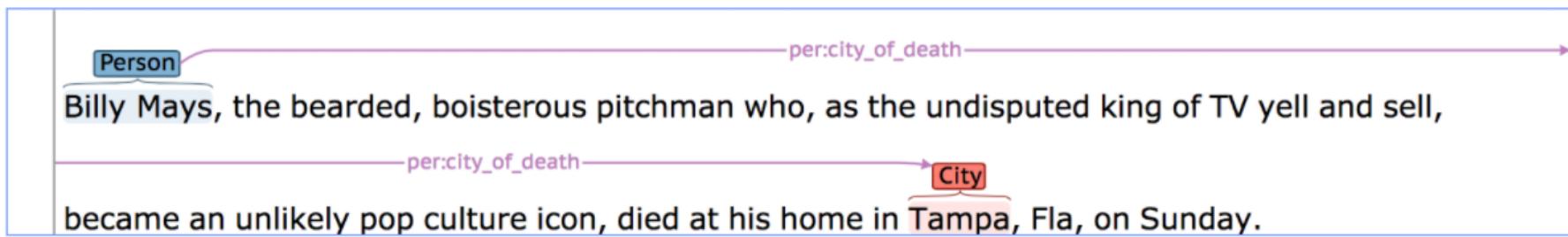
- 문장 수준 관계 추출 (Sentence-level RE)
    - 한 문장 내에 존재하는 개체들 간의 관계를 추출
    - 대표 말뭉치: TAC Relation Extraction Dataset (TACRED)
  - 문서 수준 관계 추출 (Document-level RE)
    - 여러 문장으로 구성된 문서 내에 존재하는 개체들 간의 관계를 추출
    - 대표 말뭉치: Document-level Relation Extraction Dataset (DocRED)
  - 종단형 관계 추출 (End-to-End RE)
    - 문장 내에 존재하는 개체명을 찾고 그들 간의 관계를 추출
    - 대표 말뭉치: ACE 2005 (Automatic Content Extraction Dataset 2005)
- 



# Sentence-level Relation Extraction

- 문장 수준 관계 추출 특징
  - 문장에 존재하는 두 개체 간의 관계 추출
  - 한번에 하나의 트리플을 추출, 대부분 no-relation
  - 언어 모델, 개체 인코딩 방법에 따라 성능이 좌우됨
  - 개체 정보, 구문 정보 등의 자질이 중요함

TACRED, SEMEVAL,  
ACE 등의 말뭉치  
특징



# Leaderboard on TACRED (with Paper)

<https://paperswithcode.com>

Rank	Model	F1 ↑	Extra Training Data	Paper	Code	Result	Year
1	RECENT+SpanBERT	75.2	✗	Relation Classification with Entity Type Restriction			2021
2	KU_NLP (baseline)	74.9	✗	Improving Sentence-Level Relation Extraction through Curriculum Learning			2021
3	Relation Reduction	74.8	✗	Relation Classification as Two-way Span-Prediction			2020
4	RoBERTa-large-typed-marker	74.6	✗	An Improved Baseline for Sentence-level Relation Extraction			2021
5	LUKE	72.7	✓	LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention			2020
6	DeNERT-KG	72.4	✓	DeNERT-KG: Named Entity and Relation Extraction Model Using DQN, Knowledge Graph, and BERT			2020

Key Approaches

Entity Marking, Restriction

Task-Oriented Language Modeling, Knowledge Graph

Span Prediction



Edited by Harksoo Kim

# Entity Marking & Restriction

- Entity Marking

- 개체 태입 정보를 언어모델에 추가하여 관계 추출 성능을 개선
- 특수기호로 개체 마킹 후 해당 기호의 인코딩 벡터를 개체 인코딩 벡터로 사용

Method	Input Example	BERT <sub>BASE</sub>	BERT <sub>LARGE</sub>	RoBERTa <sub>LARGE</sub>
Entity mask	[SUBJ-PERSON] was born in [OBJ-CITY].	69.6	70.6	60.9
Entity marker	[E1] Bill [/E1] was born in [E2] Seattle [/E2].	68.4	69.7	70.7
Entity marker (punct)	@ Bill @ was born in # Seattle #.	68.7	69.8	71.4
Typed entity marker	<code>(S:PERSON) Bill (/S:PERSON) was born in (O:CITY) Seattle (/O:CITY).</code>	<b>71.5</b>	<b>72.9</b>	71.0
Typed entity marker (punct)	@ * person * Bill @ was born in # ^ city ^ Seattle #.	70.9	72.7	<b>74.6</b>

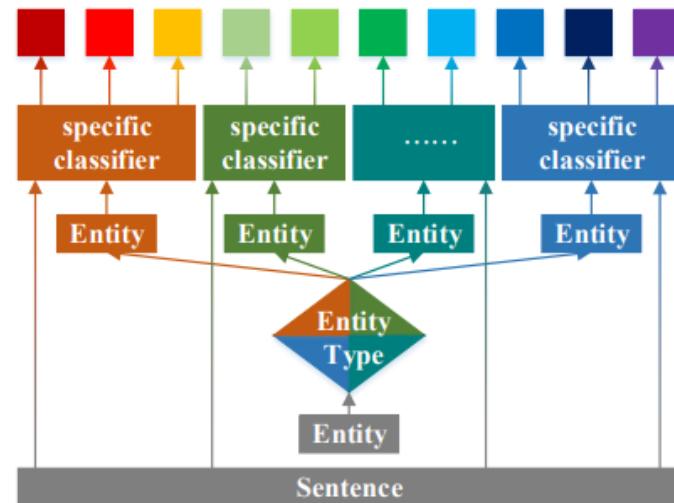
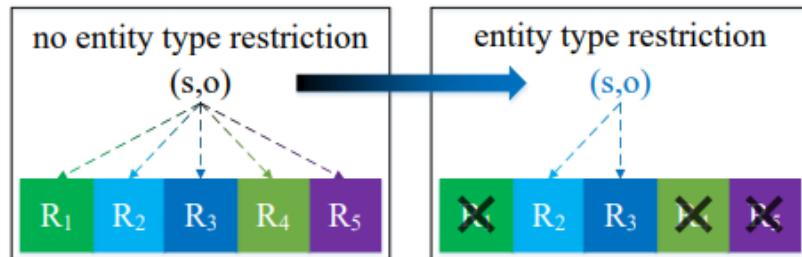


출처: Zhou et al., An Improved Baseline for Sentence-level Relation Extraction arXiv:2102.01373, 2021  
Soares et al., Matching the Blanks: Distributional Similarity for Relation Learning, ACL 2019



# Entity Marking & Restriction

- **Restriction**
  - 개체 쌍 태입에 따라 가질 수 있는 관계 태입 제약 적용
  - 각 개체 쌍 태입의 관계 제약마다 판별기 구축

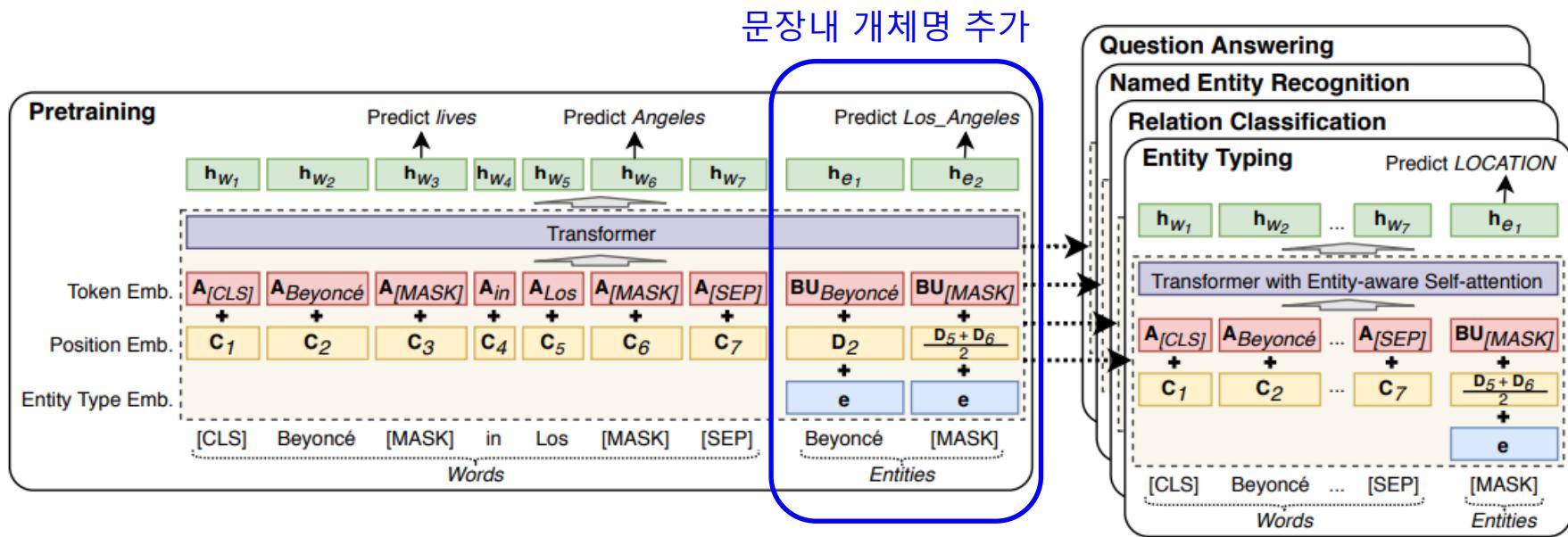


출처: Lyu et al., Relation Classification with Entity Type Restriction, arxiv:2105.08393, 2021



# Language Modeling & External Knowledge

- LUKE (Language Understanding with Knowledge-based Embeddings)
    - 언어 모델에 개체 정보 반영을 위한 학습 방법 제안
    - 개체명 부착된 말뭉치를 활용한 언어 모델 추가 학습



출처: Yamada et al., LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention, EMNLP 2020



# Span prediction

- Relation reduction

- 관계 추출을 span prediction 문제로 변형해서 수행

$$f_{rc}(c, e_1, e_2) \mapsto r \quad \xrightarrow{\text{purple arrow}} \quad f_{qa}(c, e_q, r_q) \mapsto e$$

- 각 관계 태입에 해당하는 질의 템플릿 제작

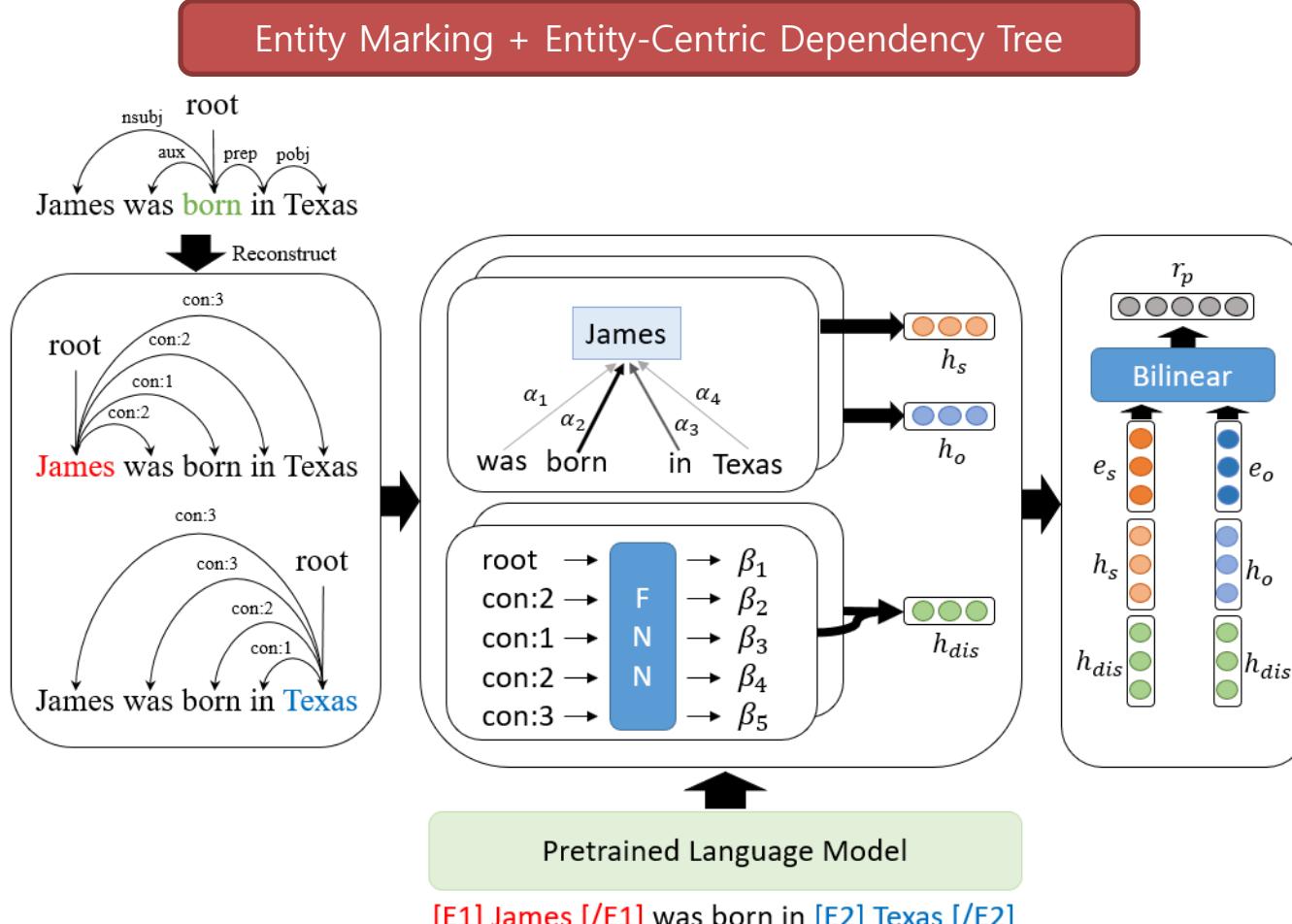
RC sample	Relation candidates	Question (Reverse Question)	Answer
<b>John was born on 1991</b>	"Date of birth"	When was John Born? (Who was born on 1991?)	1991 John
	"Date of death"	When did John die? (Who died on 1991?)	N/A N/A
<b>Mary is John's employer</b>	"employer of"	Who is employed by Mary? (Who is John employer?)	John Mary
	"siblings"	Who is the sibling of Mary? (Who is the sibling of John?)	N/A N/A
	"parents of"	Who is the child of Mary? (Who is the parent of John?)	N/A N/A



출처: Cohen et al., Supervised Relation Classification as Two-way Span-Prediction, arxiv:2010.04829, 2020.



# Sentence-level RE Model in KUNLP

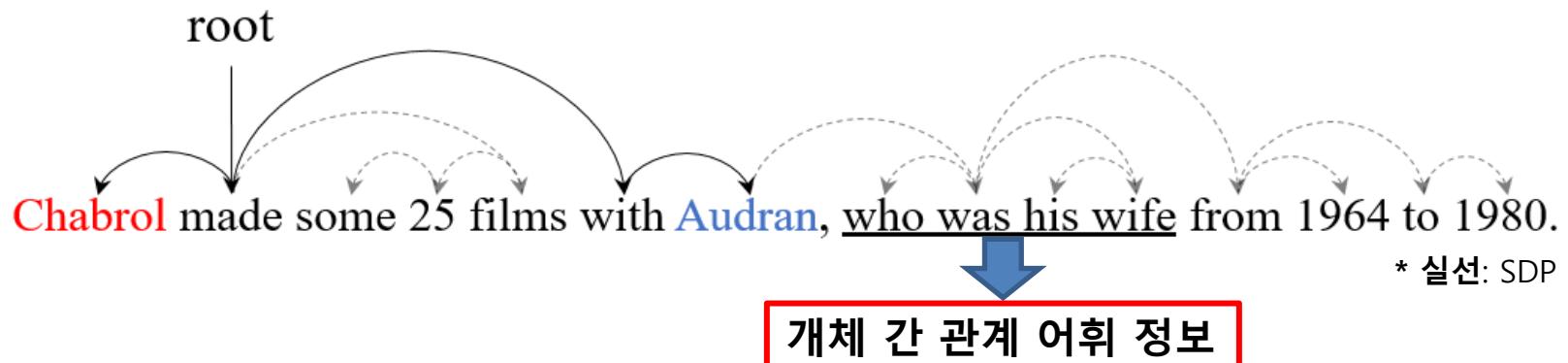


# Dependency Features

- 구문 정보
  - 문장 수준 관계 추출에서 중요하게 여겨지는 자질
  - LCA(Lowest common ancestor), SDP(Shortest dependency path)를 사용하면 최적의 정보로 관계 추출 가능

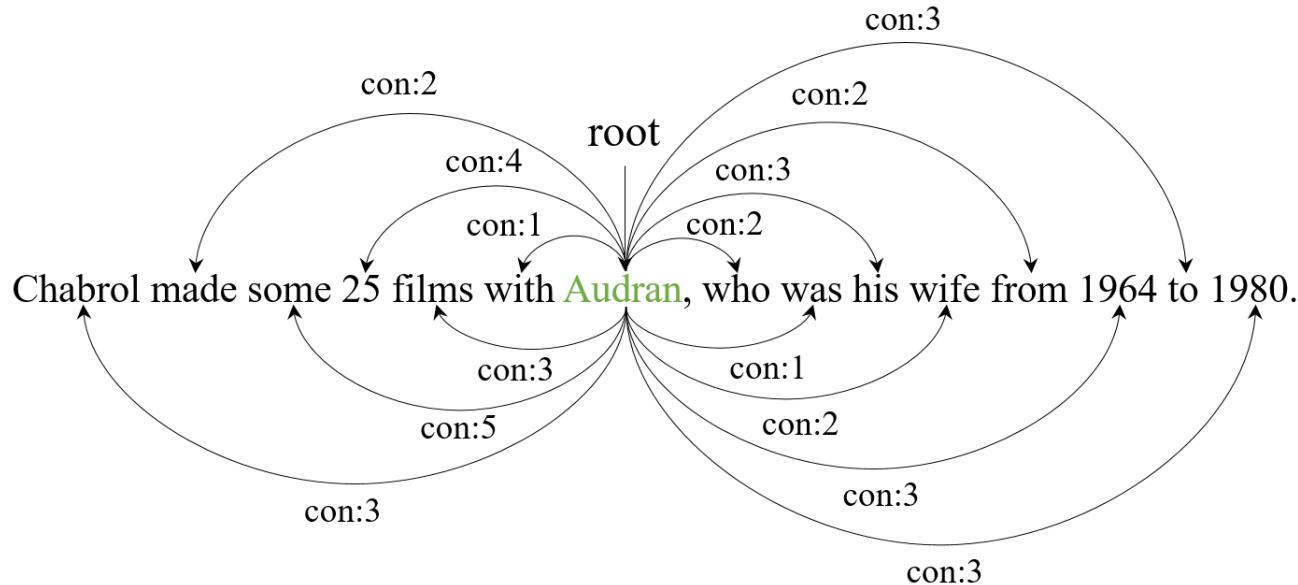
But!

