착수보고서

전기컴퓨터공학부 정보컴퓨터공학전공
신우창(201424479)
이태오(201424513)
김윤호(201424427)

과제명 : 딥러닝과 검색엔진을 활용한 질의응답시스템

지도교수 : 권 혁 철

목차

1. 과제의 목표

딥러닝과 검색엔진을 활용한 질의응답 시스템

- 제한된 도메인에서 정보검색 기술을 바탕으로 사용자의 질문에 응답하는 한국어 질의응답 시스템을 개발한다.
- 검색 엔진으로 사용자의 질문의 주제에 해당하는 위키피디아 문서를 검색하고, 딥러닝을 이용하여 위키피디아 문서 내에서 해당하는 답변을 추출하는 것을 목표로 한다.

2. 대상 문제 및 요구사항 분석

2.1. 대상문제

방대한 양의 정보에서 필요한 지식을 추출해 내는 것은 고도의 지능을 요구하는 작업으로, 이는 인간의 생산활동 중 가장 중요한 부분이다. 지식추출은 관련 분야의 지식, 논리력 등의 고도의 지능을 요구한다. 지식추출 인공지능기술은 지식추출작업에 필요한 시간과 자원 그리고 인간이 지식추출작업에 필요한 능력을 함양하기 위해 필요한 시간과 자원을 절약시키고, 인간의 신속하고 효율적인 의사결정을 보조하여 생산성을 향상 시킬 수 있을 것이다.

2.2. 요구사항

1. User Interface(질의)

   클라이언트 프로그램은 사용자가 질문을 하고 답변을 받는 과정에 있어서 여러가지 편의성을 제공하고, 사용자는 클라이언트 프로그램의 UI를 통해 제한 없이 질문을 입력 할 수 있어야 한다.

2. 검색엔진

   검색엔진은 사용자의 질의에 따라 한국어 위키피디아의 모든 문서에서 관련 정보가 있는 텍스트를 고속으로 검색하여 출력하여야 한다.

3. 질의응답 모델 구현

   질의 응답 모델은 자연어 처리 모델의 자연어 처리 결과를 입력으로 받아 사용자의 질의에 대해 가장 높은 응답을 출력해야 한다.

4. User Interface(응답)

   사용자는 클라이언트의 UI를 통해 질의에 대한 응답을 확인 할 수 있어야 한다.

3. 현실적 제약사항 분석 및 대책

3.1 제약사항

1. 한국어 질의 응답 시스템

   과제의 목표에 따라 질의 응답 시스템의 언어처리는 한국어로 제한된다.

2. 검색엔진

연구실에서 제공하는 미리내 검색엔진을 개량하여 사용한다.

3.2 대책

1. 제약사항 1. 에 대한 대책

   한국어 Machine Reading Comprehension을 위해 만든 dataset Korquad를 사용한다.

2. 제약사항 2. 에 대한 대책

   대용량 검색 처리를 위한 inverted index 알고리즘과 검색 키워드에 가장 부합하는 문서를 검색 결과 최상위에 배치하는 알고리즘 TF-IDF을 사용하여 구현한다.

4. 설계 문서

4.1. 기술소개

1. 딥러닝

   딥러닝, 심층학습은 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화를 시도하는 머신러닝 알고리즘의 집합으로 정의되며, 큰 틀에서 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한분야라고 이야기할 수 있다.

2. 자연어(한국어) 처리

   자연어(natural language)란 우리가 일상 생활에서 사용하는 언어를 말한다. 자연어 처리(natural language processing)란 이러한 자연어를 의미를 분석하여 컴퓨터가 처리할 수 있도록 하는 일을 말한다. 자연어 처리는 음성 인식, 내용 요약, 번역, 사용자의 감성 분석 텍스트 분류작업, 질의 응답 시스템, 챗봇과 같은 곳에서 사용되는 분야입니다.

4.2. 개발환경

개발 언어 : Python(딥러닝), JAVA(클라이언트), C++(검색엔진)

개발 도구 : PyCharm(딥러닝), Android Studio(클라이언트), Visual Studio (검색엔진)

대상 시스템 : 윈도우PC(서버), 안드로이드PC(클라이언트)

