Ⅰ 비즈니스 문제 정의

1. 스마트 워킹

최근 화두가 되고 있는 스마트 워킹에 대해 알아본다.

최근 스마트워킹에 대한 관심이 높아지고 있다. 정부에서 앞으로 근로자 30%를 스마트워킹으로 전환하는 것을 목표로 정책을 추진 중이고 대기업을 중심으로 기업에서도 스마트 워킹을 적극적으로 채택하려는 움직임을 나타내고 있다. 우리나라의 경우 스마트 워킹이 최근에 부상하고 있지만 유럽이나 미국 그리고 일본의 경우 이미 오래전부터 추친하고 있는 주제이다. 미국은 2001년 연방정부 차원에서 재택근무 시험 프로젝트 추진을 시작으로 공공부문을 중심으로 스마트 워크를 확산시키고 있다. 일본 또한 2005년 '원격근무 인구 배증 계획', '2010년 스마트 워킹 20% 확대 정책' 등 적극적 정책을 통해 스마트 워킹을 견인하고 있다.

개별 기업 차원에서도 스마트 워킹이 활성화되고 있는데, 대표적 사례는 영국의 BT이다. BT는 90년대 중반 스마트 워킹을 시범 도입한 이래 직원의 15% 이상인 14,000여명이 재택근무를 하고 있으며, 63,000여명이 유연 근무를 하는 등 전 직원의 85% 이상이 스마트 워킹 체제로 근무하고 있다.

그렇다면 이러한 스마트 워킹이란 정확히 무엇을 말하는 것일까?

일반적으로 스마트 워킹을 모바일 오피스, 재택 근무, 시차 근무제, 탄력 근무제 등 IT 솔루션이나 복무 제도의 하나 정도로 인식하는 경우가 대다수이다. 그러나 이것은 좁은 의미에서의 스마트 워킹이다. 조금 더 의미를 확장해 보면, **기존의 사무실에서 벗어나 언제 어디서나 일을 편리하고 효과적으로 하는 것임과 동시에 이를 위해 일하는 방식과 제도, 프로세스 그리고 문화를 변화시키는 일체의 활동**을 의미한다.

[그림 1] 스마트 워크를 통한 일하는 방식의 변화

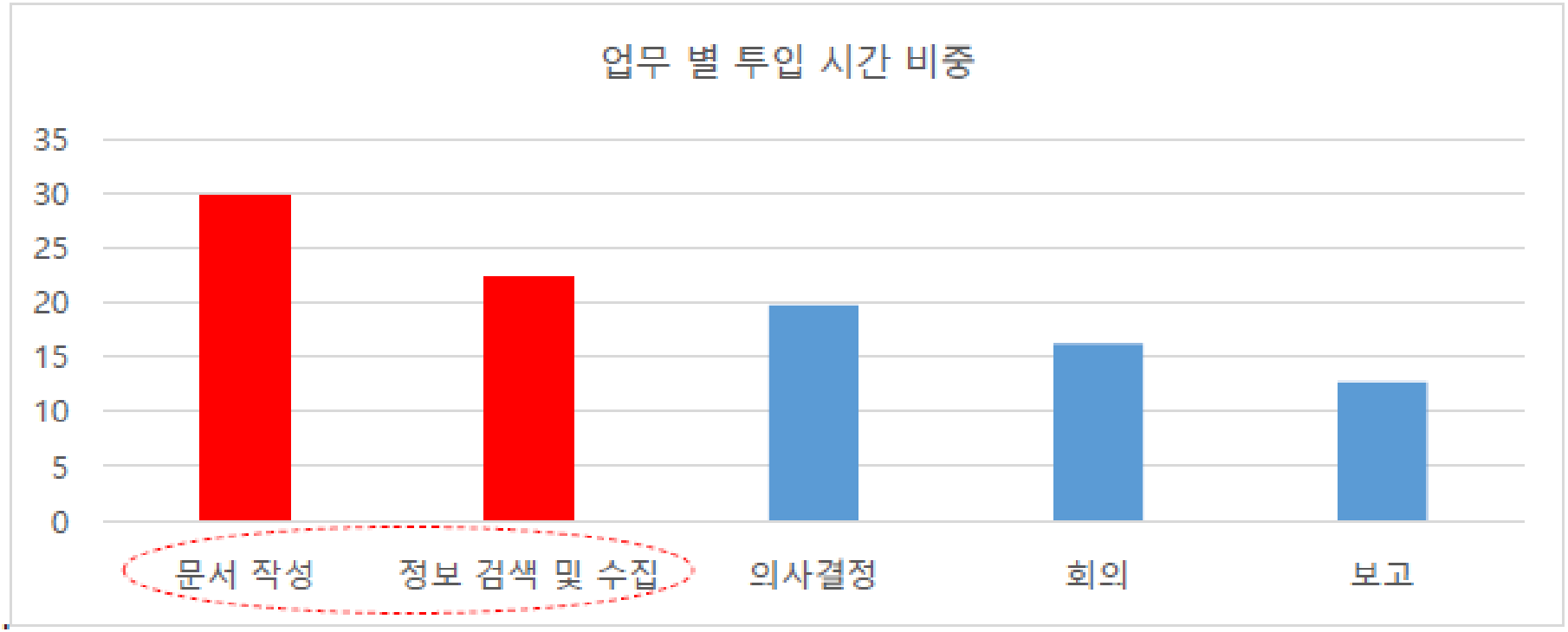
이러한 스마트 워킹을 통해 발생할 수 있는 기대효과는 무수히 많다. 먼저 사회적 측면에서 자유롭고 효율적으로 일할 수 있는 환경을 조성함으로써 출산율 건전성 확보, 고용 증대, 지역 균형 발전은 물론 환경 친화적 경제성장에 기여할 수 있다. 또한 기업적 측면에서는 혁신과 창의성 그리고 개방적 협력과 성과 중심의 문화를 정착시키고, 직원 만족을 기여함으로써 생산성 향상, 비용 절감, 우수 인재 유치 등의 효과를 얻을 수 있다. 개인적 측면에서도 각 개인들이 일과 삶을 조화롭게 영위하여 윤택한 생활을 하는데 도움을 줄 수 있다.