관계 추출에 필요한 어휘 정보가 누락될 가능성 존재



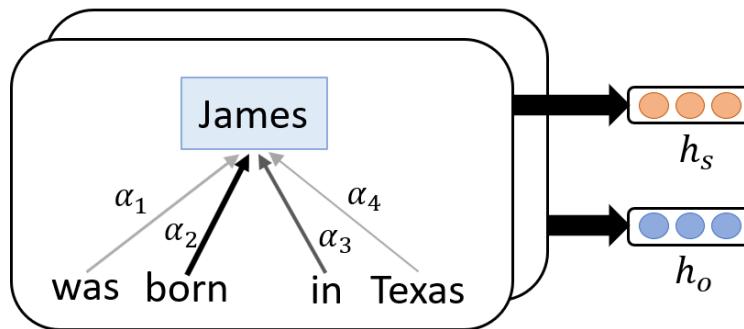
# Entity-Centric Dependency Tree

- 개체 중심 구문 트리
  - SDP, LCA 활용 시 정보 손실을 해소하기 위해 개체 중심의 구문 트리로 변환
  - Edge: 원본 구문 트리에서의 구문 거리 정보 할당
  - 관계 추출에 어휘 정보 손실 없이 구문 정보 활용 가능



# Entity-Centric Dependency Tree

- Graph Attention Network
  - EDT에 주어진 개체 노드와 단어 노드 사이의 attention 계산
  - 의미적으로 중요한 단어에 집중된 벡터 계산



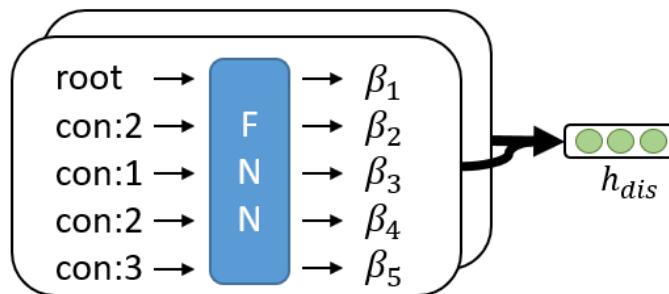
$$h_e = \sum_{i \in N} \alpha_i emb(w_i)$$

$$\alpha = softmax\left(\frac{emb(e) \cdot W_l \cdot emb(w)^T}{\sqrt{d}}\right)$$



# Entity-Centric Dependency Tree

- Dependency information
  - 각 개체와 단어 사이 구문 거리 정보 결합
  - 두 개체 입장에서 구문적으로 중요한 단어에 집중된 벡터 계산



$$\beta_e = \text{softmax}(\text{relu}(W_\beta z + b_\beta))$$

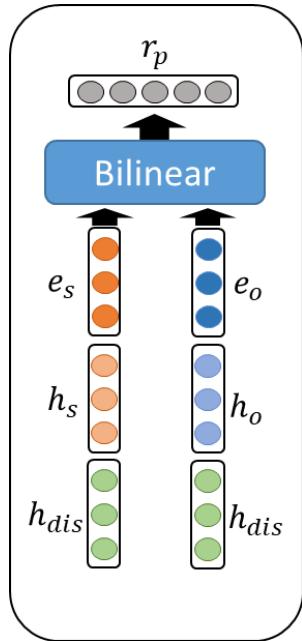
$$\beta = \frac{\beta_s * \beta_o}{\sum_{t=1}^T (\beta_s * \beta_o)_i}$$

$$h_{dis} = \sum_{t=1}^T \beta_t \text{emb}(w_t)$$



# Entity-Centric Dependency Tree

- Relation classification
  - 개체 인코딩, GAT 출력, 개체 사이의 구문 정보를 반영하여 관계 분류 수행



$$g_s = \text{relu}(W_e[\text{emb}(e_s); h_s; h_{dis}] + b_e)$$

$$g_o = \text{relu}(W_e[\text{emb}(e_o); h_o; h_{dis}] + b_e)$$

$$r_p = g_s \cdot W_p \cdot g_o^T$$



# Experiments

---

- Datasets
  - TACRED
  - SemEval 2010 task 8

	TACRED	SemEval 2010 Task 8
Length of sentence (min / max / avg)	6 / 100 / 41.0	5 / 87 / 19.2
# of relation types	42	19
# of entity types	23	-
# of samples (train / dev / test)	68K / 22K / 15K	8K / - / 2.7K
# of positive / negative samples	21K / 84K (22% / 78%)	8.8K / 2K (82% / 18%)



# Experiments

---

- Implementation Details
  - RoBERTa-large 언어 모델 (24 transformer layer, 1,024 hidden, 355M trainable par)

Hyper parameter	Value
Max dependency distance	5
Distance embedding size	128
FNN weight size	768
Drop-out	0.1
Learning rate	3e-5



# Experiments

- Entity-Centric Dependency Tree (EDT)의 효과
  - 언어모델에 상관없이 0.5 ~ 1.5%p 성능 향상

Models	F1	
	Base	EDT
BERT	71.4	71.9
ELECTRA	70.2	71.7
RoBERTa	71.3	71.9

Context	Predict	Target	
The defending league champion, Sky Blue FC, hired a coach from Finland, Pauliina Miettien.	Employee_of	Employee_of	
Chabrol made some 25 films with Audarm, who was his wife from 1964 to 1980.	Spouse	Spouse	최단경로 문제 해결
That too may have resonated with militants in that region, said Ahmed Rashid, a Lahore-based analyst and author of a book on the Taliban.	Title	No_relation	말뭉치 문제 → Re-TACRED
Knox's father, Curt Knox, said his daughter looked "confident in what she wants to say."	No_relation	children	Coref. Resolution 필요



# Experiments

- 성능 비교
  - EDT 활용 시 문장 수준 관계 추출에서 출판 논문 모델 중 SOTA
  - 외부 지식 사용 없이도 가장 좋은 성능을 보임

<TACRED>

Model	F1
CP	69.5
SpanBERT-large	70.8
RC-KnowBERT <sup>+</sup>	71.5
BERT-MTB <sup>+</sup>	71.5
DeNERT-KG <sup>+</sup>	72.4
LUKE <sup>+</sup>	72.7
<b>EDT (Our)</b>	<b>72.8</b>

<SemEval-10 task 8>

Model	F1
Entity-aware BERT	89.0
KnowBERT <sup>+</sup>	89.1
R-BERT	89.3
BERT-MTB <sup>+</sup>	89.5
EPGNN	90.2
Skeleton-aware BERT	90.4
<b>EDT (Our)</b>	<b>90.5</b>

+ : Models Using External Knowledge



# Ranks in TACRED & Re-TACRED (with Paper)

<https://paperswithcode.com>

**TACRED**

Rank	Model
1	RECENT+SpanBERT
2	KU_NLP (baseline)
3	Relation Reduction

**Re-TACRED**

Rank	Model
1	KU_NLP (baseline)
2	RoBERTa-large-typed-marker
3	SpanBERT

Extra  
F1 ↑ Training Data Paper Code Result

2021년 7월 21일 현재

tacred leaderboard

전체 뉴스 이미지 동영상 지도 더보기 도구

검색결과 약 476개 (0.35초)

<https://paperswithcode.com/sota/relation-extraction-...>

**TACRED Benchmark (Relation Extraction) | Papers With Code**

Rank	Model	F1	Year
1	RECENT+SpanBERT	75.2	2021
2	KU_NLP; (baseline)	74.9	2021
3	Relation Reduction	74.8	2020

25행 더보기

<https://paperswithcode.com/sota/relation-extraction-...>

**Re-TACRED Benchmark (Relation Extraction) | Papers With ...**

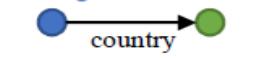
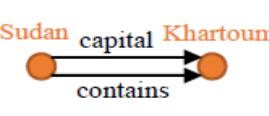
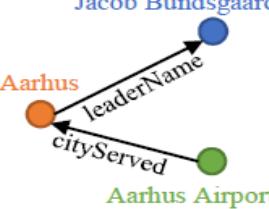
Rank	Model	F1	Year
1	KU_NLP; (baseline)	91.2	2021
2	RoBERTa-large-typed-marker	91.1	2021
3	SpanBERT	85.3	2021

2행 더보기



# Document-level Relation Extraction

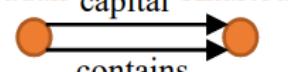
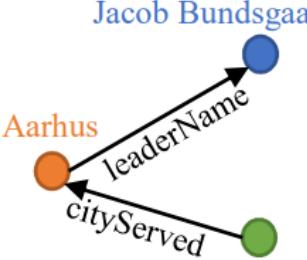
- 문장 수준 관계 추출
  - 한 문장에 개체가 한 쌍만 존재한다고 가정
  - 두 개체 간의 관계 추출만 고려
- 문서 수준 관계 추출
  - 한 문장에 여러 개체가 존재할 수 있음
  - 여러 문장에 걸쳐서 관계를 형성할 수 있음

Normal	S1: Chicago is located in the United States. {<Chicago, country, United States>}	
EPO	S2: News of the list's existence unnerved officials in Khartoum, Sudan 's capital. {<Sudan, capital, Khartoum>, <Sudan, contains, Khartoum>}	
SEO	S3: Aarhus airport serves the city of Aarhus who's leader is Jacob Bundsgaard. {<Aarhus, leaderName, Jacob Bundsgaard>, <Aarhus Airport, cityServed, Aarhus>}	



# DocRED

- Document-Level Relation Extraction Dataset (DocRED)
  - 문서 단위의 관계 추출 연구를 위한 데이터
  - 6개 개체 타입과 96개 관계 타입 존재
  - Entity Pair Overlap(EPO), Single Entity Overlap(SEO) 형태의 관계 추출 해결이 핵심

EPO	S2: News of the list's existence unnerved officials in <b>Khartoum</b> , <b>Sudan</b> 's capital.  $\{\langle \text{Sudan}, \text{capital}, \text{Khartoum} \rangle, \langle \text{Sudan}, \text{contains}, \text{Khartoum} \rangle\}$	<b>Sudan</b> capital <b>Khartoum</b> 
SEO	S3: <b>Aarhus airport</b> serves the city of <b>Aarhus</b> who's leader is <b>Jacob Bundsgaard</b> .  $\{\langle \text{Aarhus}, \text{leaderName}, \text{Jacob Bundsgaard} \rangle, \langle \text{Aarhus Airport}, \text{cityServed}, \text{Aarhus} \rangle\}$	<b>Jacob Bundsgaard</b> <b>Aarhus</b> leaderName <b>Jacob Bundsgaard</b> <b>Aarhus Airport</b> cityServed <b>Aarhus</b> 



# DocRED

- Document-Level Relation Extraction Dataset (DocRED)

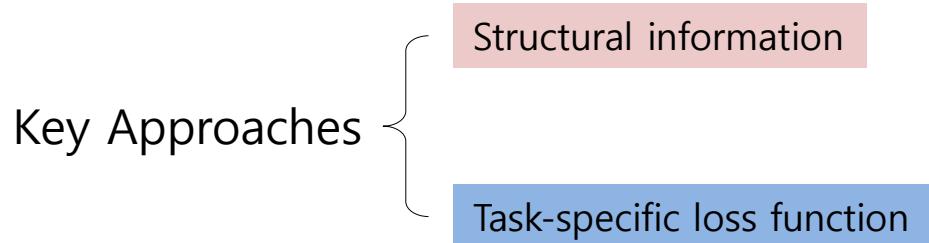
Reasoning Types	%	Examples
Pattern recognition	38.9	<p>[1] <b>Me Musical Nephews</b> is a 1942 one-reel animated cartoon directed by Seymour Kneitel and animated by Tom Johnson and George Germanetti. [2] Jack Mercer and Jack Ward wrote the script. ...</p> <p><b>Relation:</b> <code>publication_date</code>      <b>Supporting Evidence:</b> 1</p>
Logical reasoning	26.6	<p>[1] “Nisei” is the ninth episode of the third season of the American science fiction television series The X-Files. ... [3] It was directed by David Nutter, and written by <b>Chris Carter</b>, Frank Spotnitz and Howard Gordon. ... [8] The show centers on FBI special agents <b>Fox Mulder</b> (David Duchovny) and Dana Scully (Gillian Anderson) who work on cases linked to the paranormal, called X-Files. ...</p> <p><b>Relation:</b> <code>creator</code>      <b>Supporting Evidence:</b> 1, 3, 8</p>
Coreference reasoning	17.6	<p>[1] <b>Dwight Tillery</b> is an American politician of the Democratic Party who is active in local politics of Cincinnati, Ohio. ... [3] He also holds a law degree from the <b>University of Michigan Law School</b>. [4] <b>Tillery</b> served as mayor of Cincinnati from 1991 to 1993.</p> <p><b>Relation:</b> <code>educated_at</code>      <b>Supporting Evidence:</b> 1, 3</p>
Common-sense reasoning	16.6	<p>[1] <b>William Busac</b> (1020-1076), son of William I, Count of Eu, and his wife Lesceline. ... [4] <b>William</b> appealed to King Henry I of France, who gave him in marriage <b>Adelaide</b>, the heiress of the county of Soissons. [5] <b>Adelaide</b> was daughter of Renaud I, Count of Soissons, and Grand Master of the Hotel de France. ... [7] <b>William</b> and <b>Adelaide</b> had four children: ...</p> <p><b>Relation:</b> <code>spouse</code>      <b>Supporting Evidence:</b> 4, 7</p>



# Leaderboard on DocRED (with Paper)

<https://paperswithcode.com>

Rank	Model	F1 ↑	Ign F1	Paper	Code	Result	Year	Tags
1	SSAN-RoBERTa-large+Adaptation	65.92	63.78	Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction			2021	
2	DocuNet-RoBERTa-large	64.55	62.4	Document-level Relation Extraction as Semantic Segmentation			2021	
3	ATLOP-RoBERTa-large	63.40	61.39	Document-Level Relation Extraction with Adaptive Thresholding and Localized Context Pooling			2020	
4	GAIN-BERT-large	62.76	60.31	Double Graph Based Reasoning for Document-level Relation Extraction			2020	



# Structural Information

- SSAN (Structured Self-Attention Network)
  - 문서에서 개체와 단어가 가질 수 있는 구조적 정보 정의
    - intra+coref: 같은 문장에서 같은 개체를 지칭하는 mention
    - intra+relate: 같은 문장에서 다른 개체를 지칭하는 mention
    - inter+coref: 다른 문장에서 같은 개체를 지칭하는 mention
    - inter+relate: 다른 문장에서 다른 개체 지칭하는 mention
    - intraNE: 개체와 같은 문장에서 등장하는 일반 단어
    - NA: 개체와 구조적 관계가 없는 단어

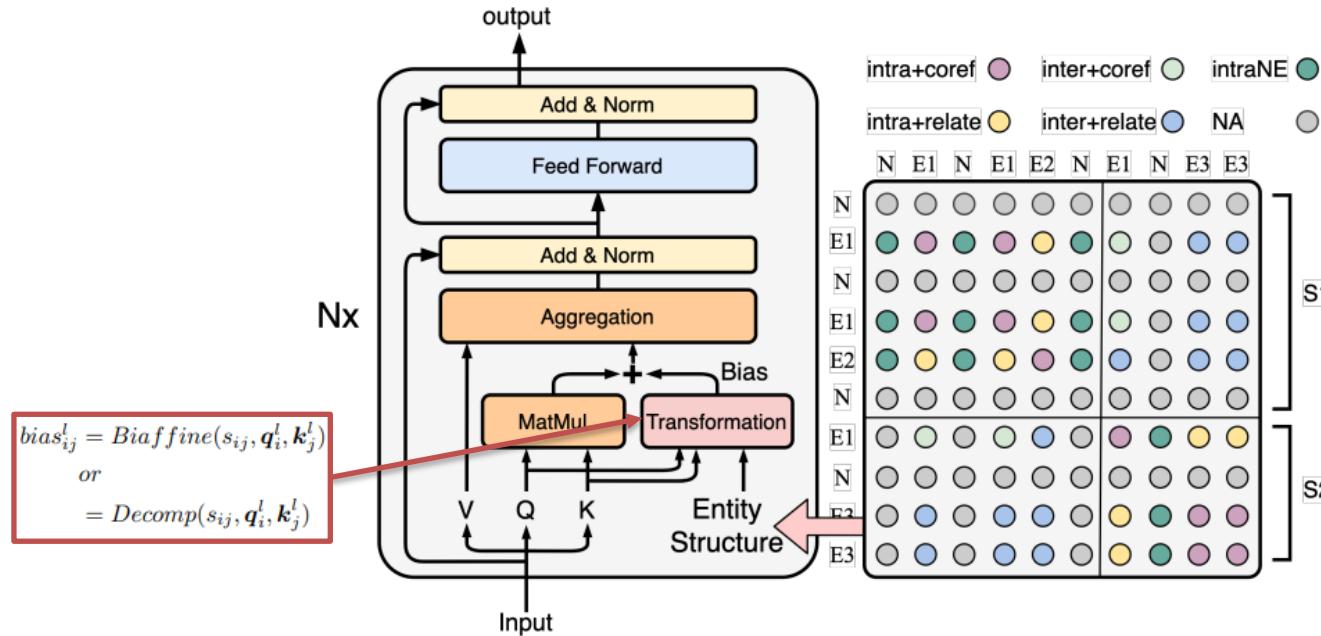
		Coreference	
		True	False
Co-occurrence	True	<i>intra+coref</i>	<i>intra+relate</i>
	False	<i>inter+coref</i>	<i>inter+relate</i>

출처: Xu et al., 2021, Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction, arXiv: 2102.10249



# Structural Information

- SSAN (Structured Self-Attention Network)
  - 문서에서 습득한 구조적 정보를 transformer 계층에서 활용
  - 언어 모델에서 문서의 구조적 정보 직접 반영 가능