4.3. 시스템 구성도

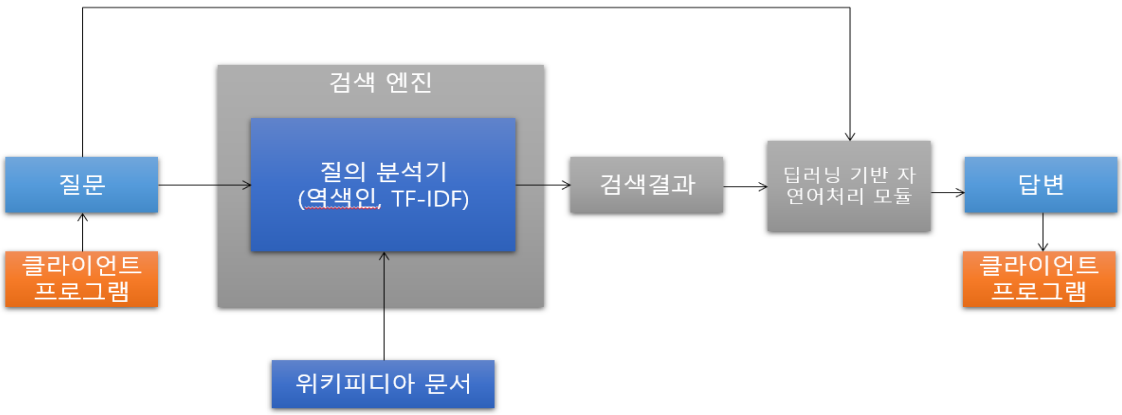


Figure 1 시스템 구성도

4.4. 주요 모듈

1. 검색엔진

Traditional SQL 에서는 LIKE 검색이 INDEX 기능을 이용할 수 없다는 단점이 있어서, 본 과제에서는 그 문제를 극복하기 위해서 단어(Term)로 인덱싱을 하는 "Inverted Index" 방식을 사용한다. 기존의 데이터베이스가 하나의 구분자(Primary Key)가 여러 필드를 지정하고 있었다면 Inverted Index에서는 하나의 값(Term)이 해당 Term이 들어간 document id를 지정하고 있다.

Original documents		Lucene's inverted index			
Doc #	**Content field**	**Term**	**Doc #**	**(Continued)...**	
1	A Fun Guide to Cooking	a	1,3,4,5,6,7,8	...	...
2	Decorating Your Home	becoming	8	guide	1,6
3	How to Raise a Child	beginner's	6	home	2,5,7,8
4	Buying a New Car	buy	9	house	6,9
5	Buying a New Home	buying	4,5,6	how	3,9
6	The Beginner's Guide to Buying a House	car	4	new	4,5,8
		child	3	owner	8
7	Purchasing a Home	cooking	1	purchasing	7
8	Becoming a New Home Owner	decorating	2	raise	3
9	How to Buy Your First House	first	9	the	6
		fun	1	to	1,6,9
		...	...	your	2,9

인덱싱(Indexing;미리 순서대로 정렬해서 저장해두는 방식)을 하기 때문에 탐색 속도가 매우 빠르다.

검색엔진은 사용자가 검색한 단어를 기반으로 유사한 문서를 먼저 출력해 주어야 한다. 사용자의 검색어와 document 의 Relevancy를 Scoring방식으로 평가한다.

Score(q,d) =

$$\sum_{t \text{ in } q} \left(\mathbf{tf}(t \text{ in } d) \cdot \mathbf{idf}(t)^2 \cdot \mathbf{t.getBoost}() \cdot \mathbf{norm}(t,d) \right) \cdot \mathbf{coord}(q,d) \cdot \mathbf{queryNorm}(q)$$

Where:

t = term; **d** = document; **q** = query; **f** = field

$\mathbf{tf}(t \text{ in } d) = \text{numTermOccurrencesInDocument}^{1/2}$

$\mathbf{idf}(t) = 1 + \log (\text{numDocs} / (\text{docFreq} + 1))$

$\mathbf{coord}(q,d) = \text{numTermsInDocumentFromQuery} / \text{numTermsInQuery}$

$\mathbf{queryNorm}(q) = 1 / (\text{sumOfSquaredWeights}^{1/2})$

$\mathbf{sumOfSquaredWeights} = q.\text{getBoost}()^2 \cdot \sum_{t \text{ in } q} (\text{idf}(t) \cdot t.\text{getBoost}())^2$

$\mathbf{norm}(t,d) = d.\text{getBoost}() \cdot \text{lengthNorm}(f) \cdot f.\text{getBoost}()$

- Term frequency (tf)

문서에 단어의 빈도가 많으면 score가 올라간다.

- Inverse document frequency (idf)

흔하지 않은 단어가 사용되면 score가 올라간다.

2. 텐서플로우

텐서플로우는 구글이 개발한 머신러닝 프레임워크 오픈소스로, C++, Java 등에서 사용 가능하지만 대표적으로 파이썬에서 가장 많이 사용되고 있다. 여러 머신러닝과 인경신경망 관련 객체와 함수들이 API 형태로 구현되어 있다.