[그림2] 스마트 워크를 통한 기대효과

1. 문제 정의

그렇다면 정보와 기업 차원에서 적극적으로 추진하고 있는 스마트 워킹을 저해하는 요소는 무엇일까?

실제 모 기업에서 실시한 '스마트 워킹을 저해하는 요소'라는 설문조사 결과에 따르면 그 1위가 '불필요한 보고서'이다. 이는 최근 '재택근무' 및 '모바일 근무'와 같은 스마트 워킹을 위한 인프라들이 상당 수준 제공되어 있음에도 불구하고 '편리하고 효과적인 일'을 하기 위해서는 결국 비효율적인 기업 문화의 개선이 필요함을 의미한다. 또한 한국생산성본부(KPC)의 조사에 따르면 한 달에 6건 이상의 기획/제안/발표 문서를 작성하는 직장인이 전체의 23.6% 그리고 매달 10장이 넘는 보고서를 만드는 직장인도 21.6%를 차지하였다. 특히 직장인이 평소 문서 작성에 투입하는 시간이 전체 업무 시간의 29.7% 그리고 정보검색 및 수집에 들이는 시간은 22.3%로 정보검색 및 수집과 문서작성에만 일과의 절반 이상(52.0%)을 소비하는 것으로 나타났다



[출처] 한국생산성본부(KPC) '스마트워크 및 조직 창의성 보고서'

이렇게 업무 생산성 및 효율성을 크게 저해하고 있는 '불필요한 보고서 작성', 그 해결 방안은 무엇일까?

Ⅱ 비즈니스 문제 해결 방안

1. 문제 해결 방안

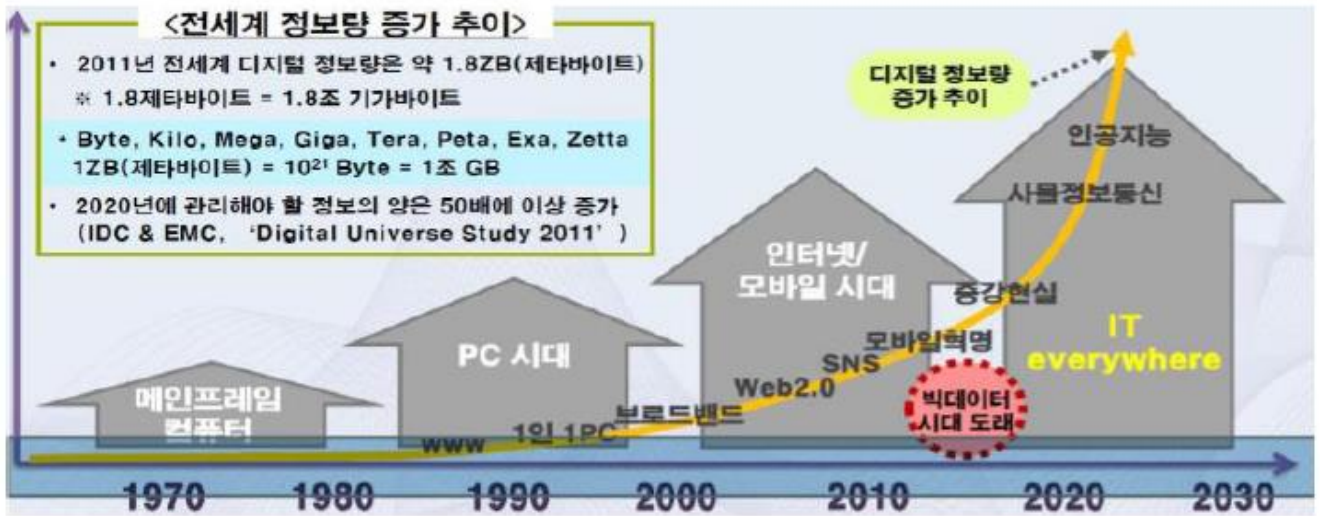
근본적인 기업 문화 변화

'불필요한 보고서 작성' 문제가 완전히 해결되기 위해서는 근본적인 기업 문화가 변화되어야 한다. 실제 한 대기업에서는 '보고서 없는 사무실'을 만들어 대리나 과장급들이 보고서를 만드느라 온종일 신간을 빼앗겨온 관행을 없애고 있다. 또 다른 기업은 회의를 위한 완성된 보고서를 준비하는데 쓸데없는 시간을 낭비하지 않도록 '보고서 없는 회의'를 추진하여 업무 생산성을 높이고 있다. 그러나 이러한 기업들의 노력에도 불구하고 여전히 '정보 검색 및 수집'과 '문서 작성'은 전체 업무량의 절반을 넘어가고 있다.

지능 정보를 활용한 인공지능 보고서 봇

본 문서는 이러한 문제의 해결 방안을 지능 정보에서 찾고자 한다. 즉 **온라인 상에서 정보를 검색하고 리서치 하여 통합된 새로운 형태의 보고서를 작성하는 것을 사람이 아닌 인공지능이 대신** 하여 **업무의 생산성 및 효율성을 높이는 것**이다.

