Sentence Generation for Entity Description with Content-Plan Attention

2020 AAAI

2021.11.1 seminar

배수영

Content

- Abstraction
- Introduction
- Data-to-text generation(Content selection and Planning)
- Model Framework
- Experiments
- Results
- Conclusion

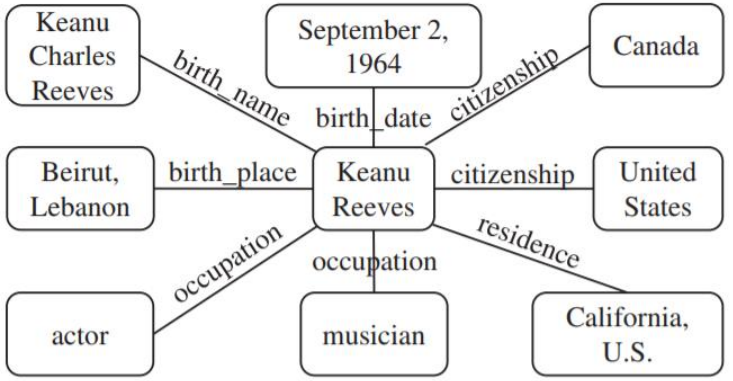
Abstraction

- Target data 정보를 입력으로 하고, 이 정보들을 포함하는 문장을 생성해 내는 것이 목표
- End-to-end data to text generation model 제안
- Joint learning of content planning and text generation
- Content planner + attention model -> generate texts (중요한 정보, 순서가 포함된)
- Baseline SOTA 에서 약 3% 향상, BLEU score 에서 약 5% 향상

Introduction

- Our goal: **structured data** -> sentence

Input	⟨name; Keanu Reeves⟩ ⟨birth_place; Beirut, Lebanon⟩ ⟨occupation; actor⟩ ⟨occupation; musician⟩ ⟨birth_date; September 2, 1964⟩ ⟨residence; California, U.S.⟩ ⟨birth_name; Keanu Charles Reeves⟩ ⟨citizenship; Canada⟩ ⟨citizenship; United States⟩
Output	Keanu Charles Reeves (born September 2, 1964 in Beirut, Lebanon) is a American actor who lives in California, U.S.



- Structured data 종류는 크게 두가지가 있다.
  1) Table to text generation : first col: key, second col: value
  2) RDF to text generation : Knowledge graph 형식

} 모두 Star-Shaped graph 형식

- 이런 structured data를 가지고 이 데이터 내용들을 포함하는 문장을 생성해야 하는 task.

Introduction

- How?
- Seq2seq 구조 사용
- 이때 **input 속성들의 순서**가 generation task에서 중요한 역할을 한다.

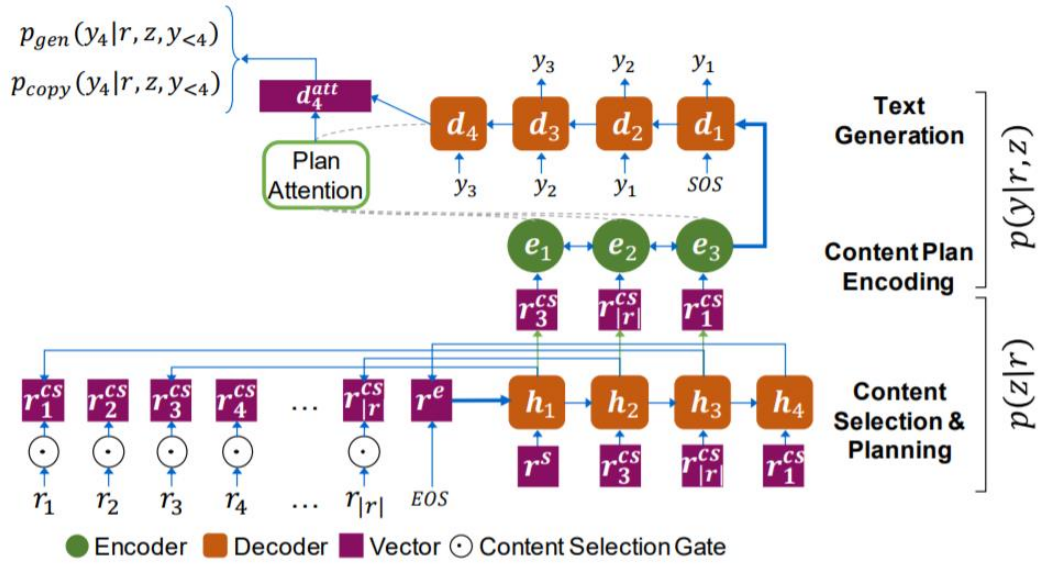
content-plan: <birth name, birth date, birth place, occupation, residence>
output: Keanu Charles Reeves (born September 2, 1964 in Beirut, Lebanon) is a American actor who lives in California, U.S.

- Neural Content Planning (NCP)
  - Two-stage model
  1) Content planner: pointer network 사용해서 content-plan 생성
  2) Text generator: content-plan을 encoder-decoder model input으로 사용
  - Content planner, text generator사이 error 발생
- **Encoder-decoder model의 attention model에 content-plan을 연결해 end-to-end model을 구현함.**

Data-to-text generation(Content Planning)

- Data-to-text generation: non-linguistic input을 사용해서 input 정보를 포함하는 문장을 생성하는 것.
- Content Planning: 생성되는 문장에 대해 정해진 형식을 결정하는 것.
- Two-stage model

1. Content Plan
   - Record representation: self-attention mechanism 사용해서 주변 정보 담아준다.
   - Content planning output token: pointer network 사용해서 각 position에 대한 token을 생성한다. =content plan sequence
2. Text generation
   - Encoder decoder model의 encoder에 순서가 정해진 record가 입력으로 들어가고 decoder를 통해 문장이 생성된다.