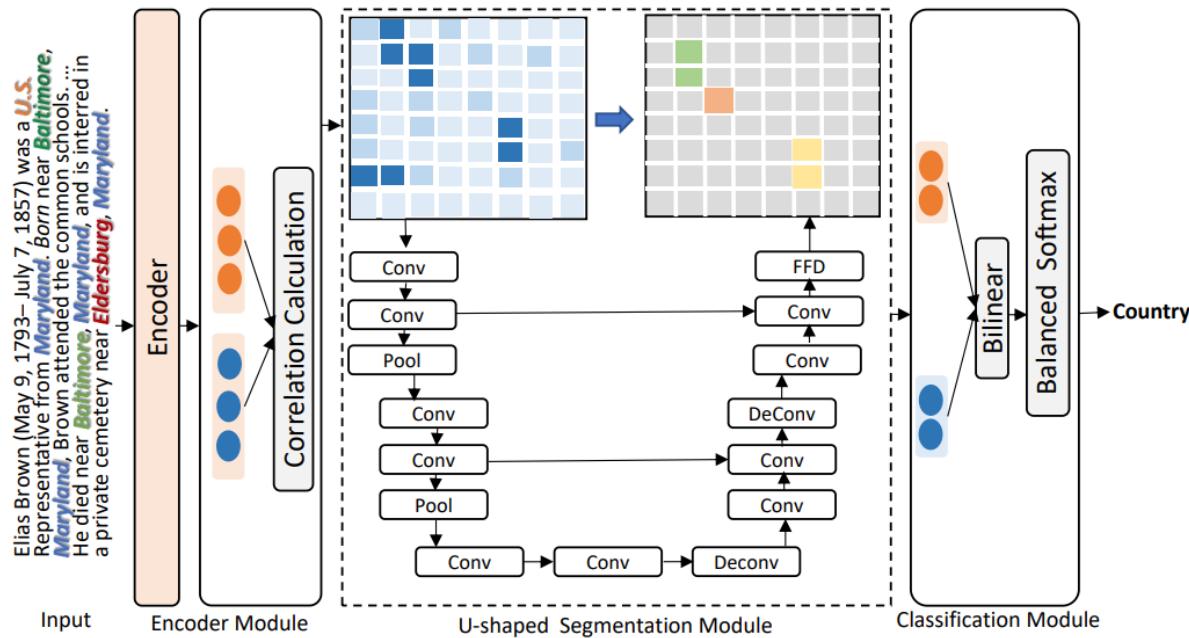


출처: Xu et al., 2021, Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction, arXiv: 2102.10249



# Structural Information

- DocuNet
  - 개체 간 구조 정보를 Conv 연산으로 down-sampling (encoding)
  - DeConv 연산으로 up-sampling (semantic segmentation)



출처: Zhang et al., Document-level Relation Extraction as Semantic Segmentation, IJCAI 2021

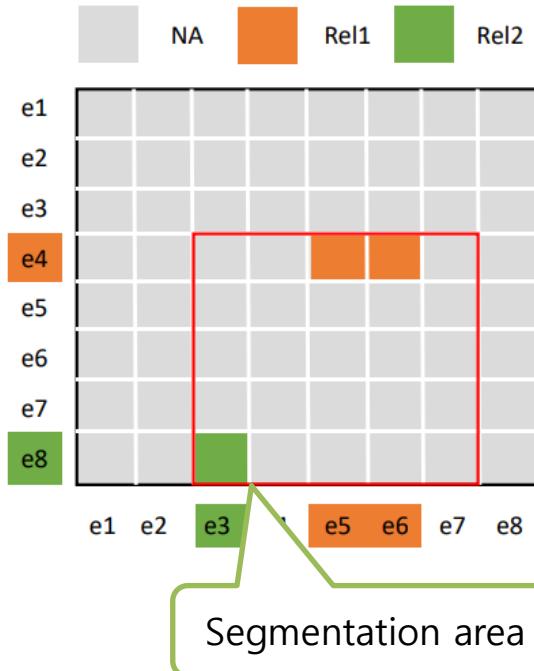


# Structural Information

- DocuNet

- Matrix feature vector

<Entity level relation matrix>



$$\mathbf{F}(e_s, e_o) = [e_s \odot e_o; \cos(e_s, e_o); e_s W_1 e_o] \quad \text{Similarity-based}$$

$$\begin{aligned} \mathbf{F}(e_s, e_o) &= W_2 H a^{(s,o)} \\ a^{(s,o)} &= \text{softmax}\left(\sum_{i=1}^K A_i^s \cdot A_i^o\right) \end{aligned} \quad \text{Context-based}$$

- Relation classification

$$z_s = \tanh(W_s \mathbf{e}_s + Y_{s,o}),$$

$$z_o = \tanh(W_o \mathbf{e}_o + Y_{s,o}),$$

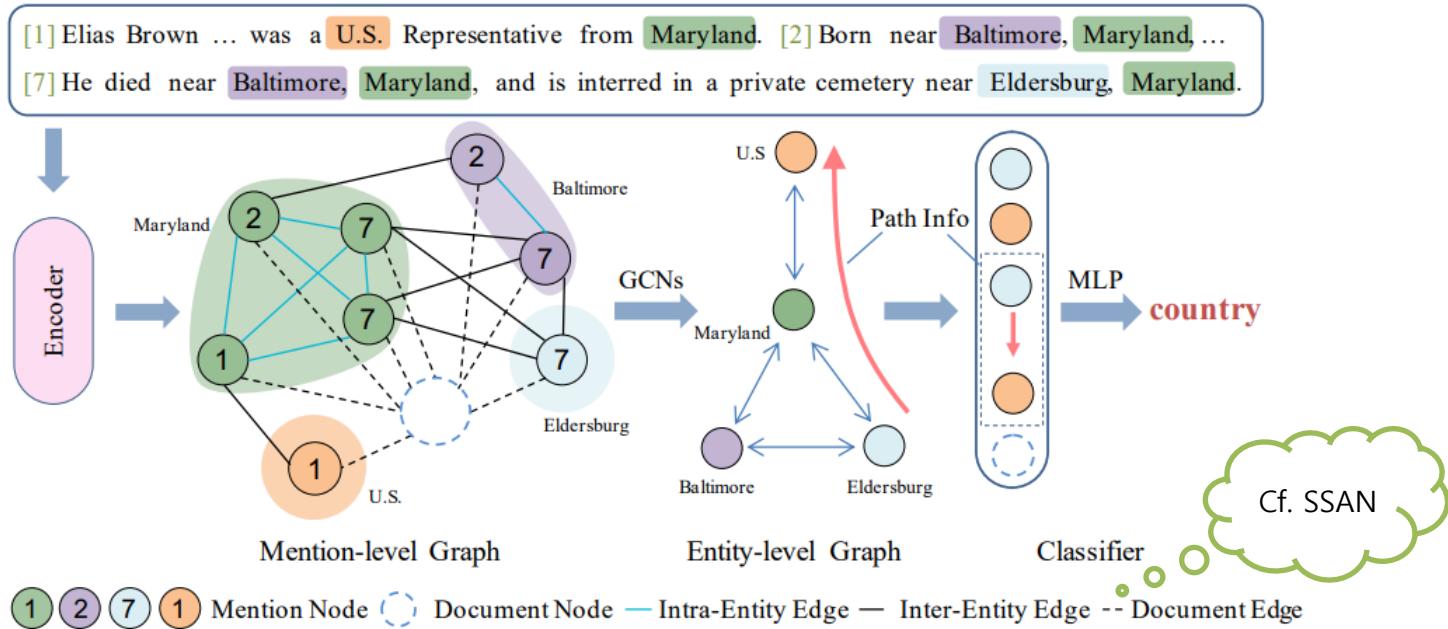
$$P(r|e_s, e_o) = \sigma(z_s W_r z_o + b_r)$$

출처: Zhang et al., Document-level Relation Extraction as Semantic Segmentation, IJCAI 2021



# Structural Information

- GAIN (Graph Aggregation-and-Inference Network)
  - 개체 정보, 문장 정보를 통해 멘션(mention) 그래프 생성
  - 멘션 그래프로 개체(entity) 그래프 생성 후 관계 추출에 활용



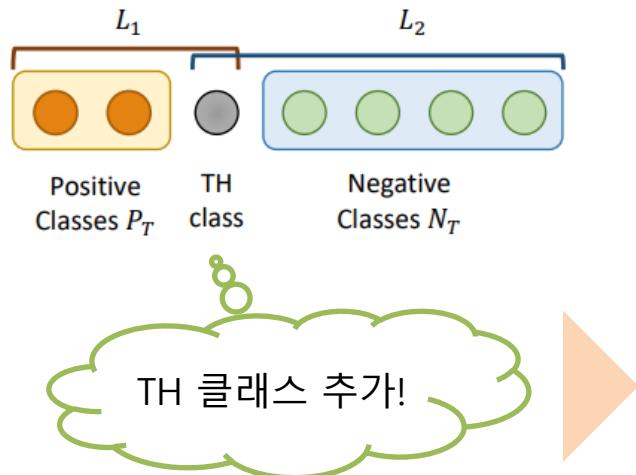
출처: Zeng et al., Double Graph Based Reasoning for Document-level Relation Extraction, EMNLP 2020



# Task-specific Loss Function

- ATLOP (Adaptive Thresholding and Localized cOntext Pooling)
  - 개체 쌍과 관계 타입에 따라 다른 임계 값 적용
  - 임계 값은 모델에서 동적으로 학습

## Categorial Cross-Entropy



$$\mathcal{L}_1 = - \sum_{r \in \mathcal{P}_T} \log \left( \frac{\exp(\text{logit}_r)}{\sum_{r' \in \mathcal{P}_T \cup \{\text{TH}\}} \exp(\text{logit}_{r'})} \right)$$

$$\mathcal{L}_2 = - \log \left( \frac{\exp(\text{logit}_{\text{TH}})}{\sum_{r' \in \mathcal{N}_T \cup \{\text{TH}\}} \exp(\text{logit}_{r'})} \right),$$

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2.$$

예측 시 TH 클래스의 로짓값보다 큰 클래스를 positive 클래스로 선택

출처: Zhou et al., Document-Level Relation Extraction with Adaptive Thresholding and Localized Context Pooling, AAAI 2021



# Document-level RE Model in KUNLP

- 기존 그래프 기반 모델 한계점
  - Mention 인코딩에만 집중하기 때문에 관계 정보를 내포하는 어휘의 정보를 놓칠 수 있음
  - 개체를 지칭하는 대명사(pronoun)의 정보를 활용하지 못 함

<0> Andrea Lilio(1555-1642) was an Italian painter born in Ancona, hence he also is known as L'Anconitano.

<1> L'Anconitano painted mainly in **his** native city, as well as in Rome, where he was active from the beginning of the 17th century until around 1640.

:

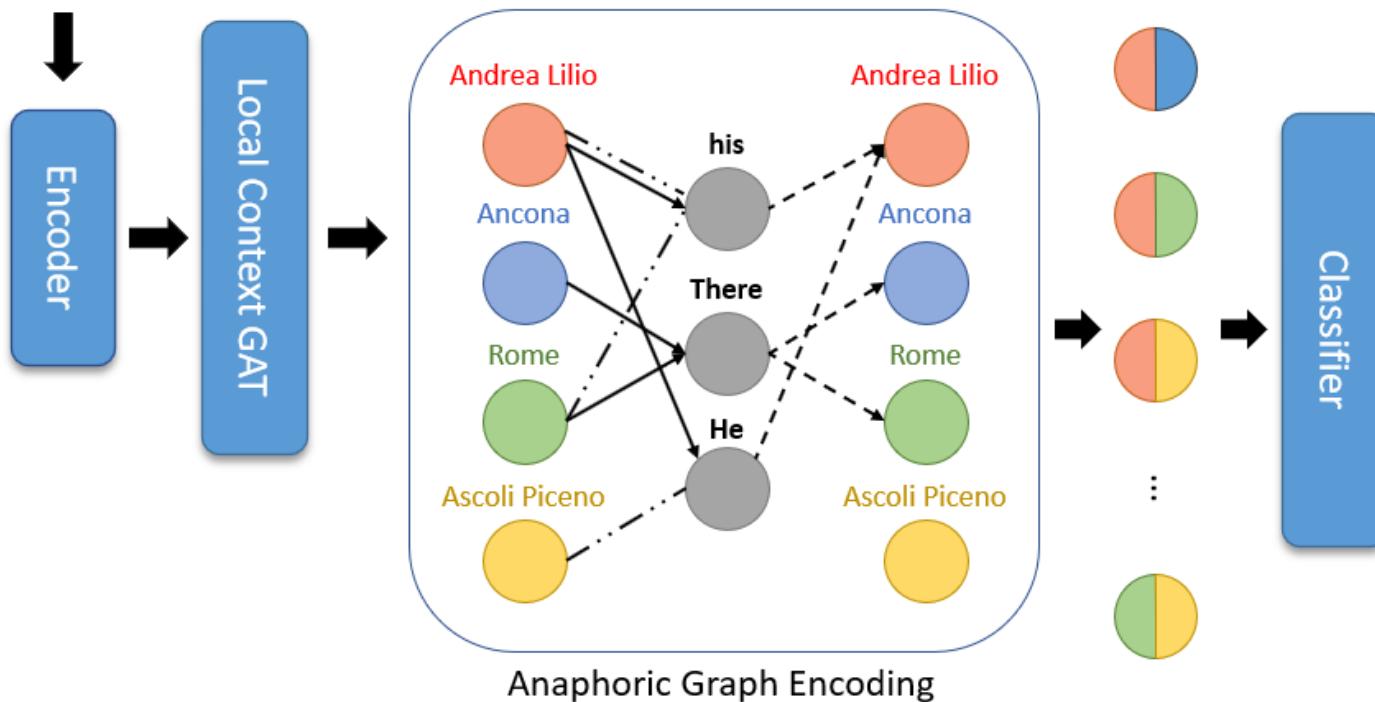
<6> **He** died at Ascoli Piceno.

	Subject	Relation	Object
Triple 1	Andrea Lilio	place of birth	Ancona
Triple 2	Andrea Lilio	citizen of	Rome
Triple 3	Andrea Lilio	place of death	Ascoli Piceno



# Document-level RE Model in KUNLP

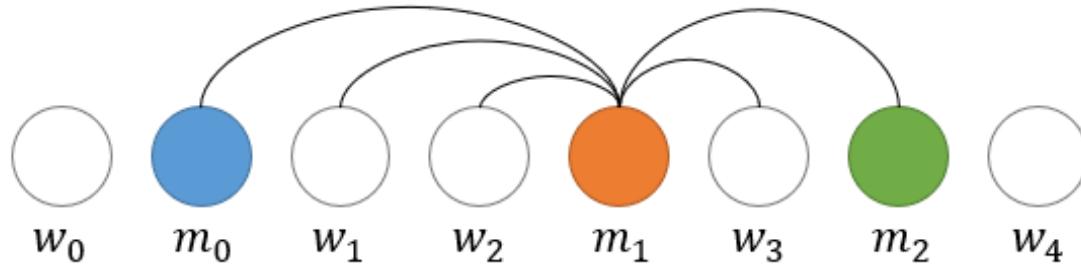
- [0] Andrea Lilio(1555-1642) was an Italian painter born in Ancona, hence he also is known as L'Anconitano.  
[1] L'Anconitano painted mainly in his native city, as well as in Rome, where he was active from the beginning  
of the 17th century until around 1640.  
⋮  
[6] He died at Ascoli Piceno.



# Improving Graph-based RE

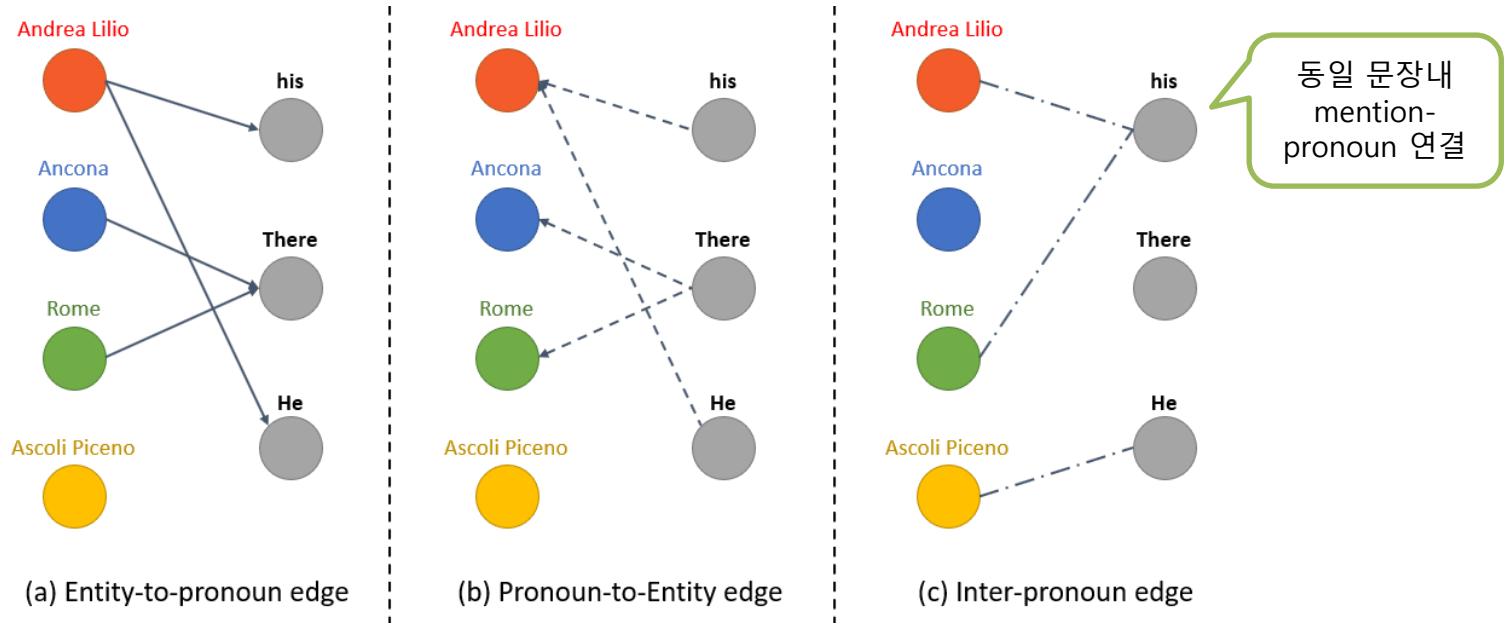
---

- Local-context graph
  - Mention 지역적 어휘 정보 반영을 위한 그래프
  - 현재 mention과 이웃 mention 사이의 모든 어휘 연결
  - GAT에 입력해 현재 mention encoding 업데이트



# Improving Graph-based RE

- Anaphoric graph
  - Mention과 pronoun 사이의 연관성 계산을 위한 그래프
  - Mention과 pronoun 사이의 관계에 따라 3가지 edge 타입 설계
  - 각 edge의 특징에 따라 다른 방식의 graph encoding 수행