3. 딥러닝 기반 자연어처리 모듈

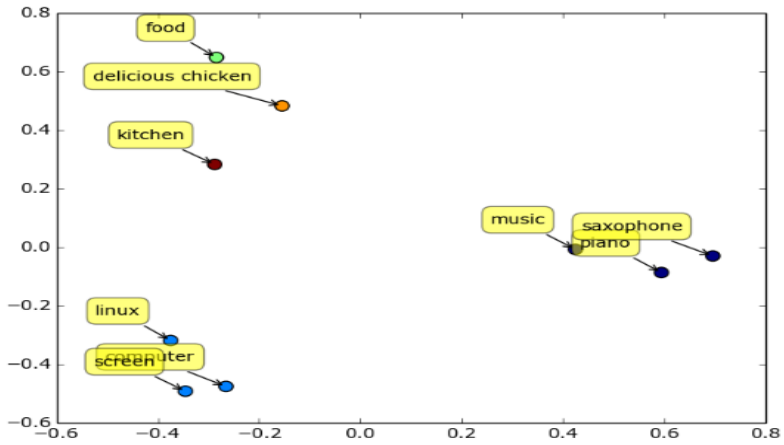
LG CNS에서 만들어 배포하고 있는 KorQuad Dataset을 이용한 모델들을 참고한다. KorQuAD는 한국어 Machine Reading Comprehension을 위해 만든 dataset이다. 모든 질의에 대한 답변은 해당 Wikipedia 아티클 문단의 일부 하위 영역으로 이루어진다. Stanford Question Answering Dataset(SQuAD) v1.0과 동일한 방식으로 구성되어 있다. 전체 데이터는 1,560 개의 Wikipedia article에 대해 10,645 건의 문단과 66,181 개의 질의응답 쌍으로, Training set 60,407 개, Dev set 5,774 개의 질의응답쌍으로 구분하였다.

자연어 처리에서 사람의 언어를 컴퓨터에서 사용하기 위해서는 단어를 컴퓨터에서 사용할 수 있도록 Vector화하는 과정이 필요하기에 Word Embedding 방식을 사용한다. Word Embedding은 Word2Vec나 Glove 등과 같은 방식을 통해 Language Model을 학습시키고 그것을 통해 얻은 Vector를 단어를 나타내는 Input으로 사용하는 방식이다. 본 과제에서는

Figure 2 Word Embedding

GloVe로 학습된 Pre-Trained Model을 이용한다.

본 과제에선 BiDAF, Bert, Google QaNet 모델들을 참고하여 기능을 개발한다.