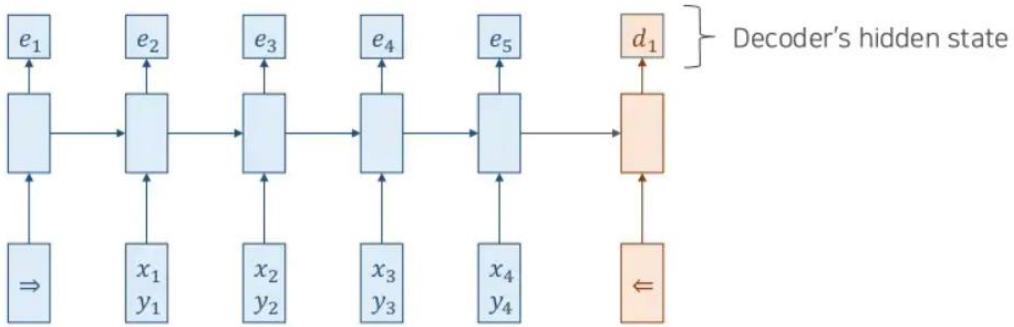
$$p(y|r) = \sum_z p(y, z|r) = \sum_z \underbrace{p(z|r)}_{\text{Content planner}} \underbrace{p(y|r, z)}_{\text{Text generation}}$$

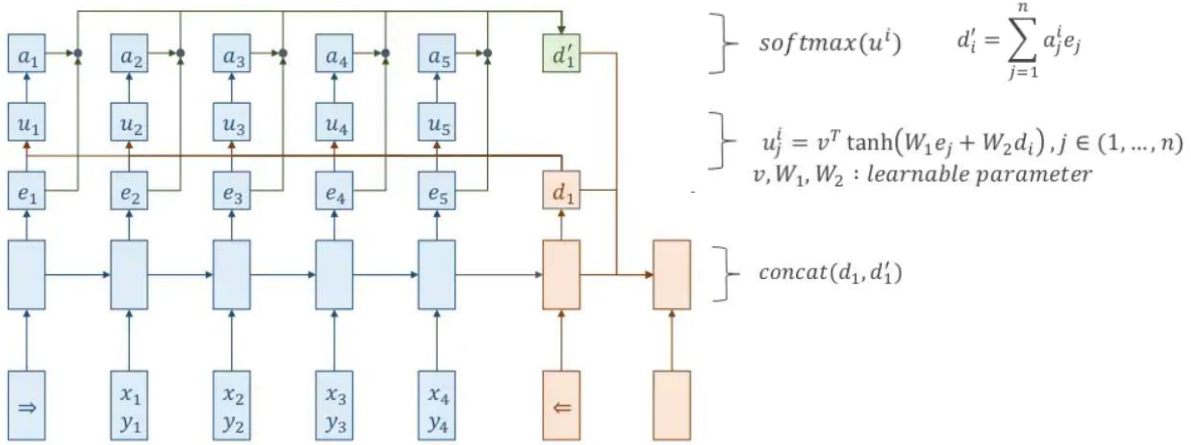
y: output text
r: input record
z: content plan