# Experiments

---

- Datasets
  - DocRED
    - 문서 수준 관계 추출 말뭉치
    - 개체 타입: 6개, 관계 타입: 96개

Name	Value	
	Max	Avg
# of words	511	197.7
# of sentences	25	7.9
# of entities	41	19.5
# of triples	70	12.5



# Experiments

---

- 성능 비교 및 결과 분석
  - 그래프 기반 관계 추출 모델 중 가장 높은 성능
  - 부가적인 기술 적용으로 성능 고도화 가능성

Model	Dev		Test	
	Ign F1	F1	Ign F1	F1
BERT <sub>base</sub>	-	54.16	-	53.20
SSAN	57.03	59.19	55.84	58.16
ATLOP	59.22	61.09	59.31	61.30
AFL	<b>60.08</b>	62.03	<b>60.04</b>	<b>62.08</b>
LSR	52.30	59.00	56.97	59.05
GCGCN	55.43	57.35	54.53	56.67
GAIN	59.14	61.22	59.00	61.24
DRE	59.33	61.39	59.15	61.37
<b>KUNLP</b>	59.98	<b>62.05</b>	59.88	62.07



# End-to-End Relation Extraction

---

- Relation Extraction Task
  - 개체명 인식 결과(NE boundary & category)가 주어짐
  - 개체명 범주를 관계 추출에 활용
- End-to-End Relation Extraction Task
  - 언어처리가 되지 않은 문장이 주어짐
  - 주어진 문장으로부터 다양한 범주의 관계 추출 대상을 선별하고 그들 사이의 관계를 분류 → 파이프라인 구조 주로 사용!
  - 세분류 개체명 인식 또는 중첩 개체명 인식 모델 필요

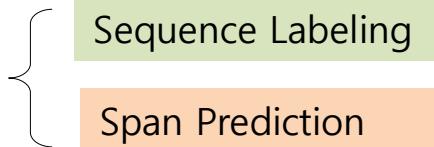


# Leaderboard on ACE 2005 (with Paper)

<https://paperswithcode.com>

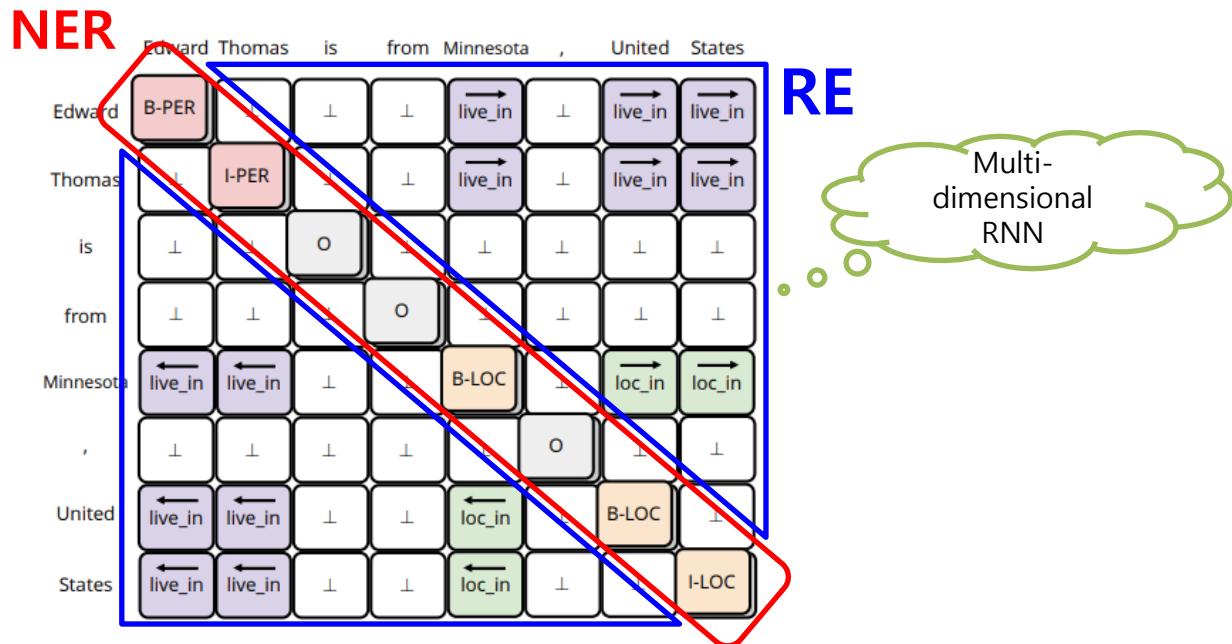
Rank	Model	RE Micro F1	RE+ Micro F1	NER Micro F1	Sentence Encoder	Relation classification F1	Extra Training Data	Paper	Code	Result	Year
1	Table-Sequence	67.6	64.3	89.5	ALBERT		×	Two are Better than One: Joint Entity and Relation Extraction with Table-Sequence Encoders			2020
2	MRC4ERE++		62.1	85.5	BERT base		×	Asking Effective and Diverse Questions: A Machine Reading Comprehension based Framework for Joint Entity-Relation Extraction			2020
3	Multi-turn QA		60.2	84.8	BERT base		×	Entity-Relation Extraction as Multi-Turn Question Answering			2019

Key Approaches



# Sequence Labeling

- Table sequence
  - 개체명 인식과 관계 추출을 위한 테이블 구조 제안
  - 개체명 인식은 sequence labeling, 관계 추출은 table filling으로 간주



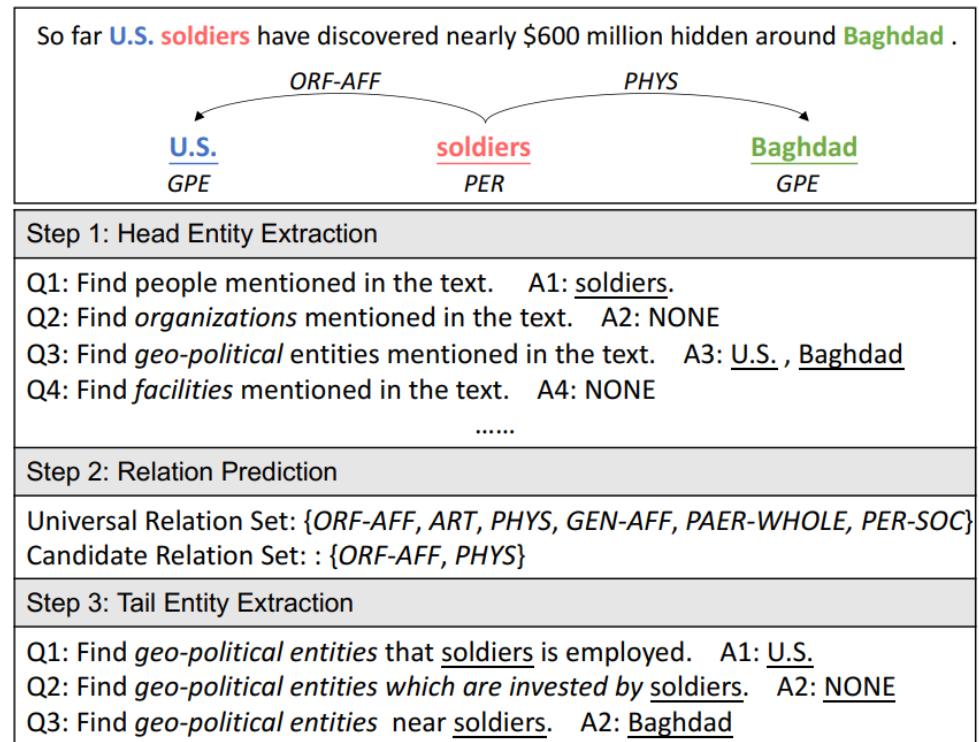
출처: Wang et al., Two are Better than One: Joint Entity and Relation Extraction with Table-Sequence Encoders, EMNLP 2020



# Span Prediction

- MRC4ERE

- 개체, 관계에 따른 질의 템플릿 구축
- 개체 관련 질의를 통해 개체 우선 추출
- 추출된 개체와 연관된 관계 질의를 통해 트리플 추출



출처: Zhao et al., Asking Effective and Diverse Questions: A Machine Reading Comprehension based Framework for Joint Entity-Relation Extraction, EMNLP 2020



---

# Text Summarization

---

# Extractive vs. Abstractive

---

- 추출 요약
  - 핵심 문장 선별 태스크(문장 분류 문제의 범주)
  - 단점: 인간 수준의 일반화/추상화된 요약 제공 불가
  - 최근 동향: 단어와 문장 중요도를 동시에 반영하기 위한 다양한 방법들이 연구
- 추상 요약
  - 인간의 요약 방법을 모사
  - 단점: 사실이 아닌 정보를 출력하거나 비문법적 문장을 생성
  - 최근 동향: 다양한 심층 신경망을 통해 비문 생성 가능성을 줄이고, 추상화 수준을 끌어올리기 위한 다양한 방법들이 연구



# Summarization Corpus

---

- CNN/Daily Mail 말뭉치
  - 영어권 자동 문서 요약 성능 평가 시 가장 많이 사용되는 말뭉치
  - 신문 기사 원본과 사람이 작성한 요약 문서의 쌍으로 구성
  - 학습 데이터: 286,817개
  - 개발 데이터: 13,368개
  - 평가 데이터: 11,487개
  - 원본 문서의 평균 길이: 76.6 단어(29.74 문장)
  - 요약 문서의 평균 길이: 53 단어(3.72 문장)



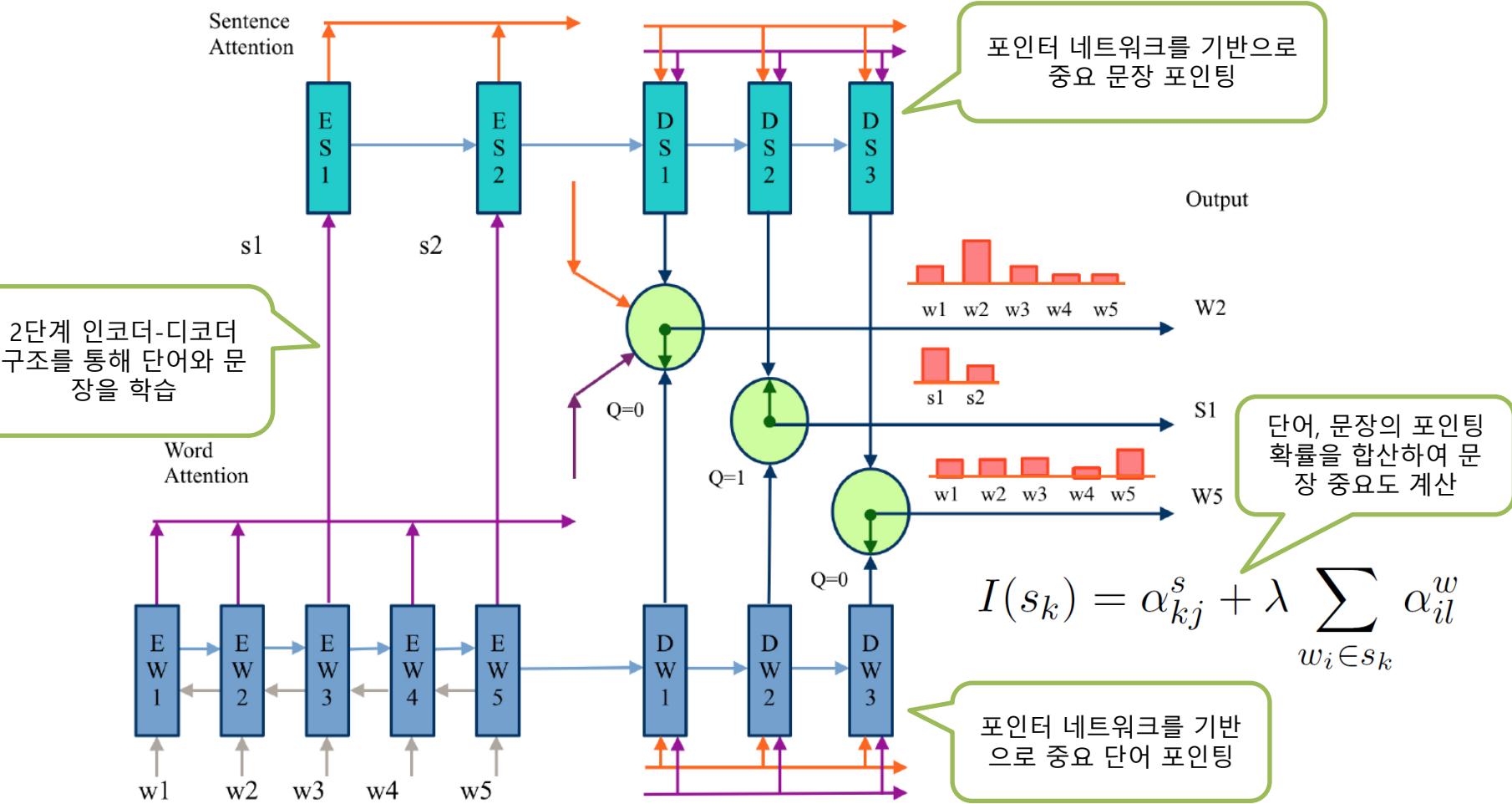
# Evaluation Measure

---

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation)
  - 대표적인 재현율 중심의 정량적 자동 평가 척도
  - 단순 단어 일치도에 따른 평가 → 문법적 오류나 의미적 유사성을 반영하지 못함
  - 정성적 평가 또는 Embedding 유사성 비교 등을 통해 보완 필요



# Extractive Model: SWAP-NET



# Experiments

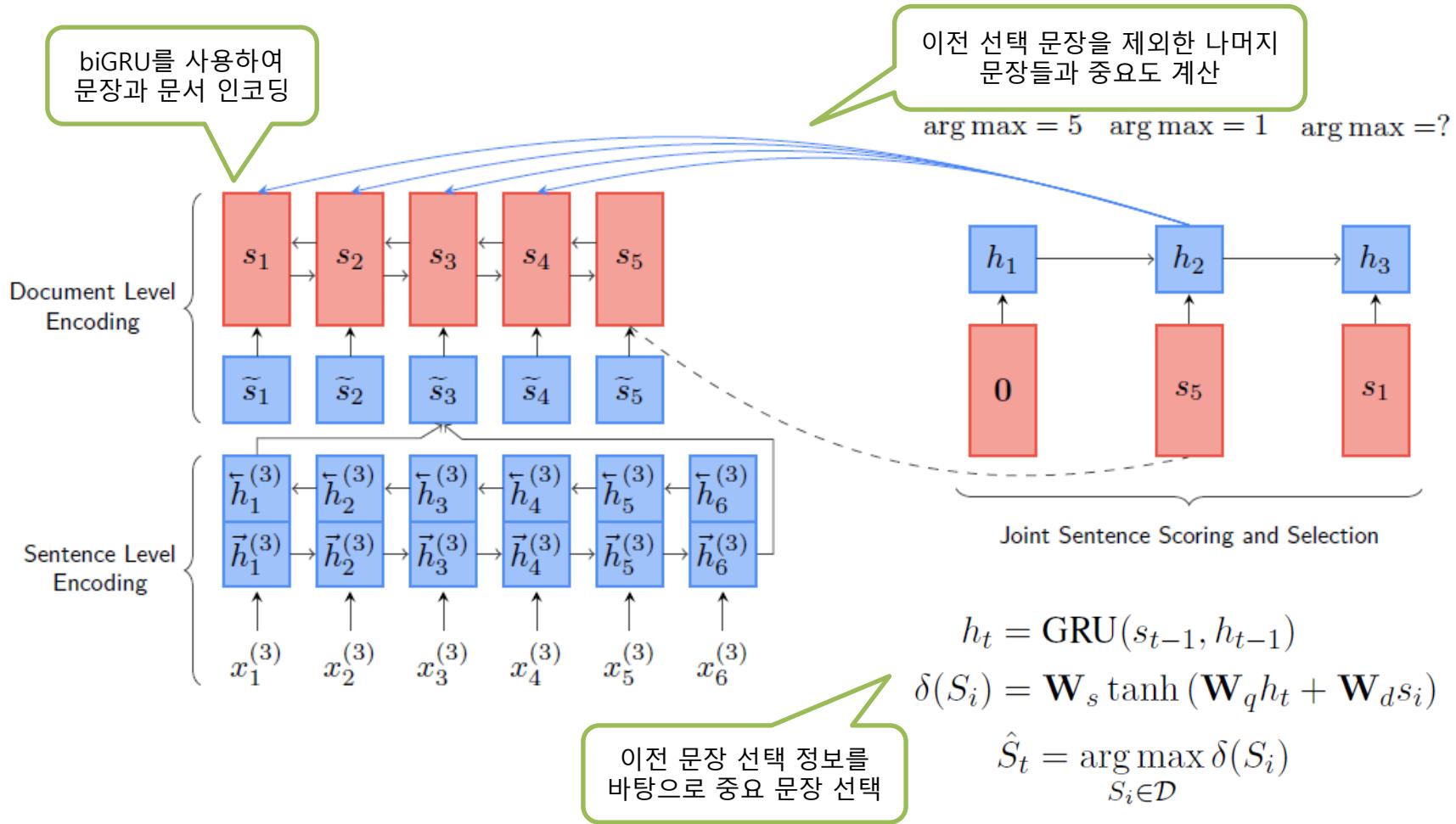
- CNN/Daily Mail 데이터를 이용한 평가 결과

Models	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	39.2	15.7	35.5
SummaRuNNer (Nallapati 외, 2017)	39.6	16.2	35.3
SWAP-NET	<b>41.6</b>	<b>18.3</b>	<b>37.7</b>

- 기존 대표적 추출 요약 방법보다 약 2%p 우수한 성능을 보임
- SWAP-NET의 추출 문장은 98%로 핵심어를 포함 (LEAD-3 : 92.2%)
- 문장과 단어의 중요도를 모두 고려하는 것이 타당하다는 결과