과거에는 이러한 인공지능이 불가능하였다. 여러가지 원인이 있겠지만 그 중 하나를 꼽자면 바로 '정보의 부족'이다. 즉 보고서 작성을 위해 학습하고자 하는 정보의 양이 절대적으로 부족하였다. 하지만 지금은 바야흐로 빅데이터 시대이다. 다시 말해 각종 디지털 매체(인터넷/모바일 등)들을 통해 수 많은 정보들을 손쉽게 접할 수 있다.



[그림 1] 시간의 흐름에 따른 전세계 정보량 증가 추이 차트

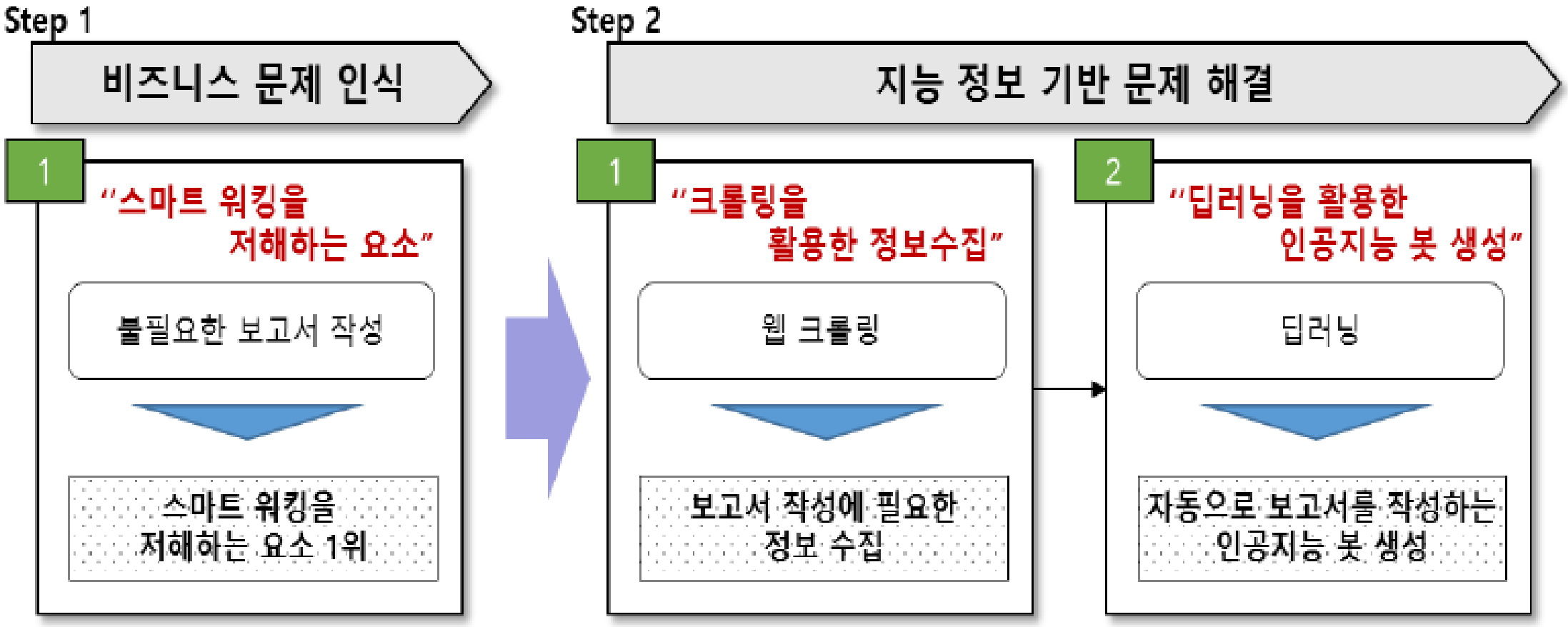
따라서 이러한 수 많은 정보를 활용하여 보고서를 생성 할 수 있는 인공지능을 개발하는 것은 불가능한 것이 아니다.

본 실습에서는 다양한 정보들 중에서도 '**뉴스 기사**' **데이터를 활용하여 취합된 새로운 형태의 보고서를 생성하는 보고서 봇**을 개발하여 이러한 문제들을 해결하도록 하겠다.

2. 문제 해결 과정

주요 비즈니스 문제인 '불필요한 보고서 작성'을 해결하기 위해 지능 정보를 활용한 문제 해결 과정을 설명한다. 특히 본 실습에서는 뉴스 기사를 기반으로 새로운 형태의 보고서를 생성하는 보고서봇 개발에 초점을 맞추어 진행 하도록 하겠다.

[문제 해결 과정]



[분석 과정]

[분석 환경]

필요 환경

- 운영체제: Ubuntu 14.04 LTS 이상 (아래 코드는 Windows에서도 테스트 완료)
- 텐서플로우 버전: r0.11 이상 (r.0.12 에서도 테스트 완료)
- 파이썬 버전: 3.5.x

[분석 기법]

필수 이론

- 딥러닝 알고리즘 RNN 및 LSTM
- 언어 모델링 알고리즘 Word Embedding
- 위의 이론들은 해당 컨텐츠를 수행하기 앞서 '딥 러닝 기본 이론' 교재를 통해 선행되어야 함.

Ⅲ 비즈니스 문제 정의

최근 경제 분야에서 큰 이슈가 되었던 삼성 하만 인수를 주제로 보고서를 작성하는 인공지능 봇을 구현한다.