https://arxiv.org/pdf/1809.00582.pdf

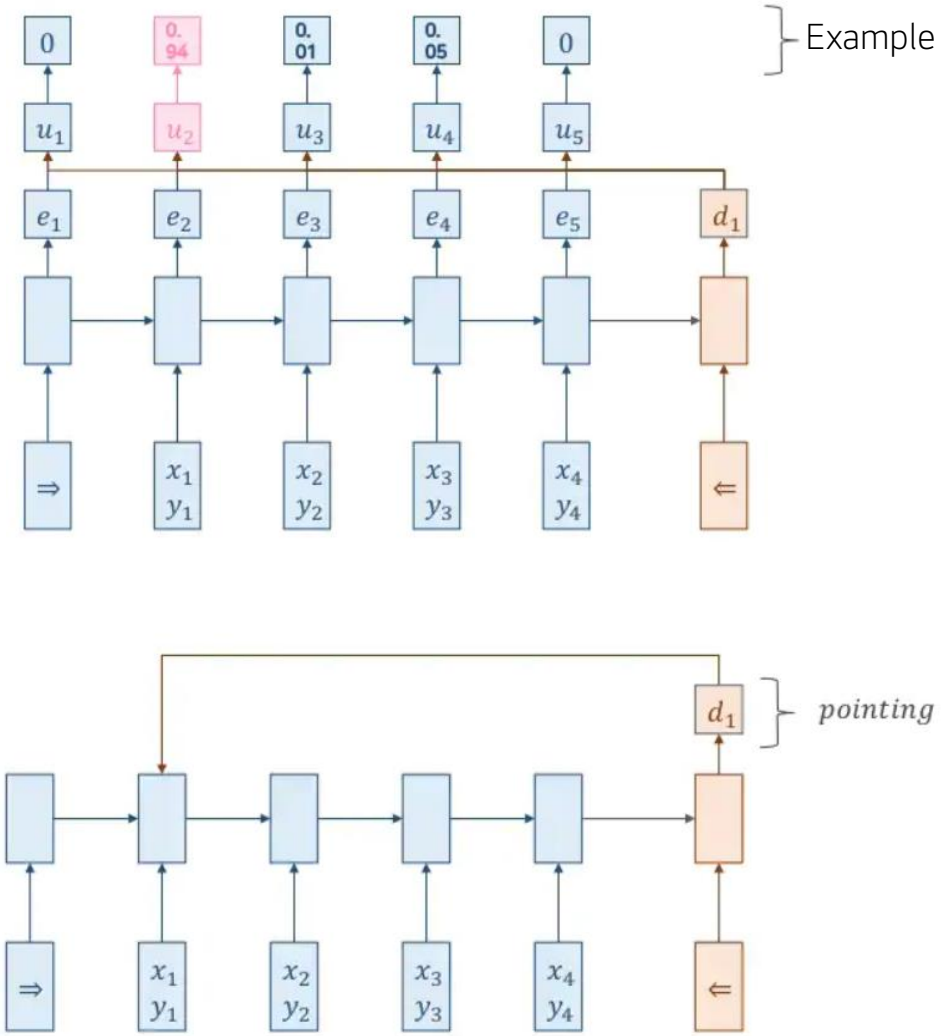
Pointer Network

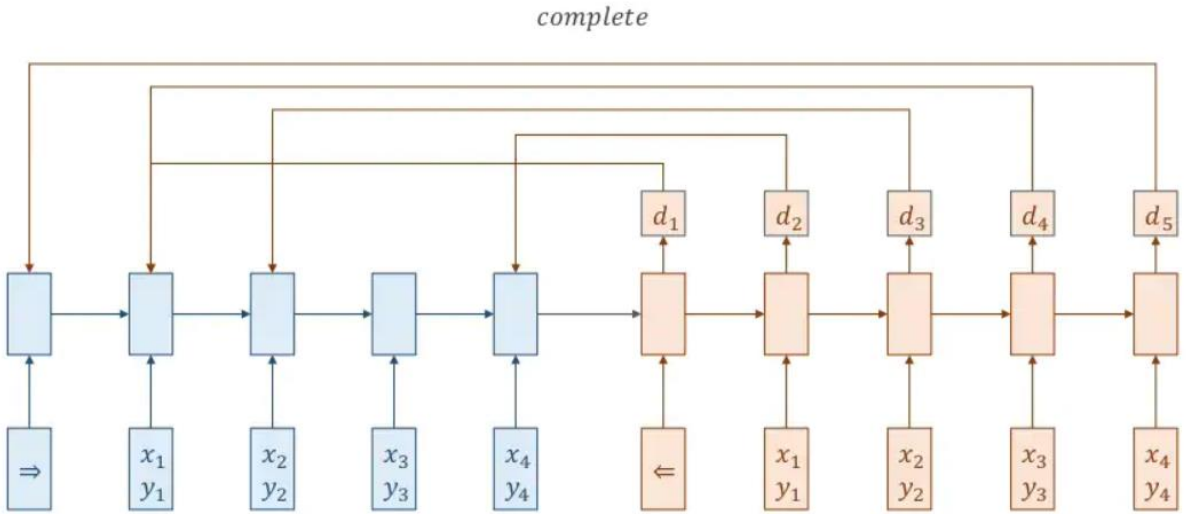
- Seq2seq 모델을 사용해서 task를 진행할 때 문제점
  - 입력 크기가 변할 때 어려움. <- 입력 크기가 고정되어 있기 때문에
  - 장거리 관계에 대한 학습 어려움 -> attention에서 해결 가능한 문제