# Extractive Model: NeuSum

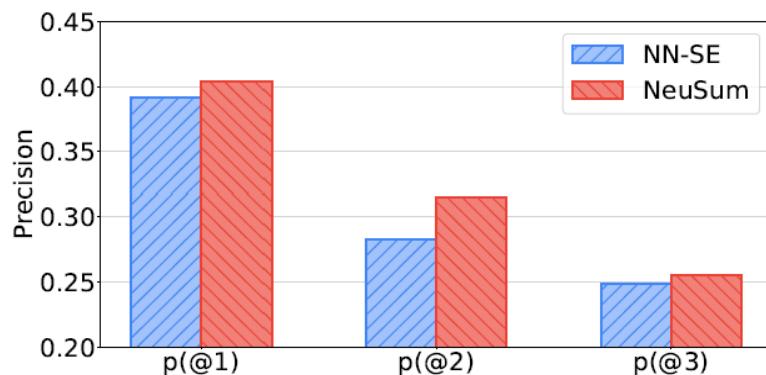


# Experiments

- CNN/Daily Mail 데이터를 이용한 평가 결과

Models	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	39.2	15.7	35.5
SWAP-NET	41.6	18.3	37.7
NeuSum	<b>41.6</b>	<b>19.0</b>	<b>38.0</b>

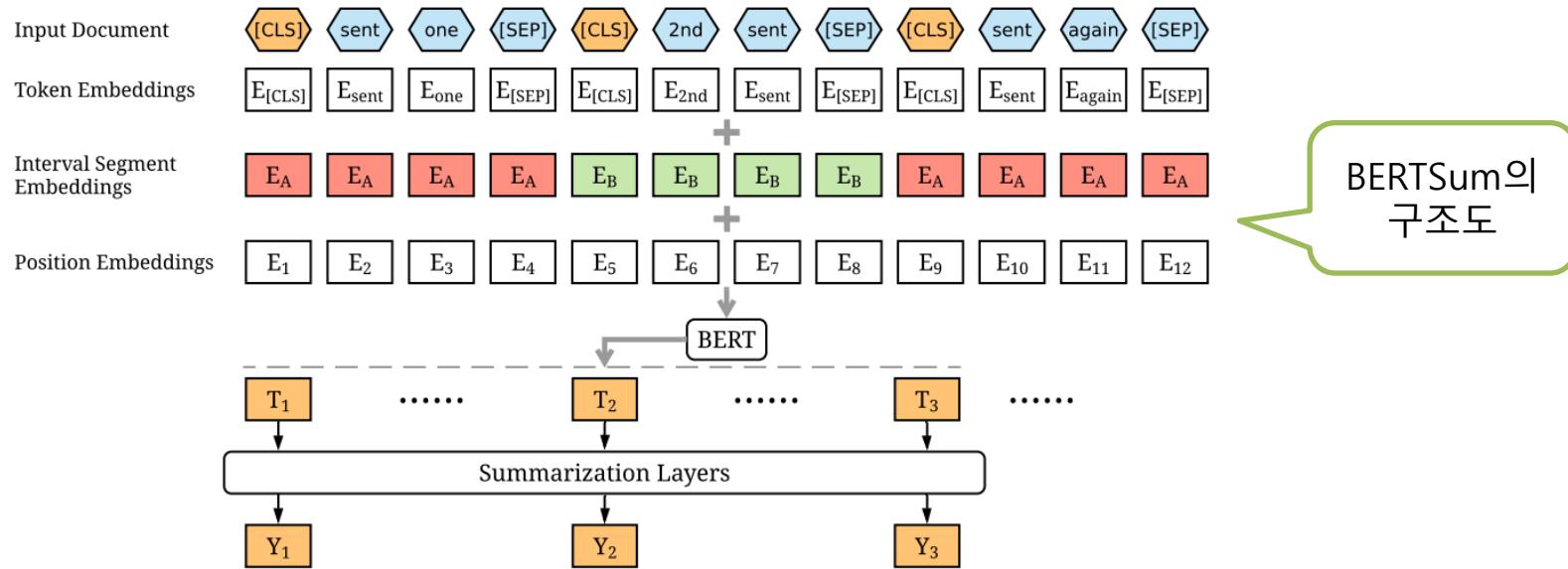
- 선택 문장 수에 따른 성능 비교



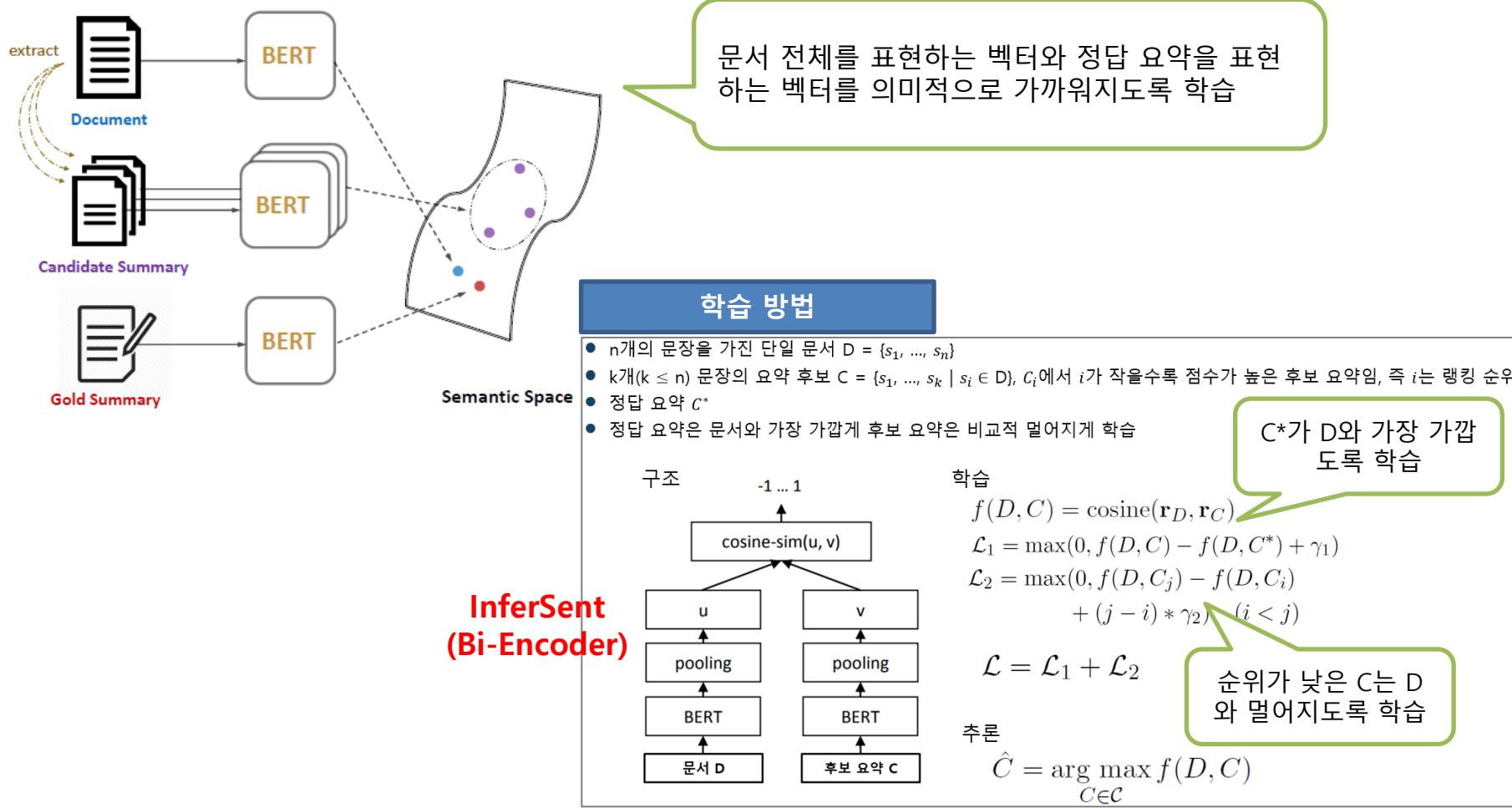
P(@2)에서 기존 모델과 NeuSum의 큰 편 차를 통해, 문장 선택 이력을 반영하는 것 이 성능 향상에 도움되는 것을 확인

# Extractive Model: MatchSum

- 기존 추출 요약 모델의 문제점
  - 각 문장이 중요한 문장인지 아닌지 독립적으로 결정하기 때문에 중복성 증가 → 비슷한 문장을 여럿 추출
  - 문서의 전체적인 의미를 고려하지 않고 일반화된 문장을 선택하는 경향이 있음



# Extractive Model: MatchSum



# Experiments

- CNN/Daily Mail 데이터를 이용한 평가 결과

Model	R-1	R-2	R-L
LEAD	40.43	17.62	36.67
ORACLE	52.59	31.23	48.87
MATCH-ORACLE	51.08	26.94	47.22
BANDITSUM (Dong et al., 2018)	41.50	18.70	37.60
NEUSUM (Zhou et al., 2018)	41.59	19.01	37.98
JECS (Xu and Durrett, 2019)	41.70	18.50	37.90
HiBERT (Zhang et al., 2019b)	42.37	19.95	38.83
PNBERT (Zhong et al., 2019a)	42.39	19.51	38.69
PNBERT + RL	42.69	19.60	38.85
BERTEXT <sup>†</sup> (Bae et al., 2019)	42.29	19.38	38.63
BERTTEXT <sup>†</sup> + RL	42.76	19.87	39.11
BERTTEXT (Liu, 2019)	42.57	19.96	39.04
BERTTEXT + Tri-Blocking	43.23	20.22	39.60
BERTSUM* (Liu and Lapata, 2019)	43.85	20.34	39.90
BERTTEXT (Ours)	42.73	20.13	39.20
BERTTEXT + Tri-Blocking (Ours)	43.18	20.16	39.56
MATCHSUM (BERT-base)	44.22	20.62	40.38
MATCHSUM (RoBERTa-base)	<b>44.41</b>	<b>20.86</b>	<b>40.55</b>

현재 SOTA!



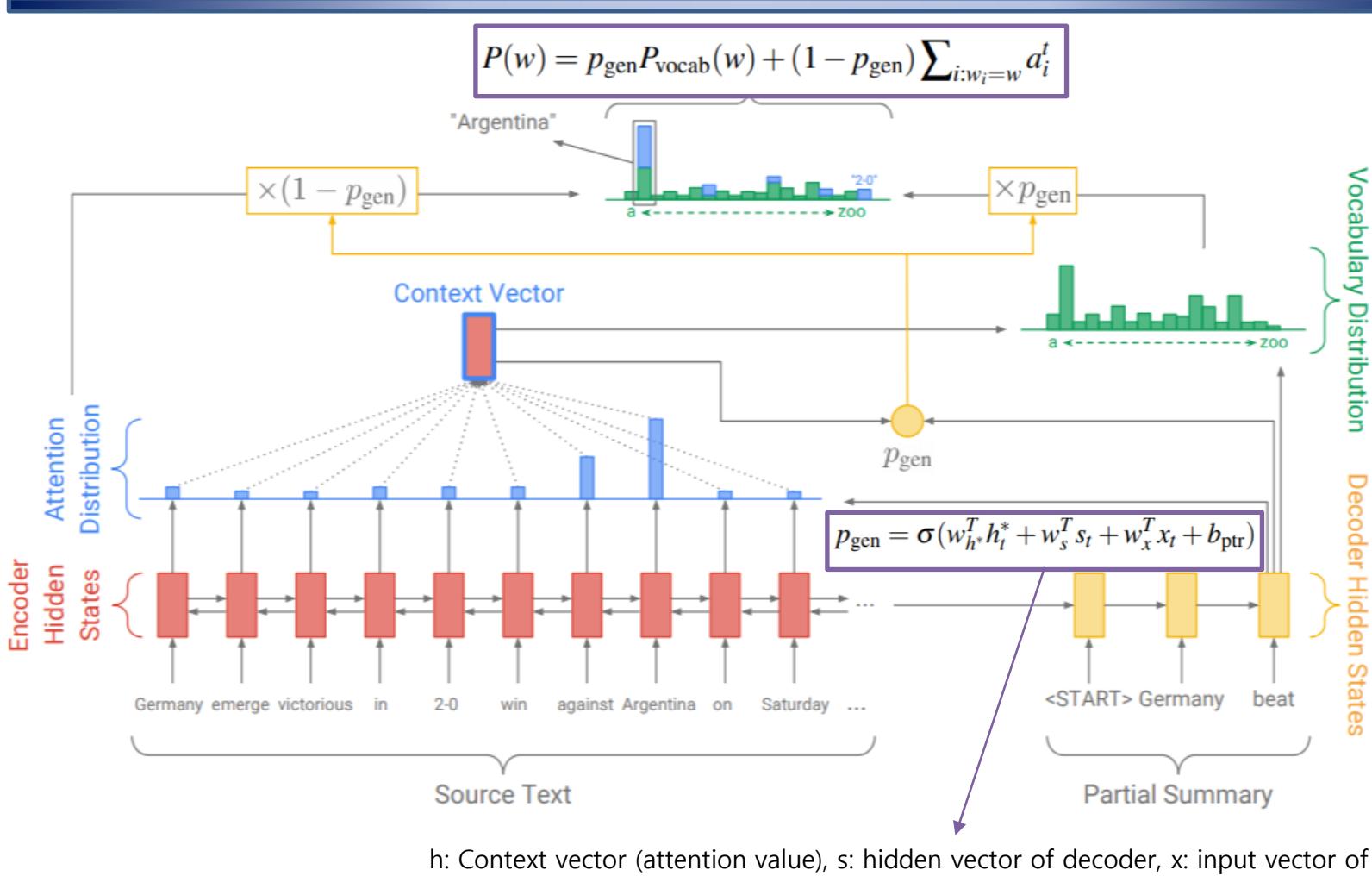
# Abstractive Model: Pointer-Generator

---

- Pointer-Generator
  - 반드시 기억해야 할 대표적인 추상 요약 모델
  - 입력 문장의 중요 단어를 복사하는 방법을 통해 추상 요약 보완
  - 추상 요약 과정에서 발생하는 유사 의미 단어 오생성 문제 해결



# Pointer Generator



# Coverage Mechanism (in Pointer Generator)

Bahdanau's Attention

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}})$$

$$a^t = \text{softmax}(e^t)$$



$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}})$$

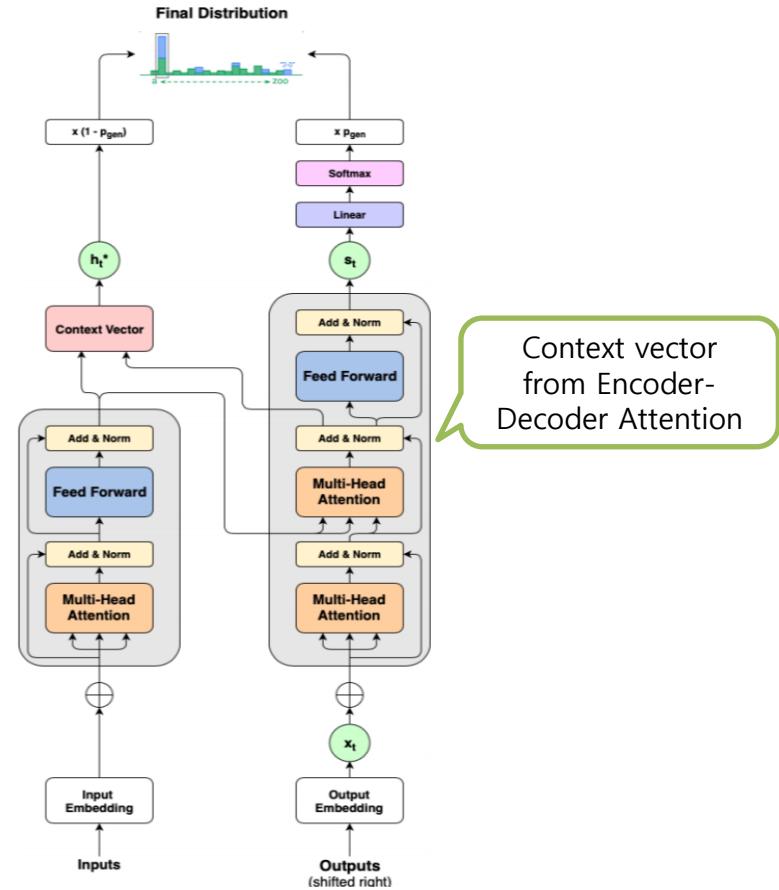
**Coverage vector**  $c^t = \sum_{t'=0}^{t-1} a^{t'}$

$$\text{loss}_t = -\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t)$$

**Coverage loss**

(두 벡터에서 공유되는 균일  
단어 분포에 대한 패널티)

Pointer Generator in Transformer



# Experiments

- CNN/Daily Mail 데이터를 이용한 평가 결과

Models	ROUGE-1	ROUGE-2	ROUGE-L
LEAD-3	40.34	17.70	36.57
Seq2Seq+Attention (150k vocab)	30.49	11.17	28.08
Seq2Seq+Attention (50k vocab)	31.33	11.81	28.83
Pointer-generator	36.44	15.66	33.42
Pointer-generator + Coverage	<b>39.53</b>	<b>17.28</b>	<b>36.38</b>