1. 필요 데이터 확보 및 탐색

최근 경제 분야에서(2016년 11월 기준) 큰 이슈가 된 '삼성의 하만 인수'에 대한 보고서 작성을 위해 관련 뉴스기사를 웹크롤링을 통해 수집한다.

우선 '삼성 하만 인수'와 관련된 뉴스 기사 데이터가 필요하다. 실제 뉴스 기사 수집은 웹 크롤링을 통해 이루어지지만 본 교육 컨텐츠의 목적 상 그 과정은 생략한다. 따라서 본 컨텐츠에서는 크롤링을 통해 수집된 데이터(약 60여개의 뉴스 기사로 이루어진 텍스트 파일)가 생성되었다는 가정 하에 다음 과정을 진행하겠다.

1.1 데이터 불러오기

실행 코드

```
# 라이브러리를 불러옵니다.
import numpy as np
import tensorflow as tf
import collections
import argparse
import time
import os
from six.moves import cPickle
```

```
# 데이터 처리를 위한 기본 라이브러리인 numpy를 불러온다.
# 기타 텍스트 데이터 전 처리를 위한 라이브러리들 역시 불러온다.
# 데이터의 수를 세기 위한 라이브러리인 collections를 불러온다.
# 전체 경과 시간을 측정하기 위한 라이브러리인 time을 불러온다.
# 데이터 경로를 합치기 위한 라이브러리인 os를 불러온다.
# byte 데이터로 저장하고 불러오기 위한 라이브러리인 cPickle을 불러온다.
```

실행 코드

```
# 데이터를 불러옵니다.
save_dir    = "/home/eduuser/NN/Data/Report"
data_dir    = "/home/eduuser/NN/Data/Report"
input_file  = os.path.join(data_dir, "Samsung.txt")
with open(input_file, "r") as f:
    data_train = f.read().split()
```

여러 개의 문장으로 이루어 진 뉴스 기사 데이터를 분석하는데 필요한 최소 단위는 '단어'이다. 따라서 '삼성 하만 인수'와 관련된 약 60여개의 뉴스 기사를 위 실행 코드와 같이 **단어 단위로 나누어 저장**한다.

1.2 데이터 탐색

[불필요한 정보 다루기]

데이터를 탐색하기 위해 단어 단위로 저장된 data_train 을 출력하여 확인한다.

실행 코드

```
print(data_train[1:30])
```

출력 결과

```
['미국의', '자동차', '전장', '전문기업', '하만(Harman)을', '9조원대에', '인수한다는', '소식에', '투자자들은', '두', '기업의', '사업', '시너지에', '큰', '기대감을', '보였다.', '15일', '금융투자업계에', '따르면', '삼성전자는', '이번', '인수로', '글로벌', '전장부품', '공급업체', '지위를', '단숨에', '획득하게', '될']
```

단어 별로 저장된 것임을 알 수 있다. 일반적으로 데이터를 분석하기 위해서는 텍스트(문장, 단어 등) 분석 시 숫자 및 기호들은 불필요한 정보로 인식하여 제거하는 것이 일반적이다. 그러나 **뉴스 기사의 특성 상 숫자 및 기호 각각은 의미(정보)**를 지니고 있다. 예를 들어 '하만(Harman)', '9 조원대에' 두 단어를 살펴 보자. 두 단어에서 '( )'와 '9'라는 기호 및 숫자를 제거해 버린다면 해당 뉴스 기사에서 말하고자 하는 바가 변질될 것이다. 따라서 본 실습에서는 숫자와 기호를 필요한 정보로 인지하여 제거하지 않도록 하겠다.

[배치 데이터 생성]

본 실습은 '취합된 형태의 보고서 생성'이 그 목적이다. 즉 단어들 그리고 해당 단어들로 이루어진 문장 다음에 **위치 할 '단어 및 문장'**을 예측하는 것이 본 실습의 핵심이다. 이는 뉴스 기사에 쓰여진 모든 단어들이 그 의미가 있고 분석에 활용될 수 있음을 뜻한다. 따라서 '시나리오 1 민원'에서와는 다르게 '단어의 수' 그리고 '문장의 길이'에 대한 탐색 과정을 생략하고 바로 **배치 데이터를 생성**하도록 하겠다.

실행 코드

```
# 언어 데이터 전처리 과정
# 각 단어의 등장 횟수를 셉니다.
counter = collections.Counter(data_train)
count_pairs = sorted(counter.items(), key=lambda x: (-x[1], x[0])) # <= Sort

# 각 단어에 고유 번호를 부여하고 해당 단어와 번호로 이루어진 사전을 만듭니다.
chars, counts = zip(*count pairs)
vocab = dict(zip(chars, range(len(chars))))

# 생성된 데이터를 저장합니다.
with open(os.path.join(save_dir, 'chars_vocab.pkl'), 'wb') as f:
    cPickle.dump((chars, vocab), f)

print(chars[1:20])
print(counts[1:20])
print(vocab)
```

```
# 추후 뉴스 생성을 위해 cPickle.dump 메서드를 활용하여 학습 데이터를 저장한다.
```