Pointer Network





입력 크기에 상관없이 가장 확률이 높은 입력 index를 선택하면 되기 때문에 문제 해결

https://www.slideshare.net/gyuhyeonNam/study-pointer-networks

Model Framework

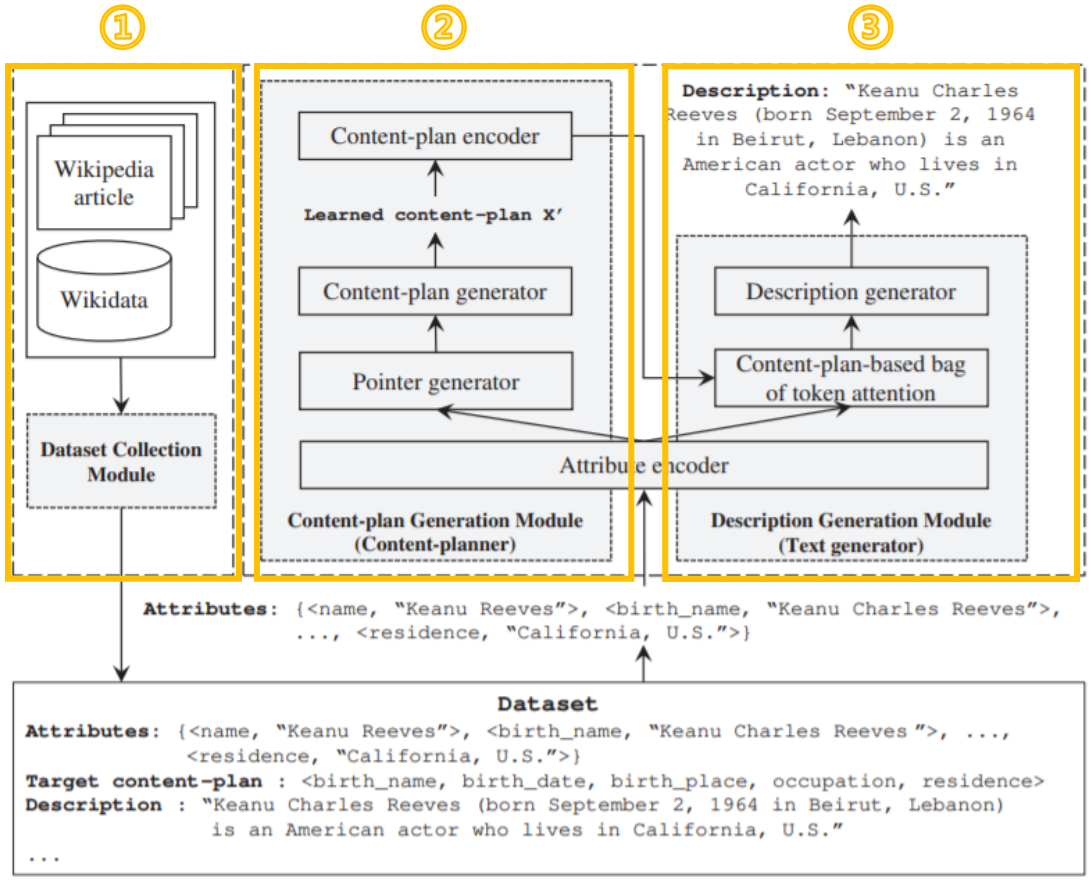


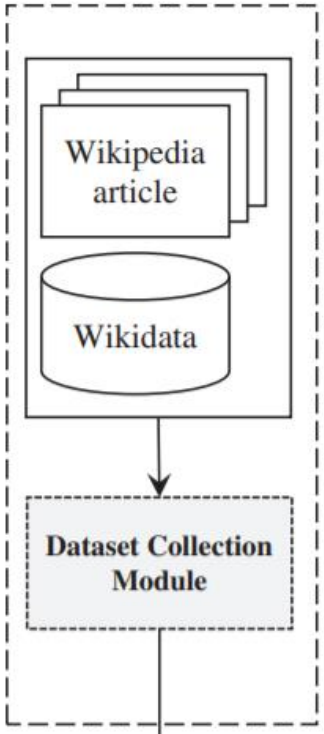
Figure 2: Overview of our proposed solution

1. Data Collection Module
   - Wikipedia로부터 real-world data 수집
   - Attributes, content-plan, description 수집

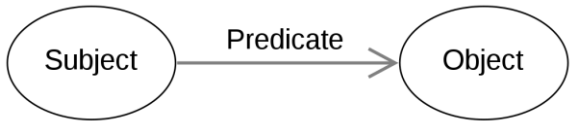
2. Content-plan Generation Module(Content Planner)
   - Train pointer network
   - 4 stage 로 구성되어 있다.
     - Attribute encoder
     - Pointer generator
     - Content-plan generator
     - Content-plan encoder

3. Description Generation Module
   - Content plan을 attention mechanism에 적용시킨다.
   - Content planner와 같은 Attribute encoder를 사용한다.
   - Coverage mechanism 도입

1. Data Collection module



- Attribute, Target content-plan, Description 수집

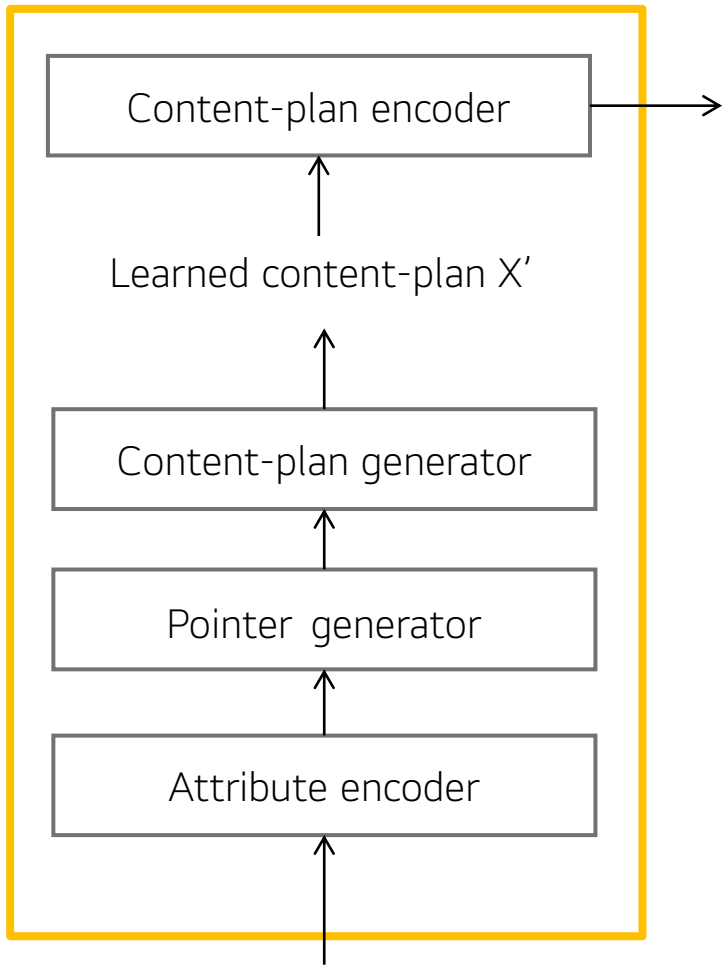
Attribute	Querying Wikidata RDF triple 형식에서 수집 Subject=target entity 
Content-plan	String matching을 사용해 description 순서대로 attribute key값을 matching 한다.
Description	Wikipedia 페이지 첫 문장

- 152,231 triples of attributes, content-plan, description (WIKIALL dataset)
- 728,321 biographies for benchmarking (WIKIBIO dataset)
- train: dev: test=80:10:10

Dataset

Attributes: {<name, "Keanu Reeves">, <birth_name, "Keanu Charles Reeves ">, ...,
<residence, "California, U.S.">}
Target content-plan : <birth_name, birth_date, birth_place, occupation, residence>
Description : "Keanu Charles Reeves (born September 2, 1964 in Beirut, Lebanon)
is an American actor who lives in California, U.S."
...