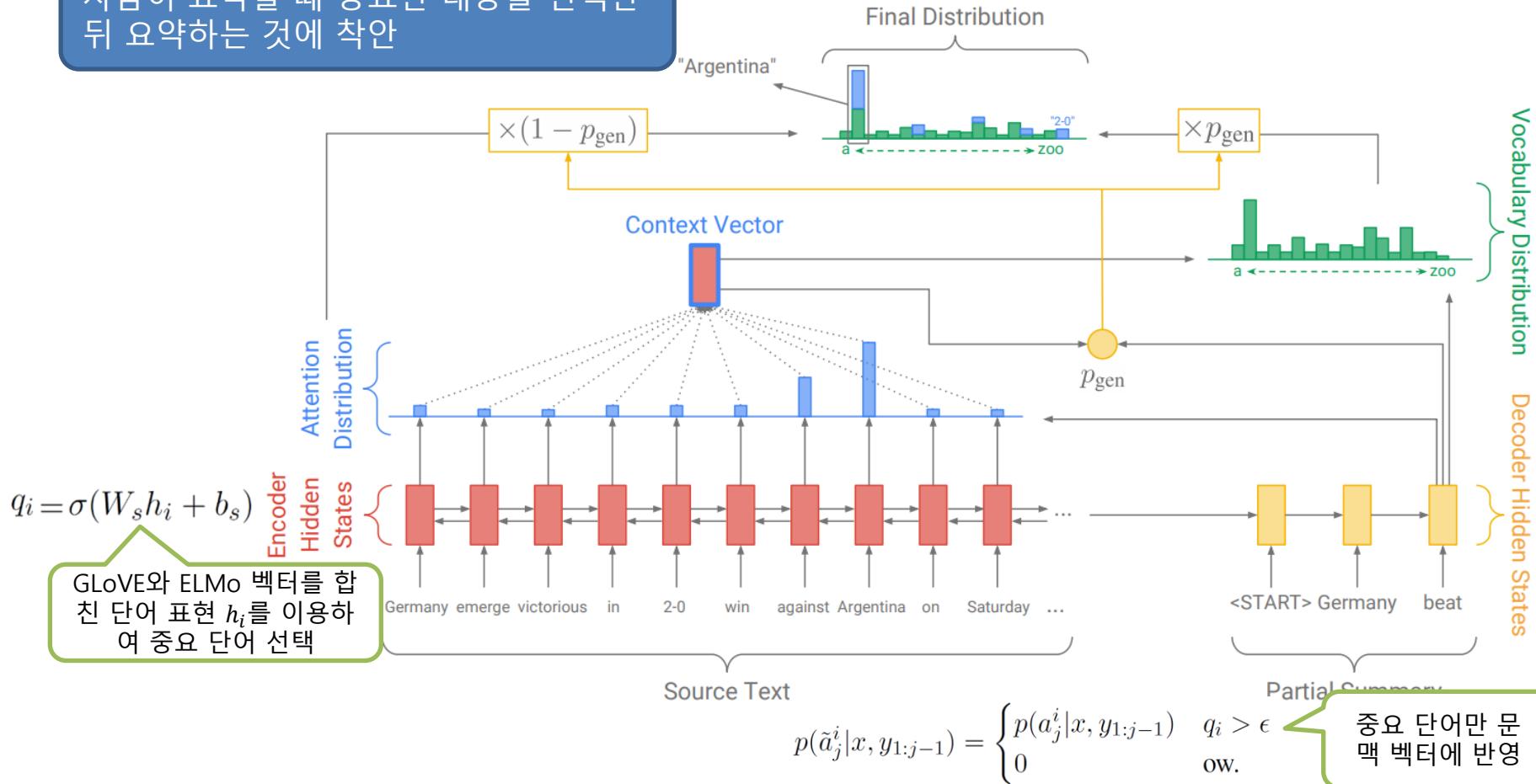
- 추출 요약 방법과 추상 요약 방법을 성공적으로 결합한 결과
- 평가 결과를 통해 추상 요약을 대표하는 모델로 자리매김함

출처: See A., Liu P. J. & Manning C. D., "Get to the point: Summarization with pointer-generator networks," ACL 2017



# Bottom-Up Abstractive Summarization

사람이 요약할 때 중요한 내용을 선택한 뒤 요약하는 것에 착안



# Experiments

- CNN/Daily Mail 데이터를 이용한 평가 결과

Models	ROUGE-1	ROUGE-2	ROUGE-L
Pointer-generator (See 외, 2017))	36.44	15.66	33.42
Pointer-generator + Coverage	39.53	17.28	36.38
Bottom-Up Summarization	<b>41.22</b>	<b>18.68</b>	<b>38.34</b>

- Pointer-generator 모델보다 약 2%p 향상된 성능 기록
- 중요한 내용을 먼저 선택하는 것이 요약 결과에 큰 영향을 미치는 것을 확인

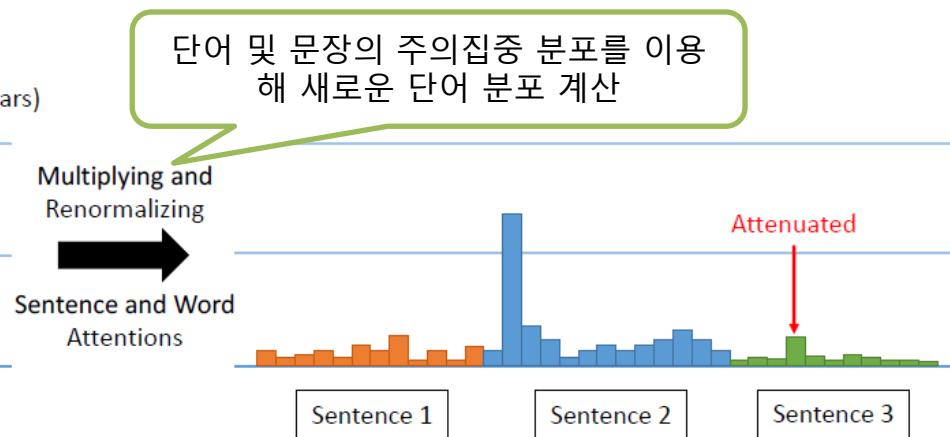
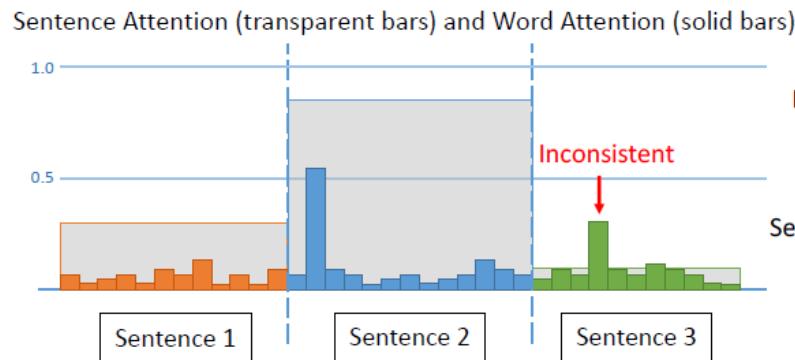
출처: Gehrmann S., Deng Y. & Rush A., "Bottom-Up Abstractive Summarization," EMNLP 2018



# A Unified Model for Extractive and Abstractive Summarization

요약과 연관 없는 단어를 생성하는 추상 요약의 문제점 보완  
요약 결과의 문장 연결이 매끄럽지 못한 추출 요약의 문제점 보완

SWAP-Net과 유사!



$$\hat{\alpha}_m^t = \frac{\alpha_m^t \times \beta_{n(m)}}{\sum_m \alpha_m^t \times \beta_{n(m)}}$$

단어 주의집중 분포와  
문장 주의집중 분포를 곱한 후 정규화

$$L_{inc} = -\frac{1}{T} \sum_{t=1}^T \log\left(\frac{1}{|\mathcal{K}|} \sum_{m \in \mathcal{K}} \alpha_m^t \times \beta_{n(m)}\right)$$

단어 및 문장의 주의집중 분포 일관성  
을 위한 손실함수 제안



# Experiments

---

- CNN/Daily Mail 데이터를 이용한 평가 결과

Models	ROUGE-1	ROUGE-2	ROUGE-L
HierAttn(Nallapati 외, 2017)	32.75	12.21	29.01
Pointer-generator(See 외, 2017)	39.53	17.28	36.38
Hsu 외, 2018	<b>40.68</b>	<b>17.97</b>	<b>37.13</b>

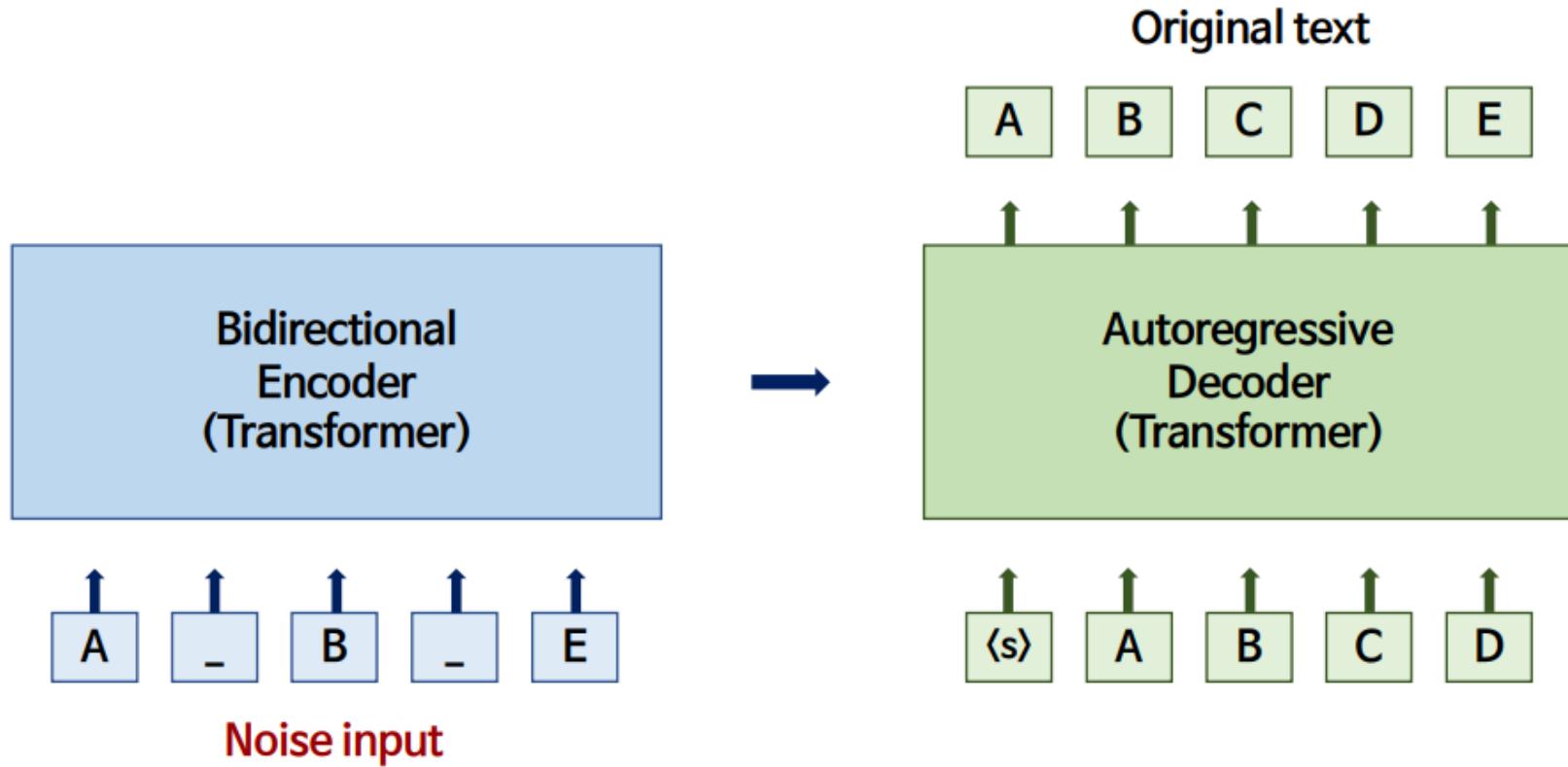
- 단일 추상, 추출모델보다 통합 모델이 더 높은 성능을 보임
- 새로운 단어 주의집중 분포가 요약 생성에 있어 이점이 있음

출처: Hsu W. T., Lin C. K., Lee M. Y., Min K., Tang J. & Sun M., "A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss," *arXiv preprint arXiv:1805.06266*, 2018

---



# BART: Bidirectional and Auto-Regressive Transformer



출처: 고려대 DSBA 연구실 이유경님 발표자료



# Experiments

Models	ROUGE-1	ROUGE-2	ROUGE-L
Pointer-generator (See 외, 2017))	36.44	15.66	33.42
Pointer-generator + Coverage	39.53	17.28	36.38
Bottom-Up Summarization	<b>41.22</b>	<b>18.68</b>	<b>38.34</b>

 +3%p

	CNN/DailyMail			XSum		
	R1	R2	RL	R1	R2	RL
Lead-3	40.42	17.62	36.67	16.30	1.60	11.95
PTGEN (See et al., 2017)	36.44	15.66	33.42	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38	28.10	8.02	21.72
UniLM	43.33	20.21	40.51	-	-	-
BERTSUMABS (Liu & Lapata, 2019)	41.72	19.39	38.76	38.76	16.33	31.15
BERTSUMEXTABS (Liu & Lapata, 2019)	42.13	19.60	39.18	38.81	16.50	31.27
BART	<b>44.16</b>	<b>21.28</b>	<b>40.90</b>	<b>45.14</b>	<b>22.27</b>	<b>37.25</b>



# Summarization in KUNLP

- 최근 기계 요약 연구는 문서로부터 일반적으로 중요한 내용으로 요약하는 문제에 집중
- 하지만, 사용자들은 문서에서 획득하고 싶은 정보의 니즈가 각기 다름

## 일반적 기계 문서 요약 예시

2년째 이어진 거리두기... 국민 5명 중 1명 '코로나 블루'

신종 코로나바이러스감염증(코로나19) 사태가 장기화하면서 국민 5명 중 1명꼴로 심각한 우울을 겪고 있는 것으로 나타났다.

정부는 사회적 거리두기 등 방역규제 완화를 고려하지만, 변이 바이러스 오미크론 확산을 앞두고 고심하고 있다.

정부는 이날 일상회복지원위원회 방역의료분과 회의, 12일 일상회복위 전체 회의, 14일 중앙재난안전대책본부 검토를 거쳐 거리두기 조정 여부를 발표하기로 했다.

홍정의 접종관리팀장은 "오미크론용 백신이 나와도 접종 여부는 지금으로서는 예단하기 어렵다"고 말했다.

기계 문서 요약 결과  
신종 코로나바이러스감염증(코로나19) 사태가 장기화하면서 국민 5명 중 1명꼴로 심각한 우울을 겪고 있는 것으로 나타났다.

## 질의 기반 문서 요약 예시

2년째 이어진 거리두기... 국민 5명 중 1명 '코로나 블루'

신종 코로나바이러스감염증(코로나19) 사태가 장기화하면서 국민 5명 중 1명꼴로 심각한 우울을 겪고 있는 것으로 나타났다.

정부는 사회적 거리두기 등 방역규제 완화를 고려하지만, 변이 바이러스 오미크론 확산을 앞두고 고심하고 있다.

정부는 이날 일상회복지원위원회 방역의료분과 회의, 12일 일상회복위 전체 회의, 14일 중앙재난안전대책본부 검토를 거쳐 거리두기 조정 여부를 발표하기로 했다.

홍정의 접종관리팀장은 "오미크론용 백신이 나와도 접종 여부는 지금으로서는 예단하기 어렵다"고 말했다.

### Q1. 코로나 블루와 관련된 내용

신종 코로나바이러스감염증(코로나19) 사태가 장기화하면서 국민 5명 중 1명꼴로 심각한 우울을 겪고 있는 것으로 나타났다.

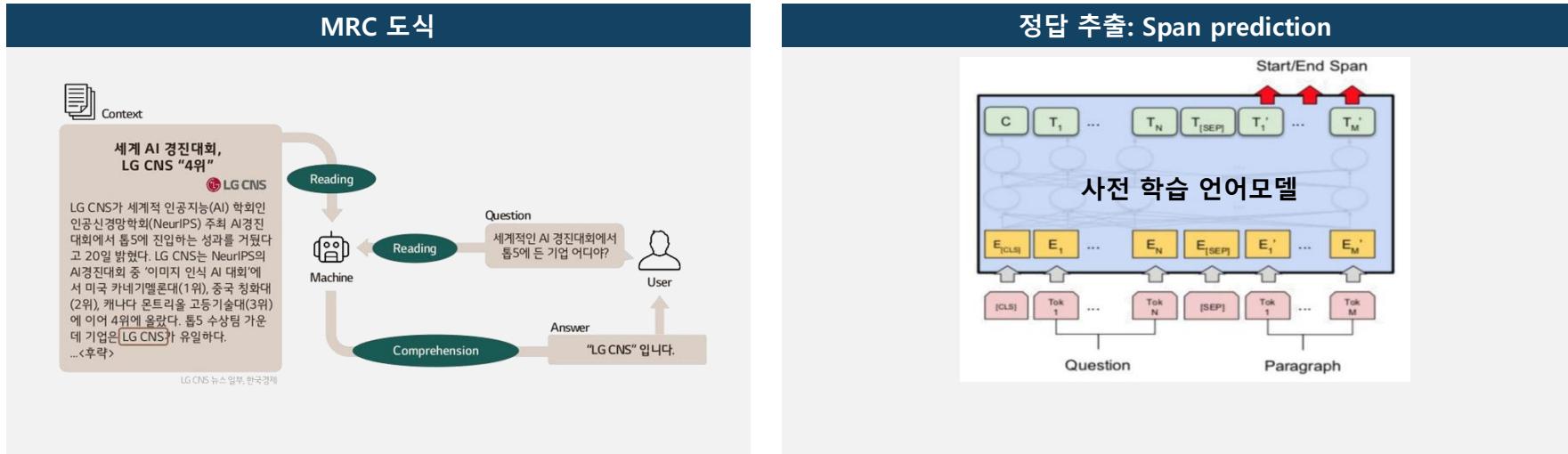
### Q2. 사회적 거리두기에 대한 내용

정부는 사회적 거리두기 등 방역규제 완화를 고려하지만, 변이 바이러스 오미크론 확산을 앞두고 고심하고 있다. 정부는 이날 일상회복지원위원회 방역의료분과 회의, 12일 일상회복위 전체 회의, 14일 중앙재난안전대책본부 검토를 거쳐 거리두기 조정 여부를 발표하기로 했다.