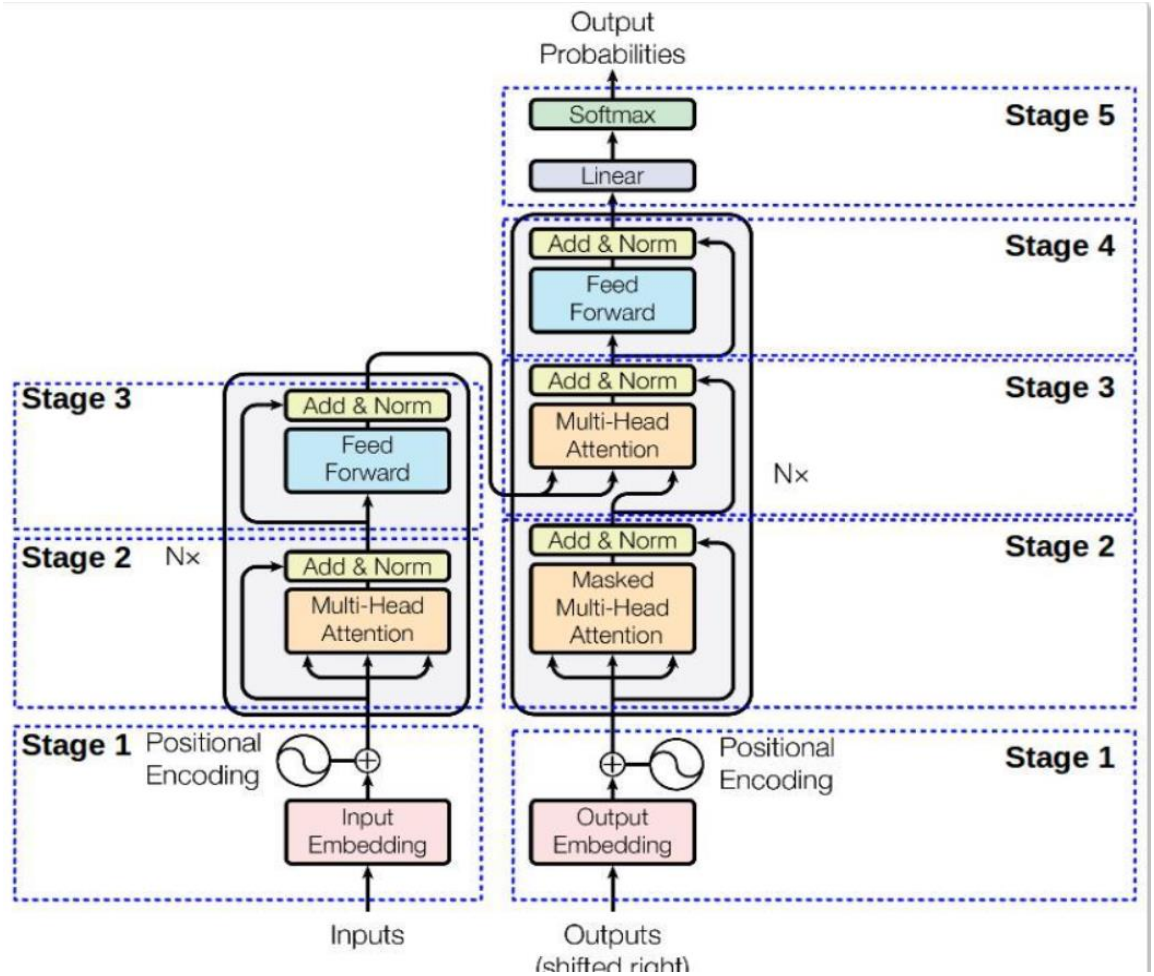


Figure 3 Bert 구조

Transformer Architecture 자체는 위에 보는 구조를 반복하는 것으로 이해 할 수 있으며, 기존의 어떤 아키텍쳐와도 다르게, CNN, RNN 구조 없이 전체다 Attention 만으로 구성된 아키텍쳐로

굉장히 많은 메모리를 요구하는 아키택쳐라고 볼 수 있다. 이 아키택쳐는 몇 가지 컴포넌트로 구성되어 있다.

- Scaled Dot Product Attention : Multi Head Attention 을 구성하는 작은 단위로 미리 이해가 필요하다.
- Multi-Head Attention : 복수의 Scaled Dot Product Attention 을 Concat 하여, 더 나은 Attention 을 구하고자 한다.
- Positional Encoding : 동일한 단어라고 하여도 위치에 따라서 다른 해석을 부여 하기 위한 장치이다.
- Short Cut & Add/Norm : Resinet 과 같은 Short Cut 을 통한 Vanising Problem 의 최소화한다.
- Decoder Side : Encoder 의 Attention 을 Decoder 에 적용하여 최종적인 판단까지 과정이다.

5. 데이터셋 부가설명

데이터셋 이름	사용 될 모듈	데이터 구성	데이터크기
위키피디아	검색엔진	위키피디아 문서 제목 및 내용	1320446개 문장
KorQuad	딥러닝 자연어 처리 모듈	질문 – 질문의 주제에 해당하는 위키피디아 문서 제목 – 해당 위키피디아 문서내에서 정답에 해당하는 단어	10,645건의 문단 66,181개의 질의응답

6. 추진체계 및 일정

6.1. 역할분담

구성원	역할
신우창	- 검색엔진 개량 - 클라이언트 개발
이태오	- 자연어 처리 모델 구현 - 질의 응답 모델 구현 - 모델 학습 및 최적화
김윤호	- 서버 개발 - 위키피디아 데이터 정제

6.2. 개발일정

월 / 진행 항목	5	6	7	8	9	10
착수보고서 작성						
서버/클라이언트						
검색엔진 개량						
중간보고서 작성						
자연어 처리모델						
데이터 정제						
최종보고서 작성						
Test 및 보완						
졸업과제 발표						
결과물 제출/ SW등록						

7. 참고문헌

[1] https://korquad.github.io/

[2] https://mrseo.co.kr/웹페이지의-랭킹을-매기는-tf-idf 기법/

[3] https://blog.lael.be/post/3056

[4] https://www.bloter.net/archives/264262

[5] https://medium.com/ai-networkkr

[6] http://www.aitimes.kr/news/articleView.html?idxno=13117

[7] https://ko.wikipedia.org/wiki/%EB%94%A5_%EB%9F%AC%EB%8B%9D