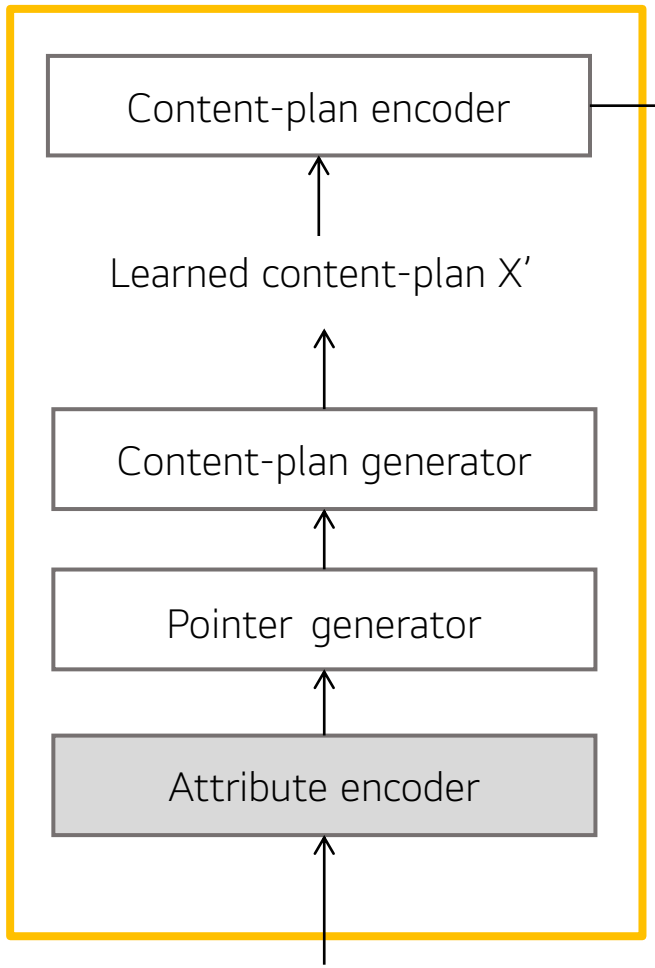
2. Content-plan Generation



- Input: key, value pair로 이루어진 attribution set {<k1;v1>,<k2;v2>,...<kn;vn>}
- Pointer network 는 Attribute의 올바른 순서를 만들기 위해 attention mechanism을 사용해서 content-plan을 학습한다.
- 4 stage
  - Attribute encoder
  - Pointer generator
  - Content-plan generator
  - Content-plan encoder

Attributes: {<name, "Keanu Reeves">, <birth_name, "Keanu Charles Reeves">, ..., <residence, "California, U.S.">}

2. Content-plan Generation



1. Attribute encoder

(1) Input A={<k1;v1>,<k2;v2>,...<kn;vn>}

(2) Transform multiple tokens into a single token representation

+positional encoding

(3) $A=[\langle k_1^1, v_1^1, f_1^1, r_1^1\rangle, \langle k_1^2, v_1^2, f_1^2, r_1^2\rangle, ..., \langle k_n^j, v_n^j, f_n^j, r_n^j\rangle]$

(4) key, value attribute에 대한 vector representation

$$\boldsymbol{z_k}_n^j = \tanh(\boldsymbol{W_k}[\boldsymbol{k}_n^j; \boldsymbol{f}_n^j; \boldsymbol{r}_n^j] + \boldsymbol{b_k})$$

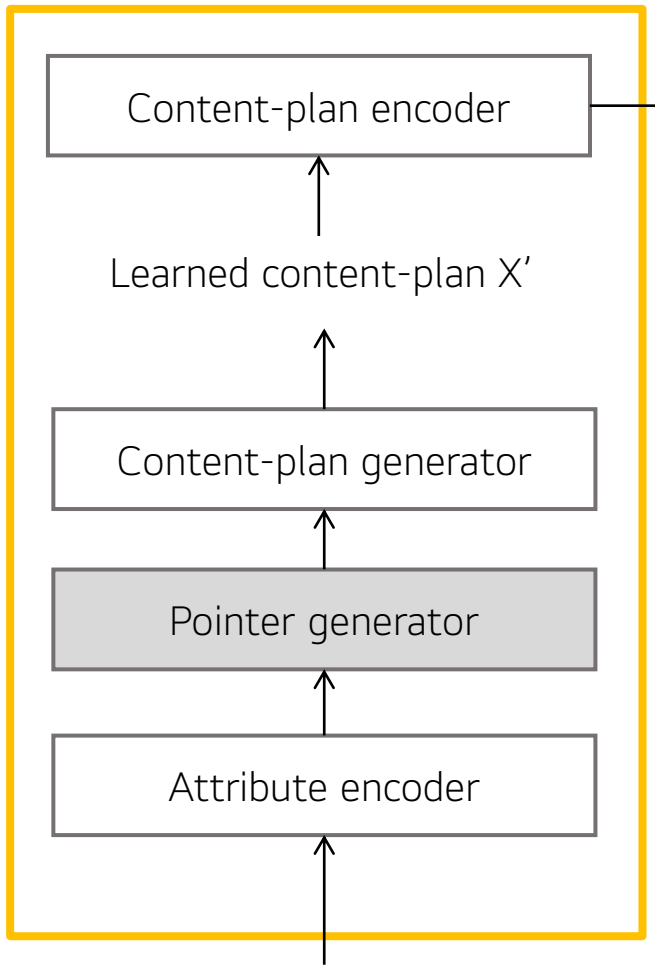
$$\boldsymbol{z_v}_n^j = \tanh(\boldsymbol{W_v}[\boldsymbol{v}_n^j; \boldsymbol{f}_n^j; \boldsymbol{r}_n^j] + \boldsymbol{b_v})$$