# MRC for Query-based Summarization

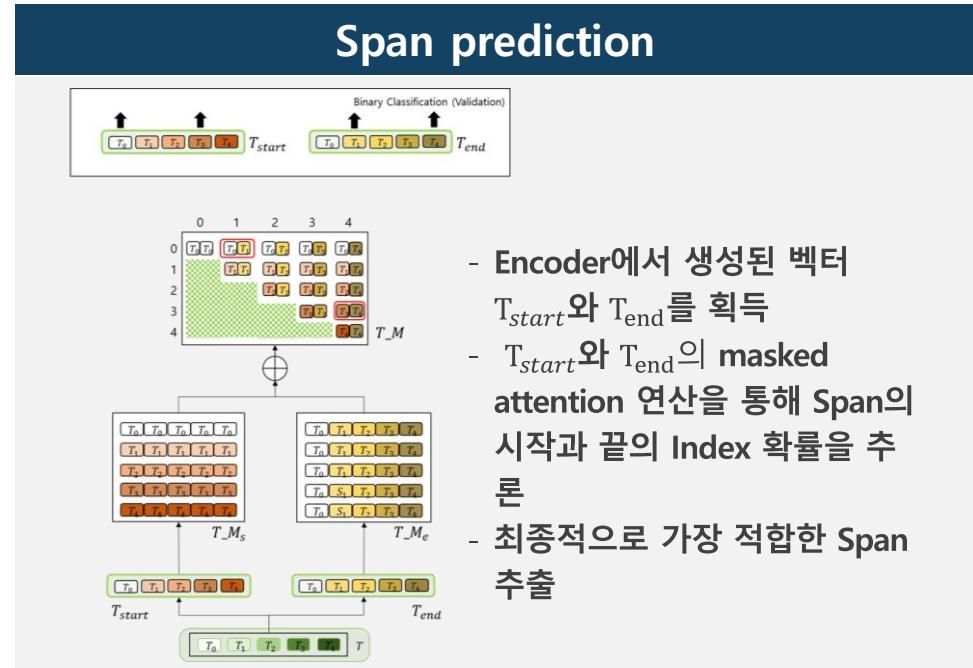
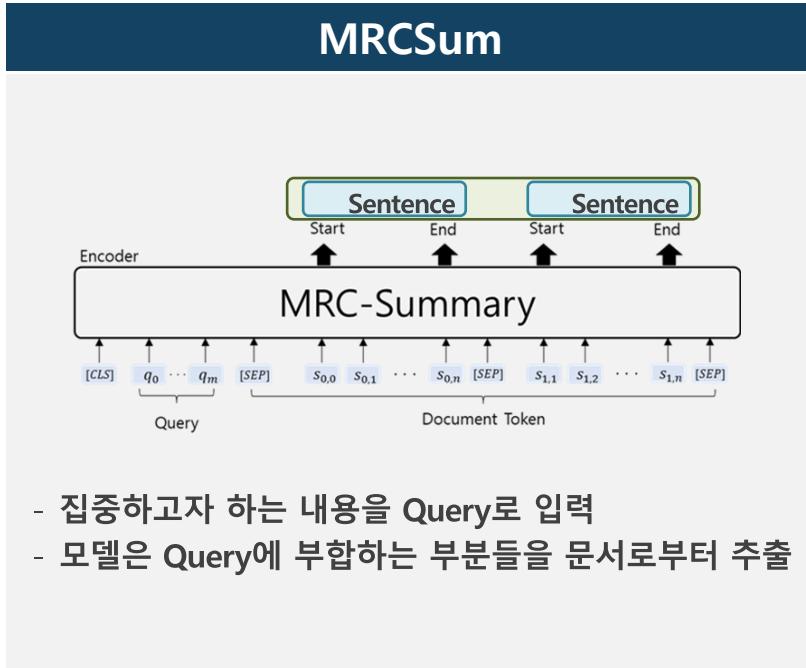
- 주어진 문서에서 사용자가 던진 질문(Query)에 대한 답을 문서(Context)로부터 추출하여 보여주는 기술
- 모델은 질문과 문서의 의미적인 상호작용(Interaction)을 통해 질문의 의도를 파악하고, 알맞은 답을 탐색



- 최근 MRC는 사전 학습 언어 모델을 이용
- 정답의 시작(Start)과 끝(End)의 위치를 추출하여 정답의 범위(Span)를 알 수 있음



# MRCSum



# Query for MRCSum

- **기존 Query**
  - 영어권에서는 Query를 단순히 "What is summary?"와 같이 간단히 구성한 것도 있었음
  - MRC의 장점인 Query와 문서의 특정 내용과의 연관성이 반영되는 장점을 잊게 됨
- **MRCSum의 Query**

## Title-Query

- 기사의 제목 Query로 입력
- 모든 요약 데이터 셋에는 기사 제목이 존재하고, 이를 이용하여 기초적인 질의 기반 문서 요약 연구의 베이스라인을 설정
- 사용 한국어 사전 학습 언어 모델은 RoBERTa 사용
- 그 외의 일반적 질의를 이용한 실험도 동시 수행 (예: “요약해 줘”, “중요한 문장이 뭐야?” 등)

문제 1.  
기사의 제목이 없는  
문서의 경우?



기사에는 카테고리가 존재하며,  
그에 맞춰 작성됨

## Topic-Query

- 문서의 Topic을 Query로 입력
- 뉴스 기사는 일반적으로 카테고리가 존재하며 토픽이 이를 대표할 수 있을 것으로 기대
- LDA를 학습시켜 Topic에 집중한 요약 모델 학습

Topic 1	말, 사람, 영화, 작품, 한국, 책, 서울, 공연, 뒤, 작가
Topic 2	국내, 시장, 말, 한국, 제품, 기업, 개발, 기술, 지난해, 세계
Topic 3	정부, 금융, 말, 경제, 기업, 투자, 지난해, 경우, 기준, 올해
Topic 4	말, 환자, 병원, 사람, 멕, 교수, 결과, 연구, 건강, 경우
Topic 5	중국, 북한, 미국, 정부, 일본, 대통령, 말, 한국, 국가, 문제
...	...

문제 2.  
사용자가 원하는  
정보는 키워드에  
포함되지 않는가?



기사 내에서 중요하  
다고 판단되는 키  
워드들을  
사용

## Keyword-Query

- 문서 내에서 추출한 Keyword를 Query로 입력
- KeyBert, TextRank, TF-IDF를 이용해 Entity추출

Title	KeyBERT	TextRank
내년 폐기물 자원화 사업에 762억 원 쏟기로	폐기물, 매립, 환경기술, 환경, 환경부	폐기물, 에너지, 매립, 시설, 예산
달아오른 부동산 경매... 낙찰가는 '싸늘'	경매, 옥션, 낙찰, 법정, 부동산	경매, 법정, 낙찰, 사람, 가격
“융합형 창의 인재 양성... 산업의 밑거름 만들겠다”	한국과학창의재단, 과학기술, 광주과학기술원, 과학, 울산과학기술원	과학기술, 이사장, 박, 융합, 문화



# Experiments in English

- CNN/Daily Mail 데이터를 이용한 평가 결과

Model	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	40.43	17.62	36.67
ORACLE	52.59	31.23	48.87
BERTSum(base)	43.25	20.24	39.63
BERTSum(large)	43.85	20.34	39.90
MatchSum(BERT-base)	44.22	20.62	40.38
MatchSum(RoBERTa-base)	44.41	20.86	40.55
MRCSum(BERT-base)	44.77	21.01	40.63
MRCSum(RoBERTa-base)	<b>44.81</b>	<b>21.07</b>	<b>40.66</b>

- MRCSum(RoBERTa-base)이 기존 SOTA인 MatchSum을 뛰어넘는 결과를 보여 줌



# Experiments in Korean

- NIA AI hub 기사 요약 말뭉치(30만 건)을 이용한 평가

	Model	ROUGE-1	ROUGE-2	ROUGE-L
Upper bound	Title	64.82	<b>49.27</b>	<b>69.21</b>
Base Line	MatchSum	61.37	46.91	61.32
Topic-Query	LDA-Topic 1	57.84	44.06	59.37
Keyword-Query	LDA-Topic 1 +Topic word*	62.20	47.66	65.93
	TFIDF	65.43	45.64	67.43
	KeyBERT	<b>66.28</b>	46.16	66.54

\* Topic word는 LDA 토픽 단어 셋에서 포함되는 단어를 추출

- MRCSum(KeyBERT) 모델이 ROUGE-1 스코어가 가장 높지만 2와 L은 MRCSum>Title) 모델이 더 높은 결과를 나타냄



# Relation Extraction

---

Practical Exercise

# 실습

예제 코드 다운로드:  
<https://github.com/KUNLP/Lecture>

- 대용량 언어 모델인 ELECTRA를 이용하여 문장 수준 관계 추출 (Sentence-level Relation Extraction) 시스템을 구현 하시오.

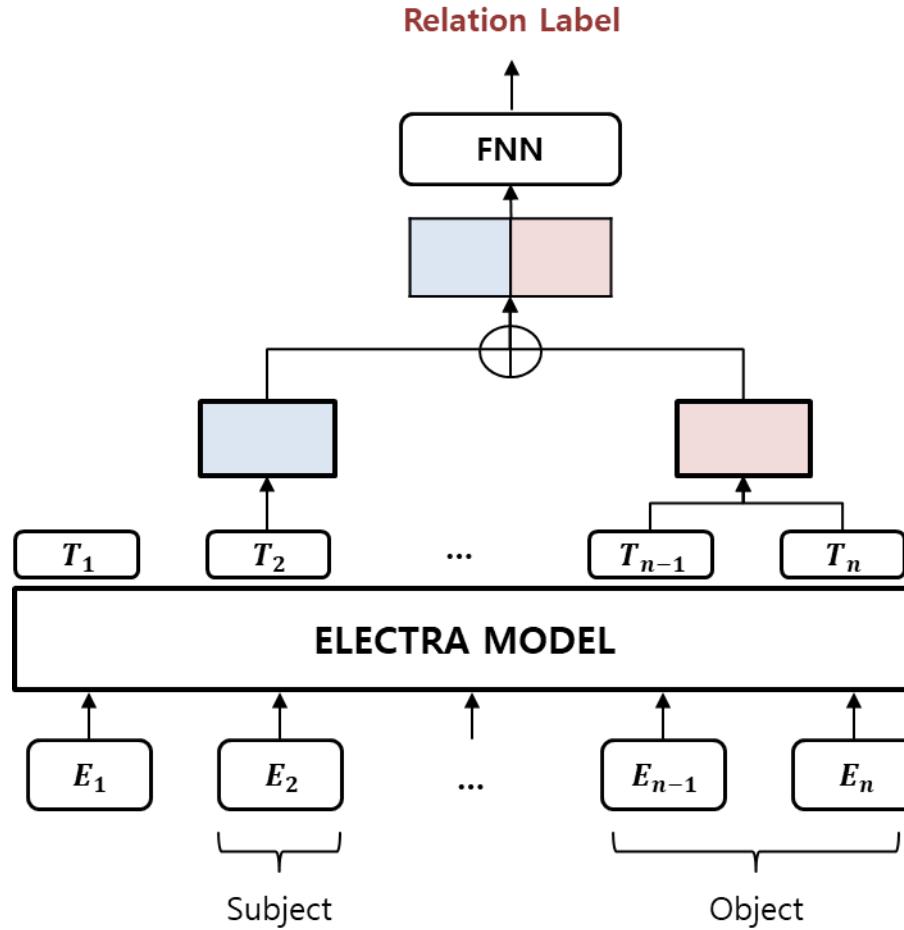
이건희는 대한민국 기업인 삼성의 회장이다. 또한, 삼성  
이재용 부회장의 아버지이다.

개체1	개체2	관계
이건희	삼성	Owner
삼성	대한민국	Located
이재용	삼성	Employee
이건희	이재용	Family

```
{  
    "guid": "klue-re-v1_dev_00000",  
    "sentence": "20대 남성 A(26)씨가 아버지 치료비를 위해 B(30)씨가 모아둔 돈을 험쳐 인터넷 방송 BJ에게 '별풍선'으로 쓴 사실이 알려졌다.",  
    "subject_entity": {"word": "A", "start_idx": 7, "end_idx": 7, "type": "PER"},  
    "object_entity": {"word": "30", "start_idx": 29, "end_idx": 30, "type": "NOH"},  
    "label": "no_relation",  
    "source": "wikitree"  
},
```



# 실습



# 실습

## 모델 설계

```
class ElectraForRelationExtraction(ElectraPreTrainedModel):
    def __init__(self, config):
        super(ElectraForRelationExtraction, self).__init__(config)

        # 분류해야 할 라벨 수
        self.num_labels = config.num_labels
        # ELECTRA 모델 선언
        self.electra = ElectraModel(config)
        # 최종 출력
        self.re_outputs = nn.Linear(config.hidden_size*2, config.num_labels)

    def forward(
            self, input_ids=None, attention_mask=None, subj_masks=None, obj_masks=None, labels=None
    ):
        outputs = self.electra(
            input_ids=input_ids,
            attention_mask=attention_mask
        )

        sequence_outputs = outputs[0]
        # (batch, seq_len, hidden)

        subj_outputs = sequence_outputs*subj_masks.unsqueeze(-1)
        # (batch, seq_len, hidden)
        obj_outputs = sequence_outputs*obj_masks.unsqueeze(-1)
        # (batch, seq_len, hidden)

        subj_outputs = torch.sum(subj_outputs, 1)
        # (batch, seq_len, hidden) => (batch, hidden)
        obj_outputs = torch.sum(obj_outputs, 1)
        # (batch, seq_len, hidden) => (batch, hidden)

        relation_outputs = torch.cat([subj_outputs, obj_outputs], -1)
        # (batch, hidden*2)

        relation_logits = self.re_outputs(relation_outputs)
        # (batch, hidden*2) => (batch, num_of_labels)

    return relation_logits
```



# 실습

## 데이터 읽기

```
def load_data(tokenizer, f_name):
    # 입력 문장 index, attention mask, subj mask, obj mask, label 저장을 위한 리스트
    input_ids = []
    attention_masks = []
    subj_masks = []
    obj_masks = []
    labels = []

    with open(f_name, 'r', encoding='utf8') as infile:
        data_list = json.load(infile)

    for data in tqdm(data_list):
        sentence = data['sentence']

        subj = data['subject_entity']
        subject_text = subj['word']
        obj = data['object_entity']
        object_text = obj['word']

        relation_type = data['label']

        input_idx, attention_mask, subj_mask, obj_mask = \
            tokenize(tokenizer, sentence, subject_text, object_text)
        if relation_type not in label2idx.keys():
            label2idx[relation_type] = len(label2idx)
            idx2label[len(idx2label)] = relation_type
        relation = label2idx[relation_type]

        input_ids.append(input_idx)
        attention_masks.append(attention_mask)
        subj_masks.append(subj_mask)
        obj_masks.append(obj_mask)
        labels.append(relation)
    # tensor 형태로 변환
    input_ids = torch.tensor(input_ids, dtype=torch.long)
    attention_masks = torch.tensor(attention_masks, dtype=torch.long)
    subj_masks = torch.tensor(subj_masks, dtype=torch.long)
    obj_masks = torch.tensor(obj_masks, dtype=torch.long)
    labels = torch.tensor(labels, dtype=torch.long)

    return TensorDataset(input_ids, attention_masks, subj_masks, obj_masks, labels)
```



# 실습

## 가중치 초기화

```
def _create_model(config):
    print("Create Model with Pretrained Parameters...")
    electra_config = ElectraConfig.from_pretrained(
        config["init_weight"] if config["mode"] == "train" else os.path.join(config["output_dir"],
                                                                           "checkpoint-{}".format(config["checkpoint"])),
    )
    electra_tokenizer = ElectraTokenizer.from_pretrained(
        config["init_weight"] if config["mode"] == "train" else os.path.join(config["output_dir"],
                                                                           "checkpoint-{}".format(config["checkpoint"])),
        do_lower_case=config["do_lower_case"],
    )
    electra_config.num_labels = len(label2idx)
    model = ElectraForRelationExtraction.from_pretrained(
        config["init_weight"] if config["mode"] == "train" else os.path.join(config["output_dir"],
                                                                           "checkpoint-{}".format(config["checkpoint"])),
        config=electra_config
    )

    model.cuda()

    return config, model, electra_tokenizer
```



# 실습

## Main

```
if __name__ == "__main__":
    output_dir = os.path.join(root_dir, "output")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    config = {
        "mode": "test",
        "data_dir": root_dir,
        "output_dir": output_dir,
        "train_file": "klue-re-v1.1_train.json",
        "test_file": "klue-re-v1.1_dev.json",
        "init_weight": "monologg/koelectra-small-v3-discriminator",
        "do_lower_case": False,
        "checkpoint": 3000,
        "batch_size": 32,
        "num_epochs": 5,
        "learning_rate": 5e-5,
    }
    config, model, tokenizer = _create_model(config)

    if config["mode"] == 'train':
        train(config, model, tokenizer)
    elif config["mode"] == 'test':
        evaluate(config, model, tokenizer)
```

Create Model with Pretrained Parameters...
100%|██████████| 7765/7765 [00:16<00:00, 477.83it/s]
F1 Score : 0.6696716033483581



---

# Text Summarization

---

Practical Exercise

# 실습

예제 코드 다운로드:  
<https://github.com/KUNLP/Lecture>

- Pointer Generator 기반 요약 모델 코드 리뷰

## 입력 문서

6월 A매치 2연전에 소집 예정이었던 권창훈은 경주 골절로 인하여 명단에서 제외됐다. 권창훈(디종 FCO)이 또 다시 부상당하며 대표팀 명단에서 제외됐다. 권창훈은 파울루 벤투 감독의 호출을 받아

...



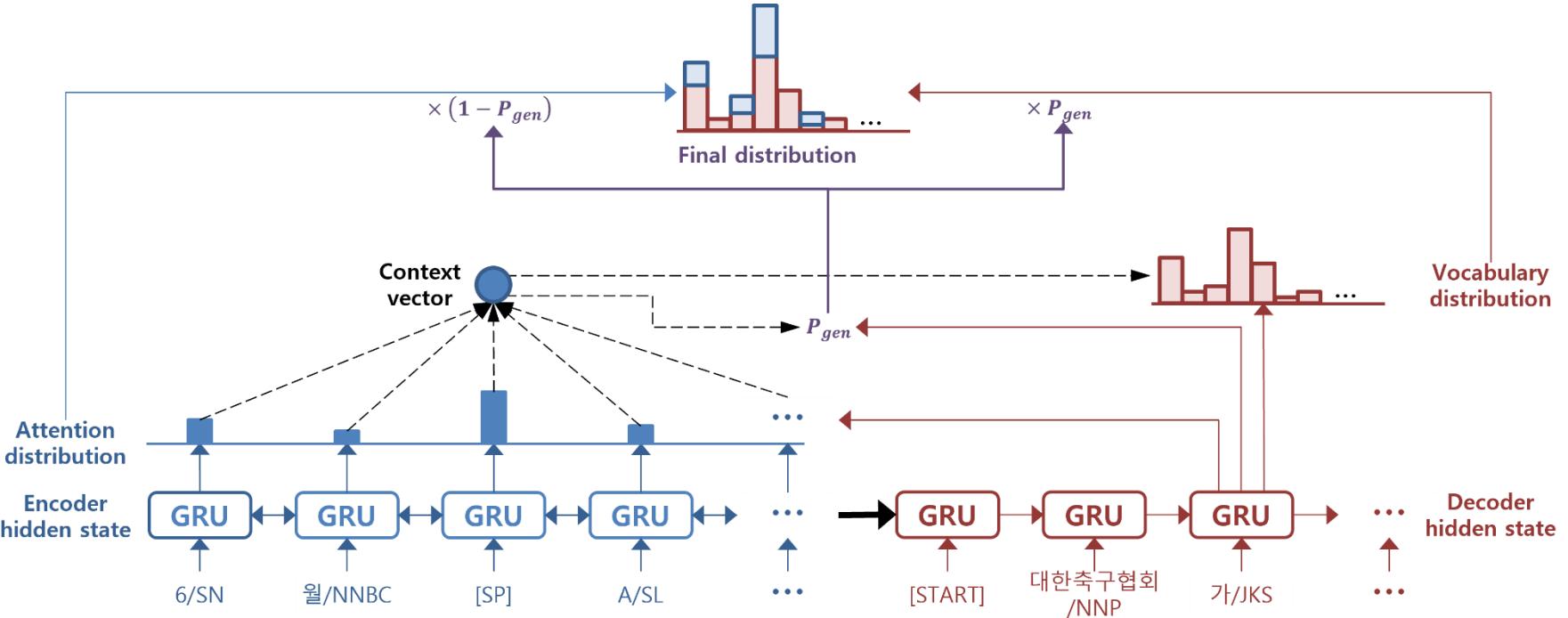
## 요약문

대한축구협회가 6월 A매치 참가 예정이었던 권창훈이 지난달 31일 리그 경기 도중 경주 골절 부상을 당함에 따라서 파울루 벤투 감독이 이끄는 대표팀 소집명단에서 제외되었다고 1일 발표했다.