출력 결과

```
('있다.', '수', '삼성전자는', '있는', '삼성전자가', '하만의', '하만', '것으로', '등', '전장', '삼성전자의', '
이', '오디오', '하만을', '통해', '하만은', '커넥티드', '이번', '글로벌')

(124, 121, 87, 82, 77, 75, 71, 68, 68, 65, 58, 52, 49, 47, 44, 44, 43, 42, 41)

{'기업이거든요.': 2702,  '특성과': 5323, '사운드': 3606, '속도에': 3791, '때': 318, '전망해주셨는데요.':
4695, '나돌았지만,': 2745, '총액은': 891, '디스카운트되는': 3039, '결정짓게': 2418, '평했다.': 5384, '완전히
': 825, '이사회는': 4397, '전장(電裝)': 1536, …., '엑소르그룹': 4100, '폐지하고': 5386, '내에': 2784, '기업
으로': 103, '기술력만큼이나': 2682, '업종': 4069, '"신성장': 936, 'BMW-벤츠-아우디가': 2047, '3년의': 1936,
'‘마크': 2174, '공급사슬을': 2516, '텔레매틱스와': 5278, '유가증권시장에서': 4308, '낼': 195}
```

위는 각 단어들에 대해 count을 진행 한 후 그 결과를 기반으로 각 고유 단어에 대한 numbering을 실시한 결과이다. 다시 말해 **가장 count가 많이 된 단어를 시작으로 '0', '1', '2', ... 의 값을** 지정해주게 되고 그 결과는 위의 3번째 출력 결과와 같이 Dictionary 형태로 나타나게 된다. 이제 각 고유 단어에 지정된 number들을 실제 학습 데이터에 적용하도록 하자.

실행 코드

```
# 학습 및 테스트 셋에 앞에서 만든 사전을 적용하여 고유번호를 부여합니다.
corpus_train = np.array(list(map(vocab.get, data_train)))
check_len = 30
print ("\n'corpus_train' looks like %s" % (corpus_train[20:check_len]))
```

출력 결과

```
'corpus_train' looks like [   3   18   67   19  123 2518 1593  116 5640   31]
```

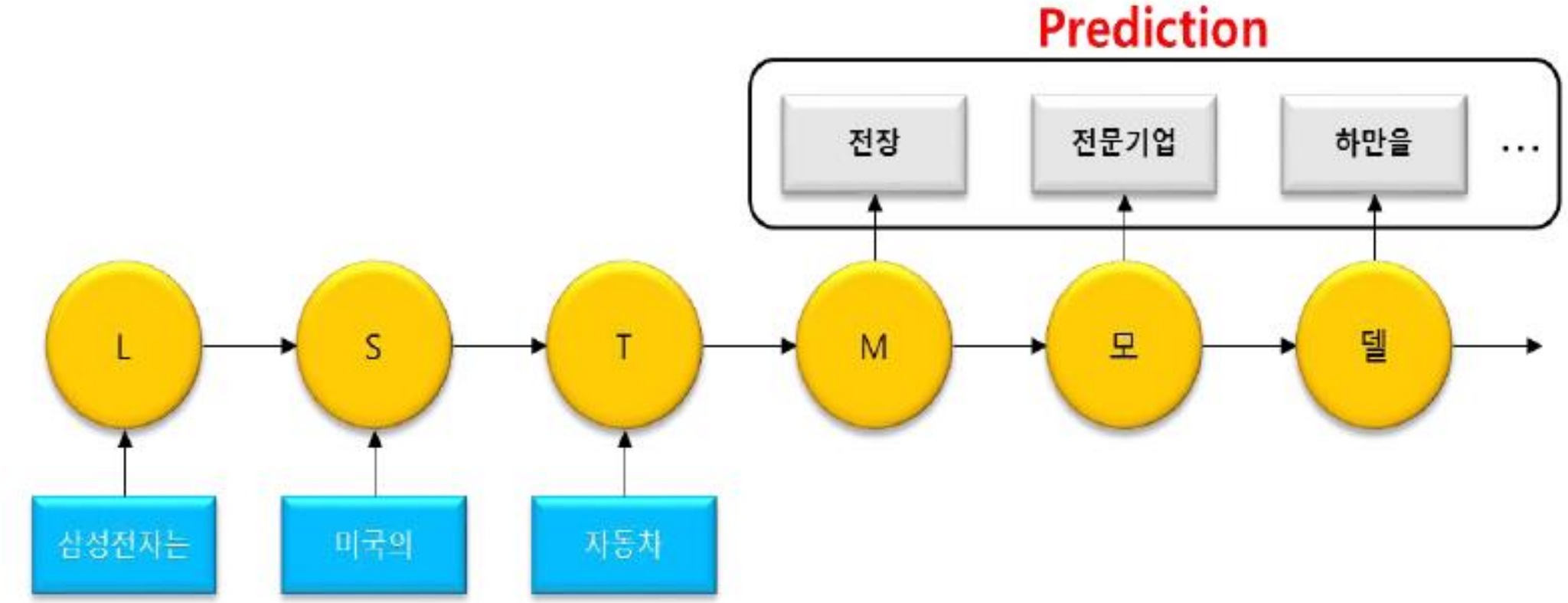
위와 같이 학습데이터가 모두 고유 단어에 지정된 'number'로 변환되었다. 이러한 데이터를 기반으로 배치 데이터를 생성해보자. 배치 데이터 생성의 경우 '고급 분석 민원' 과는 다르게 별도의 테스트 셋은 생성하지 않는다. 이는 본 실습의 목적이 '취합된 형태의 뉴스 생성'이기 때문에 생성된 뉴스 자체가 모델의 성능을 평가하는 지표가 되기 때문이다. 따라서 별도의 테스트 셋 생성 없이 batch_size 및 seq_length를 20으로 하여 배치 데이터를 생성한다.