(5) output: attribute-token representation

$$\boldsymbol{x_a}_n^j = \tanh(\boldsymbol{z_k}_n^j + \boldsymbol{z_v}_n^j)$$

Attributes: {<name, "Keanu Reeves">, <birth_name, "Keanu Charles Reeves">, ..., <residence, "California, U.S.">}

2. Content-plan Generation



2. Pointer generator

(1) Input A= $\langle \boldsymbol{x_{a1}}, \ldots, \boldsymbol{x_{am}} \rangle$

(2) LSTM사용해서 encoding:

$$\boldsymbol{c_{ptr}}_g = f_{lstm}(\boldsymbol{x}_{i_{g-1}})$$

(3) 가장 높은 attention 값을 가지는 attribute-token을 선택해서 next index를 예측한다.

$$\boldsymbol{u_{ptr}}_g = \tanh(\boldsymbol{W_{p1}} \boldsymbol{x_a} + \boldsymbol{W_{p2}} \boldsymbol{c_{ptr}}_g)$$

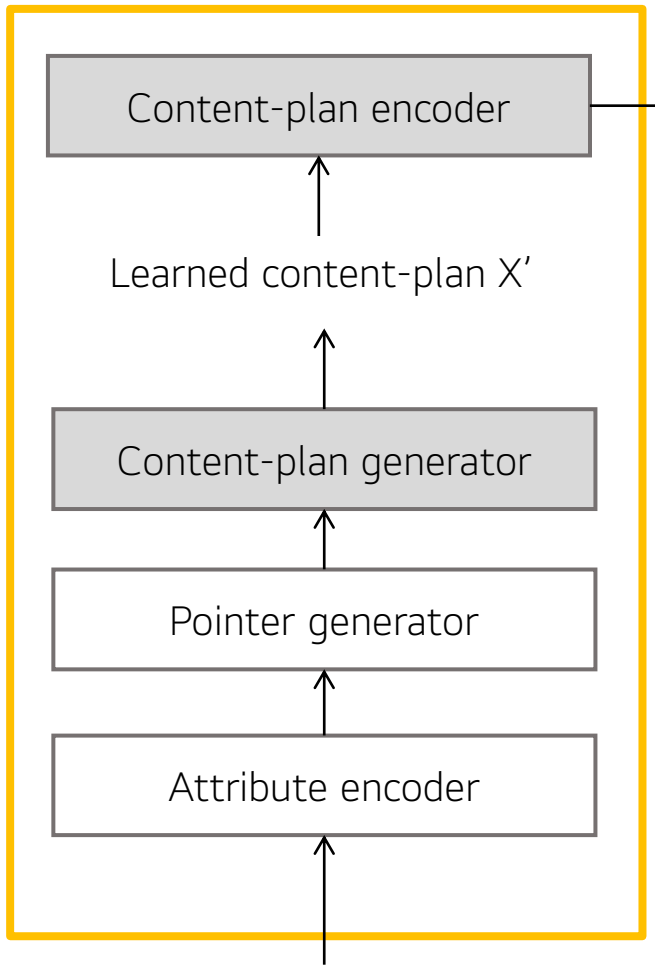
$$\hat{i}_g = \text{softmax}(\boldsymbol{u_{ptr}}_g)$$

(4) Output: pointer index 시퀀스: content-plan에서 표현하는 attribute token X의 순서를 나타내는 index 시퀀스

$$I = \langle i_1, \ldots, i_g \rangle$$

Attributes: {<name, "Keanu Reeves">, <birth_name, "Keanu Charles Reeves">, ..., <residence, "California, U.S.">}