# 실습

## 모델 구조



# 실습

모델 설계

생성자

```
class Pointer_Generator(nn.Module):
    def __init__(self, config):
        super(Pointer_Generator, self).__init__()

        # 전체 어휘 개수
        self.vocab_size = config["vocab_size"]

        # [UNK] 토큰에 대응하는 index
        self.unk_token_id = config["unk_token_id"]

        # 인코더의 최대 길이
        self.enc_max_len = config["enc_max_len"]

        # 디코더의 최대 길이
        self.dec_max_len = config["dec_max_len"]
        self.eps = 1e-31

        # 입력 데이터에 있는 각 형태소 index를 대응하는 임베딩 벡터로 치환해주기 위한 임베딩 객체
        # 기존에 사전학습 된 단어 임베딩을 사용할 수도 있고 랜덤으로 초기화 한 후,
        # 모델 학습 과정 중에 같이 학습 시키는 것도 가능
        # 예제 코드는 랜덤으로 초기화 한 후 같이 학습하도록 설정
        self.embedding = nn.Embedding(num_embeddings=config["vocab_size"],
                                      embedding_dim=config["embedding_size"],
                                      padding_idx=0)

        # 인코더
        self.encoder = Encoder(embedding=self.embedding, embedding_size=config["embedding_size"],
                               hidden_size=config["hidden_size"], dropout_rate=config["dropout_rate"])

        # 디코더
        self.decoder = Decoder(embedding=self.embedding, embedding_size=config["embedding_size"],
                               hidden_size=config["hidden_size"], vocab_size=config["vocab_size"],
                               dropout_rate=config["dropout_rate"],
                               unk_token_id=self.unk_token_id)

        self.dropout = nn.Dropout(config["dropout_rate"])

        self.teacher_forcing_rate = config["teacher_forcing_rate"]
```



# 실습

모델 설계

요약문 생성

```
def forward(self, input_features, output_features=None, ext_vocab_size=None):
    # input_features, output_features : (batch_size, max_len)

    batch_size = input_features.size(0)

    # encoder_outputs : (batch_size, curr_max_len, hidden_size*2)
    # encoder_hidden : (batch_size, hidden_size*2)
    encoder_outputs, encoder_hidden = self.encoder(input_features=self.filter_oov(input=input_features,
                                                                           ext_vocab_size=ext_vocab_size))
    curr_enc_max_len = encoder_outputs.size(1)

    # (batch_size, max_len) -> (batch_size, curr_max_len)
    input_features = input_features[:, :curr_enc_max_len]

    # (batch_size, ), 디코더의 시작 step 입력 값을 "[START]"로 초기화
    decoder_input = torch.ones(size=(batch_size, ), dtype=torch.long).cuda()
    # (batch_size, hidden_size*2) -> (1, batch_size, hidden_size*2)
    decoder_hidden = encoder_hidden.unsqueeze(0)
    decoder_outputs, enc_attn_weights = [], []
    # 학습
```



# 실습

모델 설계

요약문 생성  
(학습)

```
# 학습
if (output_features is not None):
    loss_func = nn.NLLLoss(ignore_index=0)
    loss = None

for step in range(self.dec_max_len):

    coverage_vector = torch.sum(torch.cat(enc_attn_weights), dim=0) if enc_attn_weights else None

    # decoder_output : (batch_size, ext_vocab_size)
    # decoder_hidden : (1, batch_size, hidden_size*2)
    # ptr_output : (batch_size, curr_max_len)
    # prob_ptr : (batch_size, 1)
    decoder_output, decoder_hidden, ptr_output, prob_ptr = self.decoder(decoder_input=decoder_input,
                                                                      decoder_hidden=decoder_hidden,
                                                                      coverage_vector=coverage_vector,
                                                                      input_features=input_features,
                                                                      encoder_outputs=encoder_outputs,
                                                                      ext_vocab_size=ext_vocab_size)

    decoder_output = torch.log(decoder_output + self.eps)

    if loss == None:
        loss = loss_func(decoder_output, output_features[:, step])
    else:
        loss += loss_func(decoder_output, output_features[:, step])

    if coverage_vector != None:
        coverage_loss = torch.sum(torch.min(coverage_vector, ptr_output)) / batch_size
        loss += coverage_loss

    if random.random() < self.teacher_forcing_rate:
        decoder_input = output_features[:, step]
    else:
        decoder_input = torch.argmax(decoder_output, dim=-1)

    decoder_outputs.append(decoder_output)
    enc_attn_weights.append(ptr_output.unsqueeze(0))

return loss
```



# 실습

모델 설계

요약문 생성  
(평가)

```
# 평가
else:

    decoder_outputs, enc_attn_weights = [], []
    for step in range(self.dec_max_len):

        coverage_vector = torch.sum(torch.cat(enc_attn_weights), dim=0) if enc_attn_weights else None

        # decoder_output : (batch_size, ext_vocab_size)
        # decoder_hidden : (1, batch_size, hidden_size*2)
        # ptr_output : (batch_size, curr_max_len)
        # prob_ptr : (batch_size, 1)
        decoder_output, decoder_hidden, ptr_output, prob_ptr = self.decoder(decoder_input=decoder_input,
                                                                           decoder_hidden=decoder_hidden,
                                                                           coverage_vector=coverage_vector,
                                                                           input_features=input_features,
                                                                           encoder_outputs=encoder_outputs,
                                                                           ext_vocab_size=ext_vocab_size)

        decoder_output = torch.log(decoder_output + self.eps)
        decoder_input = torch.argmax(decoder_output, dim=-1)

        decoder_outputs.append(decoder_output)
        enc_attn_weights.append(ptr_output.unsqueeze(0))

        # (max_length, batch_size, vocab_size)
        decoder_outputs = torch.stack(tensors=decoder_outputs, dim=0)
        # (max_length, batch_size, vocab_size) -> (batch_size, max_length, vocab_size)
        decoder_outputs = decoder_outputs.transpose(0, 1)

    return decoder_outputs
```



# 실습

모델 설계

인코더 생성자  
및 인코딩 단계

```
class Encoder(nn.Module):
    def __init__(self, embedding, embedding_size, hidden_size, dropout_rate):
        super(Encoder, self).__init__()

        self.embedding = embedding

        self.bi_gru = nn.GRU(input_size=embedding_size, hidden_size=hidden_size,
                           num_layers=1, batch_first=True, bidirectional=True)

        self.dropout = nn.Dropout(dropout_rate)

    def forward(self, input_features):
        # input_features : (batch_size, max_len)

        # 입력 sequence의 실제 길이
        # input_lengths : (batch_size, )
        input_lengths = (input_features != 0).sum(dim=-1)

        # (batch_size, max_len) -> (batch_size, max_len, embedding_size)
        input_features = self.embedding(input_features)

        packed_input_features = pack_padded_sequence(input=input_features, lengths=input_lengths.cpu(),
                                                      batch_first=True, enforce_sorted=False)
        # rnn_hidden : (2, batch_size, hidden_size)
        packed_rnn_outputs, rnn_hidden = self.bi_gru(input=packed_input_features)
        # rnn_outputs : (batch_size, curr_max_len, hidden_size*2)
        rnn_outputs, input_lengths = pad_packed_sequence(sequence=packed_rnn_outputs, batch_first=True)

        # (2, batch_size, hidden_size) -> (batch_size, hidden_size*2)
        rnn_hidden = torch.cat(tensors=(rnn_hidden[0], rnn_hidden[1]), dim=-1)

        rnn_outputs, rnn_hidden = self.dropout(rnn_outputs), self.dropout(rnn_hidden)
        return rnn_outputs, rnn_hidden
```



# 실습

모델 설계

디코더 생성자

```
class Decoder(nn.Module):
    def __init__(self, embedding, embedding_size, hidden_size, vocab_size, dropout_rate, unk_token_id):
        super(Decoder, self).__init__()

        self.embedding = embedding
        self.vocab_size = vocab_size
        self.unk_token_id = unk_token_id

        self.cover_weight = nn.Parameter(torch.rand(1))
        self.eps = 1e-31

        self.gru = nn.GRU(input_size=embedding_size, hidden_size=hidden_size*2, num_layers=1, batch_first=True)

        self.ptr_linear = nn.Linear(in_features=hidden_size * 4, out_features=1)

        self.gen_linear_1 = nn.Linear(in_features=hidden_size * 4, out_features=hidden_size*2)
        self.gen_linear_2 = nn.Linear(in_features=hidden_size*2, out_features=vocab_size)

        self.dropout = nn.Dropout(dropout_rate)

    def filter_oov(self, input, ext_vocab_size):
        if ext_vocab_size > self.vocab_size:
            new_input = input.clone()
            new_input[input >= self.vocab_size] = self.unk_token_id
            return new_input
        else:
            return input
```



# 실습

모델 설계

디코딩 단계

```
def forward(self, decoder_input, decoder_hidden, coverage_vector, input_features, encoder_outputs, ext_vocab_size):
    # decoder_input : (batch_size, )
    # decoder_hidden : (1, batch_size, hidden_size*2)
    # coverage_vector, input_features : (batch_size, curr_max_len)
    # encoder_outputs : (batch_size, curr_max_len, hidden_size*2)

    batch_size = decoder_input.size(0)

    # (batch_size, ) -> (batch_size, embedding_size) -> (batch_size, 1, embedding_size)
    decoder_input = self.embedding(self.filter_oov(input=decoder_input, ext_vocab_size=ext_vocab_size)).unsqueeze(1)

    # rnn_output : (batch_size, 1, hidden_size*2)
    # rnn_hidden : (1, batch_size, hidden_size*2)
    rnn_output, rnn_hidden = self.gru(decoder_input, decoder_hidden)
    rnn_output, rnn_hidden = self.dropout(rnn_output), self.dropout(rnn_hidden)

    # (batch_size, 1, curr_max_length)
    attn_weights = rnn_output.bmm(encoder_outputs.transpose(1, 2).contiguous())
    if coverage_vector != None:
        attn_weights += self.cover_weight * torch.log(coverage_vector.unsqueeze(1) + self.eps)
    attn_weights = F.softmax(attn_weights, dim=-1)

    # (batch_size, 1, hidden_size*2)
    context = attn_weights.bmm(encoder_outputs)

    # rnn_output : (1, batch_size, hidden_size*2) -> (batch_size, hidden_size*2)
    # context : (batch_size, 1, hidden_size*2) -> (batch_size, hidden_size*2)
    rnn_output, context = rnn_output.squeeze(1), context.squeeze(1)

    # (batch_size, hidden_size*4)
    combined = torch.cat(tensors=[rnn_output, context], dim=1)
```



# 실습

모델 설계

디코딩 단계

```
# pointer
# (batch_size, 1)
prob_ptr = torch.sigmoid(self.ptr_linear(combined))
# (batch_size, 1, curr_max_length) -> (batch_size, curr_max_length)
ptr_output = attn_weights.squeeze(1)

# generator
prob_gen = 1 - prob_ptr
# (batch_size, hidden_size*4) -> (batch_size, hidden_size*2)
gen_output = torch.tanh(self.gen_linear_1(combined))
# (batch_size, hidden_size*2) -> (batch_size, vocab_size)
gen_output = F.softmax(self.gen_linear_2(gen_output), dim=-1)

# (batch_size, ext_vocab_size)
output = torch.zeros(size=(batch_size, ext_vocab_size), dtype=torch.float, device=gen_output.device)
output[:, :self.vocab_size] = prob_gen * gen_output
output.scatter_add_(dim=1, index=input_features, src=prob_ptr * ptr_output)

return output, rnn_hidden, ptr_output, prob_ptr
```



# 실습

## Main

```
if (__name__ == "__main__"):
    root_dir = "/gdrive/My Drive/colab/Pointer_Generator"
    save_dir = os.path.join(root_dir, "save")
    output_dir = os.path.join(root_dir, "output")
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    set_seed(seed=1234)

    config = {"mode": "test",
              "vocab_data_path": os.path.join(root_dir, "vocab.txt"),
              "train_data_path": os.path.join(root_dir, "extracted_train_datas.json"),
              "test_data_path": os.path.join(root_dir, "extracted_test_datas.json"),
              "save_dir": save_dir,
              "output_dir": output_dir,
              "vocab_size": 10004,
              "embedding_size": 256,
              "hidden_size": 256,
              "dropout_rate": 0.1,
              "teacher_forcing_rate": 0.5,
              "unk_token_id": 3,
              "learning_rate": 1e-3,
              "enc_max_len": 500,
              "dec_max_len": 200,
              "epoch": 50,
              "batch_size": 32
             }

    if (config["mode"] == "train"):
        train(config)
    else:
        test(config)
```



# 실습

## 출력 예시

문서 : '미스트롯' 전국투어 콘서트에 오르는 12인이 베일을 벗었다. 18일 오후 방송된 TV조선 '내일은 미스트롯'(이하 '미스트롯')에서 준결승 진출자 12인이 공개됐다. 준결승 진출은 물론 서예정, 정답 요약 : 18일 오후 방송된 TV조선 '내일은 미스트롯'에서 준결승 진출자가 공개되며 전국투어 콘서트에 오르는 송가인과 흥자 등 12인이 베일을 벗었다.

출력 요약 : 18일 오후 방송된 TV조선 '내일은 미스트롯'(이하 '미스트롯')에서 준결승 진출자 12인이 공개되며 '미스트롯' 전국투어 콘서트에 오르는 12인이 베일을 벗었다.

문서 : 신용보증기금(이사장 윤대희)이 2019년도 제2차 전국본부점장회의를 13일 대구 본점에서 개최했다. 상반기 경영성과를 분석하고 하반기 주요업무 추진에 대한 각오를 다지기 위해 진행됐다.

정답 요약 : 신용보증기금(이사장 윤대희)이 상반기 경영성과를 분석하고 하반기 주요업무 추진에 대한 각오를 다지기 위해 2019년도 제2차 전국본부점장회의를 13일 대구 본점에서 개최하며 하반기 경영성과를 분석하고 하반기 주요업무 추진에 대한 각오를 다지기 위해 진행된다.

출력 요약 : 신용보증기금(이사장 윤대희)이 2019년도 제2차 전국본부점장회의를 13일 대구 본점에서 개최한 상반기 경영성과를 분석하고 하반기 주요업무 추진에 대한 각오를 다지기 위해 진행된다.

문서 : 한국수력원자력(주) 새울원자력본부(본부장 한상길)는 8~9일 양일간, 간절곶 잔디공원에서 시행된 '2019 간절곶 특산물 대축제'에 행사비 2억원을 지원했다. 지역 특산물의 가치를 널리 알리고자 기획된 행사를 마련하기 위해 8~9일 양일간, 간절곶 잔디공원에서 개최됐다.

정답 요약 : 한국수력원자력(주) 새울원자력본부(본부장 한상길)는 지역 특산물의 가치를 널리 알리고, 다양한 문화공연을 통한 지역 화합의 장을 마련하기 위해 8~9일 양일간, 간절곶 잔디공원에서 개최된 행사를 지원했다.

출력 요약 : 한국수력원자력(주) 새울원자력본부는 8~9일 양일간, 간절곶 잔디공원에서 시행된 '2019 간절곶 특산물 대축제'에 행사비 2억원을 지원하고 동시에에는 강길부 국회의원과 이선호 줄주:

문서 : 황원철 우리은행 디지털금융그룹장(오른쪽)이 한재선 그라운드X 대표와 협약 후 기념촬영했다. 우리금융그룹(회장 손태승)은 그라운드X(대표 한재선)와 블록체인 기반 금융서비스 개발을

정답 요약 : 지난 21일 카카오가 글로벌 블록체인 사업 진출을 위해 설립한 그라운드X와 우리금융그룹이 블록체인 기반 금융서비스 개발을 위해 업무협약을 체결했으며, 양사는 공동연구 등을 통해 협력할 예정이다.

출력 요약 : 우리금융그룹은 그라운드X(대표 한재선)와 블록체인 기반 금융서비스 개발을 위한 업무협약을 체결하고 블록체인 기반 금융서비스를 개발하고 공동연구 등을 통해 지속 가능한 협업 :

문서 : [헤럴드경제] '건물주 갑질' 등의 내용이 포함된 전단지를 배포했다 모욕죄로 기소당한 임차인이 대법원에서 무죄 판결을 받았다. '갑질'이라는 표현이 상대방을 불쾌하게 할 수는 있지만

정답 요약 : 대법원 2부는 '건물주 갑질' 등의 내용이 포함된 전단지를 배포했다 모욕죄로 기소당한 임차인 박모(57)씨의 사건의 상고심에서 30만원의 별금형을 선고한 원심 판결을 뒤집고 '갑질'이라는 표현을 포함한 전단지를 배포했다 모욕죄로 기소당한 임차인이 대법원에서 무죄 판결을 받았다.

문서 : 현대자동차가 공식 후원하는 최고 권위의 세계적 양궁 대회 '현대 세계 양궁 선수권 대회'가 10일부터 16일(현지 시각)까지 네덜란드 스헤르토렌보스('s-Hertogenbosch)에서 개최된다.

정답 요약 : 현대자동차는 1931년 시작된 최고 권위의 세계적 양궁 대회 '현대 세계 양궁 선수권 대회'를 2016년부터 타이틀 스폰서로 후원하고 있는데, 올해 대회는 10~16일 네덜란드 스헤르토렌보스에서 개최된다.

출력 요약 : 현대자동차가 공식 후원하는 최고 권위의 세계적 양궁 대회 '현대 세계 양궁 선수권 대회'가 10일부터 16일 네덜란드 스헤르토렌보스('s-Hertogenbosch)에서 개최되며 올해 대회는 10~16일 네덜란드 스헤르토렌보스에서 개최된다.



# 질의응답

---

Q & A

Homepage: <http://nlp.konkuk.ac.kr>  
E-mail: [nlpdrkim@konkuk.ac.kr](mailto:nlpdrkim@konkuk.ac.kr)

---