실행 코드

```
# 배치데이터를 생성합니다.
batch_size  = 20
seq_length  = 20
num_batches_train = int(corpus_train.size / (batch_size * seq_length))


corpus_train_reduced = corpus_train[:(num_batches_train*batch_size*seq_length)]

xdata_train = corpus_train_reduced
ydata_train = np.copy(xdata_train)
ydata_train[:-1] = xdata_train[1:]
ydata_train[-1]  = xdata_train[0]

xbatches_train = np.split(xdata_train.reshape(batch_size, -1), num_batches_train, 1)
ybatches_train = np.split(ydata_train.reshape(batch_size, -1), num_batches_train, 1)
```

```
# 위에서 지정한 batch_size 및 seq_length는 컴퓨터가 한번에 학습할 수 있는 용량을 고려하여 지정한 것으로 그 이상의 의미는 없다.
```

2. 딥러닝을 활용한 보고서 봇 생성



본 실습의 목적은 위 그림과 같이 해당 데이터를 기반으로 **'단어 혹은 문장' 다음에 위치할 '단어 및 문장'을 예측**하는 것이 핵심이다. 따라서 서로 연결되어 있는 언어들의 관계를 가장 잘 표현할 수 있는 분석 기법에 **가장 적합한 'LSTM'으로 모형**을 구현할 것이다.

[모델 파라미터 설정]

실행 코드

```
# LSTM 파라미터 값을 지정합니다.
vocab_size = len(vocab)
rnn_size   = 300
num_layers = 2
grad_clip  = 5.
```

```
# 인공 신경망 구축을 위한 라이브러리 tensorflow를 불러온다.
# LSTM의 state의 개수(rnn_size) 및 계층 개수(num_layers) 그리고 학습 속도(grab_clip)를 결정한다.
```

[LSTM 모델 구축]

실행 코드

```
# LSTM 모델을 정의합니다.
unitcell   = tf.nn.rnn cell.BasicLSTMCell(rnn size, state is tuple=True)
cell       = tf.nn.rnn_cell.MultiRNNCell([unitcell] * num_layers, state_is_tuple=True)
input_data = tf.placeholder(tf.int32, [batch_size, seq_length])
targets    = tf.placeholder(tf.int32, [batch_size, seq_length])
istate     = cell.zero_state(batch_size, tf.float32)

# 가중치 및 편향을 정의하고 Word-Embedding 작업을 수행합니다.
with tf.variable_scope('HangulRnn'):
    softmax_w = tf.get_variable("softmax_w", [rnn_size, vocab_size])
    softmax_b = tf.get_variable("softmax_b", [vocab_size])
    with tf.device("/cpu:0"):
        embedding = tf.get_variable("embedding", [vocab_size, rnn_size])
        inputs = tf.split(1, seq_length, tf.nn.embedding_lookup(embedding, input_data))
        inputs = [tf.squeeze(_input, [1]) for _input in inputs]

# 예측값과 손실 함수를 정의합니다.
outputs, last_state = tf.nn.seq2seq.rnn_decoder(inputs, istate, cell
                , loop_function=None, scope = 'HangulRnn')
output = tf.reshape(tf.concat(1, outputs), [-1, rnn_size])
logits = tf.nn.xw_plus_b(output, softmax_w, softmax_b)
probs  = tf.nn.softmax(logits)
loss = tf.nn.seq2seq.sequence loss by example([logits], # Input
    [tf.reshape(targets, [-1])], # Target
    [tf.ones([batch_size * seq_length])] # Weight
    )
```

```
# LSTM은 Multi-layer로 구축.
# Word Representation Vector(embedding)의 길이는 state의 개수(rnn_size)와 동일하게 지정
# tf.nn.seq2seq.sequence_loss_by_example 메서드를 활용하여 Loss Function 정의
```

[Optimizer 파라미터 설정]

실행 코드

```
# 학습을 위한 매개 변수를 정의합니다.
cost     = tf.reduce_sum(loss) / batch_size / seq_length
final_state = last_state
lr       = tf.Variable(0.0, trainable=False)
tvars    = tf.trainable_variables()
grads, _ = tf.clip_by_global_norm(tf.gradients(cost, tvars), grad_clip)
_optm    = tf.train.AdamOptimizer(lr)
optm     = _optm.apply_gradients(zip(grads, tvars))
```

```
# lr(Learning Ratio)를 설정하여 학습 속도 조절
```

[모델 학습]

실행 코드

```
# 모델을 학습합니다.
num_epochs    = 300
learning_rate = 0.002
decay_rate    = 0.97

sess = tf.Session()
sess.run(tf.initialize all variables())
saver = tf.train.Saver(tf.all_variables())
init_time = time.time()
train_cost_list = []
for epoch in range(num_epochs):
    sess.run(tf.assign(lr, learning_rate * (decay_rate ** epoch)))
    state     = sess.run(istate)
    batchidx  = 0
    for iteration in range(num_batches_train):
        start_time   = time.time()
        xbatch       = xbatches_train[batchidx]
        ybatch       = ybatches_train[batchidx]
        batchidx     = batchidx + 1
        train_cost, state, _ = sess.run([cost, final_state, optm]
            , feed_dict={input_data: xbatch, targets: ybatch, istate: state})
        train_cost_list.append(train_cost)
        train_cost_avg = np.mean(train_cost_list)
        train_perplexity = np.exp(train_cost_avg)
```

```
        total_iter = epoch*num_batches_train + iteration
        end_time     = time.time();
        duration     = end_time - start_time
        train_cost_list
        if total_iter % 100 == 0:
            print ("[%d/%d] train perplexity: %.4f / Each batch learning took %.4f sec"
                   % (total_iter, num_epochs*num_batches_train, train_perplexity, duration))

ckpt_path = os.path.join(save_dir, 'model.ckpt')
saver.save(sess, ckpt_path)
print("model saved to '%s'" % (ckpt_path))
```

```
# num_epochs를 통해 학습 횟수 조절, 총 학습 횟수는 num_epochs *num_batches_train(훈련데이터 배치 수).

# learning_rate 및 decay_rate를 통해 학습 속도 조절.

# train_perplexity를 통해 모델의 혼잡도 측정.

# 뉴스 생성을 위해 saver = tf.train.Saver(tf.all_variables()) 및 saver.save 메서드를 활용하여 모델 파라미터 값을 ckpt_path에 저장.
```