2. Content-plan Generation



3. Content-plan generator and encoder

(1) Input $I = \langle i_1, ..., i_g \rangle$

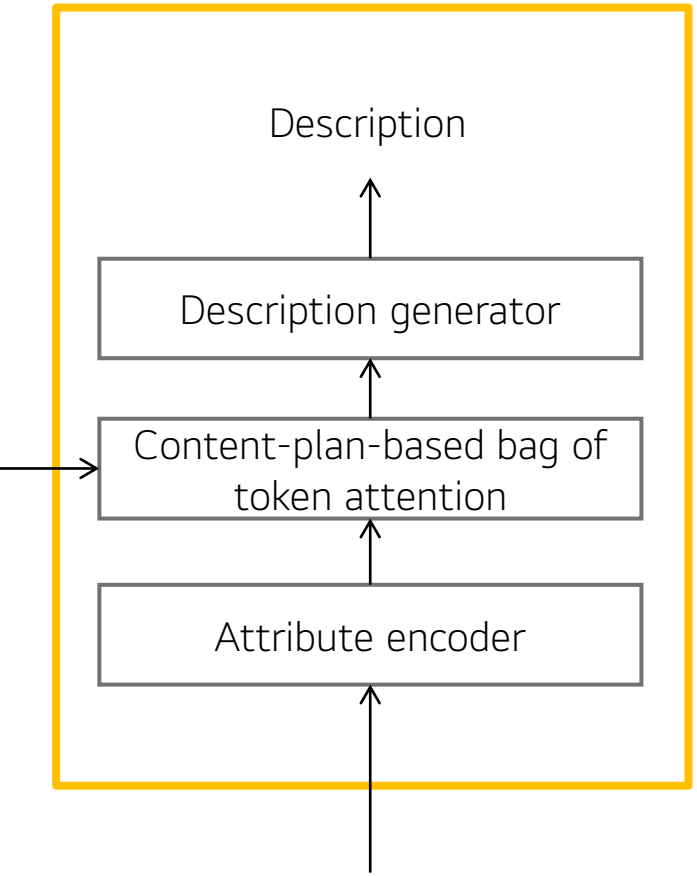
(2) Pointer index를 사용해서 attribute-token 시퀀스를 content-plan X'로 다시 표현한다.

(3) LSTM 사용해서 인코딩 $\langle \boldsymbol{x_{cp_1}}, ..., \boldsymbol{x_{cp_g}} \rangle$

(4) Text generator로 넘어가서 attention model이 올바른 순서로 attribute를 선택하도록 도와준다.

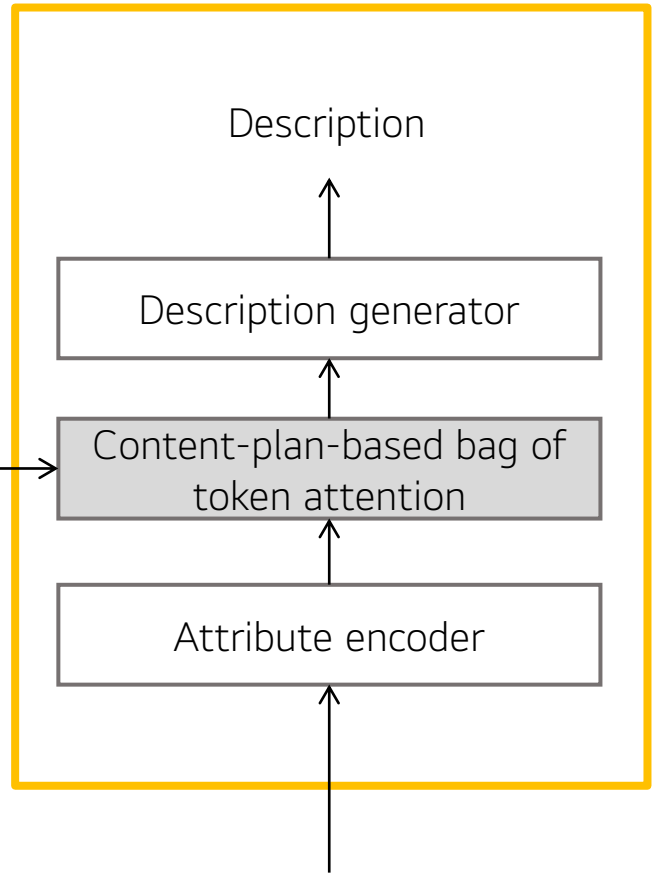
Attributes: {<name, "Keanu Reeves">, <birth_name, "Keanu Charles Reeves">, ..., <residence, "California, U.S.">}

3. Description Generation



- Content plan generation module 인코더와 같은 인코더를 사용한다.
  - -> 처음 input을 기반으로 context vector 계산
  - -> input 길이 가변적.
- 학습된 content-plan을 attention mechanism에 사용해서 다음 생성 token 예측한다.
- Coverage mechanism 사용: content-plan attribute history 저장
- 3 stage
  - Attribute encoder
  - Content-plan based bag of token attention
  - Description generator

3. Description Generation



1. Content-plan-based bag of token attention

- Disordered input을 다루기 위해 bag of token 사용한다.
- Content plan encode하기 위해 LSTM 사용
- Attention mechanism에 content-plan을 사용한다.

$$\boldsymbol{d_{cov_l}} = \sum_{l'=0}^{l-1} \boldsymbol{a_{cp_{l'}}}$$ ->이전 time step에 사용된 content-plan distribution 더하기

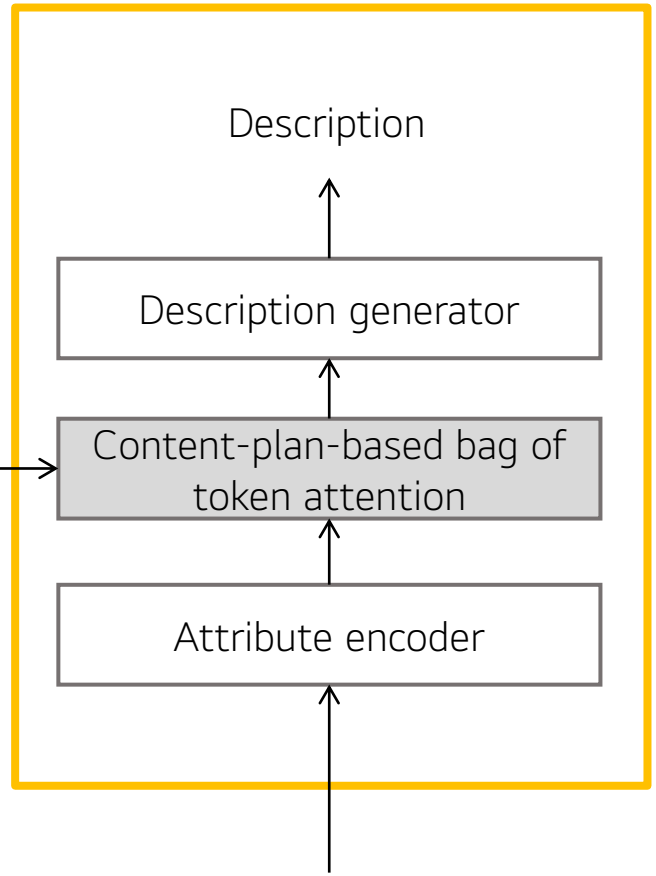
$$\boldsymbol{u_{cp_l}} = \tanh(\boldsymbol{W_{c1}x_{cp}} + \boldsymbol{W_{c2}h_{d_{l-1}}} + \boldsymbol{w_{c3}d_{cov_l}})$$