모델 학습의 경우 **딥러닝 이론 교재의 RNN에서 소개된 Penn Treebank와 그 과정이 유사하다.** 다만 본 실습의 경우 학습 된 모델을 기반으로 새로운 형태의 뉴스를 생성해야 하기 때문에 학습된 모델의 최종 파라미터 값들을 특정 Path에 저장하는 부분이 추가되었다.

학습된 모델을 기반으로 새로운 형태의 취합된 보고서를 생성해 보도록 하겠다.

실행 코드

```
load_dir = "/home/eduuser/NN/Data/Report"
with open(os.path.join(load_dir, 'chars_vocab.pkl'), 'rb') as f:
    chars, vocab = cPickle.load(f)
vocab_size = len(vocab)
print ("'vocab_size' is %d" % (vocab_size))
```

먼저 학습에 사용된 고유 단어 및 각 고유 단어에 지정된 Number값을 불러온다. 다음으로는 학습 시 구현한 LSTM모델을 다시 한번 구현하고 추정된 최종 파라미터 값을 불러온다.

실행 코드

```
# LSTM 파라미터 값을 지정합니다.
rnn_size   = 300
num layers = 2
batch_size = 1
seq_length = 1

# LSTM 모델을 정의합니다.
unitcell   = tf.nn.rnn_cell.BasicLSTMCell(rnn_size, state_is_tuple=True)
cell       = tf.nn.rnn_cell.MultiRNNCell([unitcell] * num_layers, state_is_tuple=True)
input_data = tf.placeholder(tf.int32, [batch_size, seq_length])
istate     = cell.zero_state(batch_size, tf.float32)

# 가중치 및 편향을 정의하고 Word-Embedding 작업을 수행합니다.
with tf.variable_scope('HangulRnn'):
    softmax_w = tf.get_variable("softmax_w", [rnn_size, vocab_size])
    softmax_b = tf.get_variable("softmax_b", [vocab_size])
    with tf.device("/cpu:0"):
        embedding = tf.get_variable("embedding", [vocab_size, rnn_size])
        inputs = tf.split(1, seq_length, tf.nn.embedding_lookup(embedding, input_data))
        inputs = [tf.squeeze(_input, [1]) for _input in inputs]


def loop(prev, _):
    prev = tf.matmul(prev, softmax_w) + softmax_b
```

```
    prev_symbol = tf.stop_gradient(tf.argmax(prev, 1))
    return tf.nn.embedding_lookup(embedding, prev_symbol)

# 예측값를 정의합니다.
outputs, final_state = tf.nn.seq2seq.rnn_decoder(inputs, istate, cell
                , loop_function=None, scope = 'HangulRnn')
output = tf.reshape(tf.concat(1, outputs), [-1, rnn_size])
logits = tf.nn.xw_plus_b(output, softmax_w, softmax_b)
probs  = tf.nn.softmax(logits)

# 이미 생성된 모델 파라미터 값을 불러옵니다.
sess = tf.Session()
sess.run(tf.initialize_all_variables())
saver = tf.train.Saver(tf.all_variables())
ckpt  = tf.train.get_checkpoint_state(load_dir)

print (ckpt.model_checkpoint_path)
saver.restore(sess, ckpt.model_checkpoint_path)
```

saver.restore 메서드를 통해 앞에서 학습한 최종 파라미터 값을 불러온다.

모델 구현은 앞장에서 학습한 모델과 그 과정이 같다. 다만 본 장에서는 이미 학습된 모델을 기반으로 새로운 보고서를 생성하기 때문에 또 다시 모델을 학습 할 필요가 없다. 따라서 **모델에 입력되는 Input값은 하나의 단어가 되고 이에 따라 batch_size 및 seq_length 값은 1로 지정한다.** 모델 구현 후 **saver.restore** 메서드를 통해 학습된 **최종 파라미터 값을** 불러온다.

이제 보고서를 생성해보도록 하자.

실행 코드

```
# 이미 생성된 모델을 기반으로 새로운 보고서를 작성합니다.
def weighted_pick(weights):
    t = np.cumsum(weights)
    s = np.sum(weights)
    return(int(np.searchsorted(t, np.random.rand(1)*s)))

# Sample using RNN and prime characters
#prime = "삼성전자는 미국의 자동차"
prime = "삼성전자는 자동차"
#prime = "커넥티드 카 사업을"
```

```
state = sess.run(cell.zero_state(1, tf.float32))
for char in prime.split()[:-1]:
    x = np.zeros((1, 1))
    x[0, 0] = vocab.get(char,0)
    state = sess.run(final_state, feed_dict={input_data: x, istate:state})

ret  = prime
char = prime.split()[-1]
num  = 120
for n in range(num):
    x = np.zeros((1, 1))
    x[0, 0] = vocab.get(char,0)
    [probsval, state] = sess.run([probs, final_state]
        , feed_dict={input_data: x, istate:state})
    p      = probsval[0]

    sample = np.argmax(p)

    pred   = chars[sample]
    ret   += ' ' + pred
    char   = pred

print (ret)
```

np.argmax(p) 및 chars[sample]을 통해 문장의 다음에 위치할 단어를 예측한다.

num = 250은 생성되는 뉴스의 단어 수를 뜻한다.

보고서 생성하기 위해서는 기본적으로 초기 정보가 필요하다. 다시 말해 최소한 학습에 사용된 단어 2~3개 단어가 필요하고 모델은 이러한 초기 정보를 바탕으로 뉴스를 생성하게 된다.

본 실습에서는 **1) '삼성전자는 미국의 자동차', 2) '삼성전자는 자동차', 3) '커넥티드 카 사업을'** 와 같이 총 3가지 초기 정보로 보고서를 생성하였다. 이 외에도 사용자가 관심이 있는 부분의 단어들로 얼마든지 초기 정보를 구성할 수 있다.

아래는 3개의 초기 정보를 기반으로 생성된 보고서이다.

출력 결과

####1)삼성전자는 미국의 자동차####
삼성전자는 미국의 자동차 전장 전문기업 하만(Harman)을 9조원대에 인수한다는 소식에 투자자들은 두 기업의 사업 시너지에 큰 기대감을 보였다. 15일 금융투자업계에 따르면 삼성전자는 하만을 통해 사업을 한층 본격화할 것으로 관측된다. 삼성전자는 그간 성장세가 둔화되고 있는 스마트폰 분야를 넘어 새로운 분야로 진출하기 위해 여러 인수 작업을 펼쳐왔다. 애플 시리의 개발진이 설립했던 인공지능 기업 '비브 랩스'(Viv Labs), IoT용 클라우드 서비스 공급사 조이언트(Joyent)를 인수했던 것이 대표적이다. 하만은 오디오 분야의 명가로 널리 알려진 기업으로, 헤드폰 및 블루투스 스피커, 가정용 하이파이 시스템 등의 제품을 공급하고 있다. 주요 브랜드로는 JB. AKG, 하만 카돈 등이 있다. 기업 매출의 2/3가 오디오 전자 분야에서 생성되며, 특히 차량용 인포테인먼트 매출이 77%에 달하는 등 자동차용 오디오 분야에서도 뚜렷한 존재감을 확보하고 있다.

####2) 삼성전자는 자동차####
삼성전자는 자동차 전장 전문기업 하만(Harman)을 9조원대에 인수한다는 소식에 투자자들은 두 기업의 사업 시너지에 큰 기대감을 보였다. 15일 금융투자업계에 따르면 삼성전자는 하만을 통해 사업을 한층 본격화할 것으로 관측된다. 삼성전자는 그간 성장세가 둔화되고 있는 스마트폰 분야를 넘어 새로운 분야로 진출하기 위해 여러 인수 작업을 펼쳐왔다. 애플 시리의 개발진이 설립했던 인공지능 기업 '비브 랩스'(Viv Labs), IoT용 클라우드 서비스 공급사 조이언트(Joyent)를 인수했던 것이 대표적이다. 하만은 오디오 분야의 명가로 널리 알려진 기업으로, 헤드폰 및 블루투스 스피커, 가정용 하이파이 시스템 등의 제품을 공급하고 있다. 주요 브랜드로는 JB. AKG, 하만 카돈 등이 있다. 기업 매출의 2/3가 오디오 전자 분야에서 생성되며, 특히 차량용 인포테인먼트 매출이 77%에 달하는 등 자동차용 오디오 분야에서도 뚜렷한 존재감을 확보하고 있다

####3) 커넥티드 카 사업을####
커넥티드 카 사업을 전격 미국의 오디오 부문에서 좌우된다는 점을 고려하면 이번 하만그룹 인수는 좋은 전략"이라며 "하만이 보유한 기술력은 향후 삼성전자가 준비할 만물인터넷(IoE) 전략에도 다양한 활용이 가능할 것"이라고 전망했다 삼성전자가 미국 자동차 전장(電場) 부품 전문기업인 하만(Harman)을 80억달러(약 9조3천800억원)에 인수한다. 국내 기업의 해외기업 M&A(인수합병) 사상 최대 규모다. 단순히 M&A 금액보다는 차세대 스마트카의 종착역인 '커넥티드 카' 시장에 미칠 파장이 더 주목된다. 이번 인수를 통해 삼성전자는 자사의 TV, 스마트폰은 물론 VR, 웨어러블 등 고유의 강점에 하만의 기술과 브랜드를 자유롭게 적용할 수 있게 됐다. 특히 신수종 사업으로 육성하고 있는 전장사업에 날개를 달게 됐다. 지난해 말 전장사업부를 꾸리긴 했지만 안전성이 중요한 자동차 부품의 특성상 판로 개척에 애로를 겪으며 가시적인 수주 실적을 거두지 못한 상태에서, 단숨에 수많은 완성차업체들과의 거래선을 확보하게 된 셈이다.

1)과 2)의 경우 **초기 정보가 흡사하기 때문에 거의 유사한 보고서를 생성**한 것을 알 수 있다. 본 실습에서는 약 60여개의 뉴스 기사를 학습 데이터로 사용하였다. 따라서 생성된 보고서 역시 뉴스 기사와 흡사한 형식으로 출력된다. 만약 다른 형태의 보고서를 원한다면 학습 데이터 역시 그에 맞게 변경하면 된다. 또한 이보다 더 많은 학습 데이터를 사용한다면 위의 결과보다 더욱 자연스러운 뉴스 형태의 보고서를 생성 할 수 있을 것이다.