$$\boldsymbol{a_{cp_l}} = \sum_{g} \mathrm{softmax}(\boldsymbol{u_{cp_l}})\boldsymbol{x_{cp_g}}$$ ->content-plan 사용해서 attention 계산

$$\boldsymbol{a_{d_l}} = \tanh(\boldsymbol{W_{d1}x_a} + \boldsymbol{W_{d2}h_{d_{l-1}}} + \boldsymbol{W_{d3}a_{cp_l}})$$

$$\boldsymbol{c_{d_l}} = \sum_{m} \mathrm{softmax}(\boldsymbol{a_{d_l}})\boldsymbol{x_{a_m}}$$ ->attention 값 계산해서 context vector 계산

3. Description Generation



2. Description generator

- LSTM description generator
- Output probability distribution over the vocabulary

$$\boldsymbol{h}_{\boldsymbol{d}l} = f_{lstm}([\boldsymbol{h}_{\boldsymbol{d}l-1}; \boldsymbol{t}_{l-1}; \boldsymbol{c}_{\boldsymbol{d}l}])$$
$$\hat{t}_l = \text{softmax}(\boldsymbol{V}\boldsymbol{h}_{\boldsymbol{d}l})$$

- Output description:

Description: "Keanu Charles Reeves (born September 2, 1964 in Beirut, Lebanon) is an American actor who lives in California, U.S."

Experiments

1. Datasets

- Two real-world dataset 사용
- WIKIALL: disordered
- WIKIBIO : ordered-> random shuffle

2. Models

- Field Gating with Dual Attention model(FGDA): table-to-text generation model
- Graph-based Triple LSTM encoder model(GTRLSTM): RDF-to-text generation model
- Neural Content-Planning model (NCP): data-to-text generation model / using content-plan
- modified encoder-decoder model(MED): modification of the standard encoder-decoder model that uses the embeddings of its input to compute the attention.

Results

Model	WIKIALL				WIKIBIO (disordered)				WIKIBIO (ordered)			
	BLEU↑	ROUGE↑	METEOR↑	TER↓	BLEU↑	ROUGE↑	METEOR↑	TER↓	BLEU↑	ROUGE↑	METEOR↑	TER↓
MED	58.54	54.01	42.80	34.40	38.21	33.32	30.20	55.20	40.25	35.63	30.90	56.40
GTRLSTM	62.71	57.02	44.30	29.50	42.64	38.35	32.60	54.40	42.06	37.90	32.20	54.80
NCP	63.21	57.28	44.30	28.50	43.07	38.76	33.90	53.20	43.12	38.82	33.90	53.30
FGDA	62.61	57.11	44.10	30.20	42.31	38.93	32.20	55.00	44.59	40.20	34.10	52.10
Proposed	**65.12**	**58.07**	**45.90**	**27.50**	**45.32**	**40.80**	**34.40**	**51.50**	**45.46**	**40.31**	**34.70**	**51.30**

- BLEU: 기계 번역과 사람 번역 결과가 얼마나 유사한 지.
- ROUGE: 모델이 생성한 번역본과 사람이 만들어 놓은 번역본 대조해 성능 계산
- METEOR: 생성 텍스트와 참조 텍스트 단어 1:1 매핑 계산
- TER: 번역 오류비율(error rate)
- BLEU, ROUGE에서 본 모델 성능이 SOTA보다 높은 성능을 보임
- NCP: 가 상대적으로 좋은 성능을 보여주고 있다. -> content-planning이 data-to-text generation 에 도움을 준다. (NCP, 본 모델이 같은 content-planning 기법을 사용하고 있다.)

Results

Model	Correctness	Grammaticality	Fluency
MED	2.25	2.32	2.26
GTRLSTM	2.31	2.40	2.36
NCP	2.54	2.68	2.51
FGDA	2.51	2.58	2.54
Proposed	**2.68**	**2.76**	**2.57**

- Human evaluation
- Correctness(순서가 맞는지), Grammaticality(문법), Fluency(반복이 있는지 없는지) 세 평가항목을 사람이 평가
- WIKIALL dataset에서 300개 무작위 추출
- 6명 annotators
- 250 hours
- 3: error x/ 2: one error / 1: more than one error

Conclusion

1. End-to-end data to text generation model 제안
2. content-planning과 text generation을 같이 학습시켜 content-planner과 generated text 사이의 error를 줄인다.
3. string matching을 통해 description에 대한 content-plan을 만들 때 semantic similarity를 체크하지는 못하는 한계
   -> semantic similarity 추후 연구 필요