ふんわり掴むAttention ~vol2.5~

埋め込みの獲得

埋め込みとは

テキスト、カテゴリ変数などはそのままNNに入力することはできない！

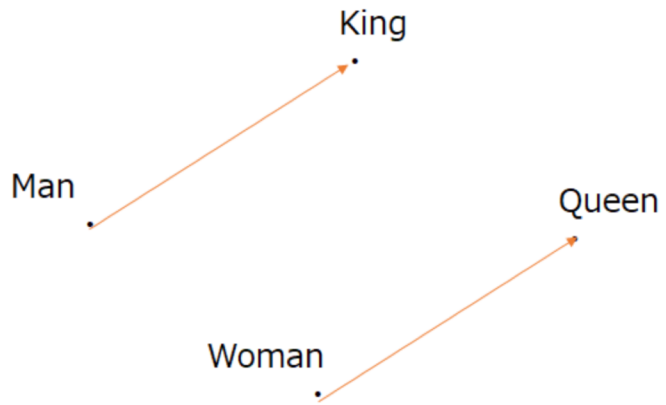
…なんとかベクトルの形にしたいがone_hotだとテキスト間の関係をそんなにうまく表現できないことがあるor　nuniqueが大きいとonehotは疎になる…

→うまく密なベクトルにしたい！！が埋め込み

手法1：word2vecをもちいる

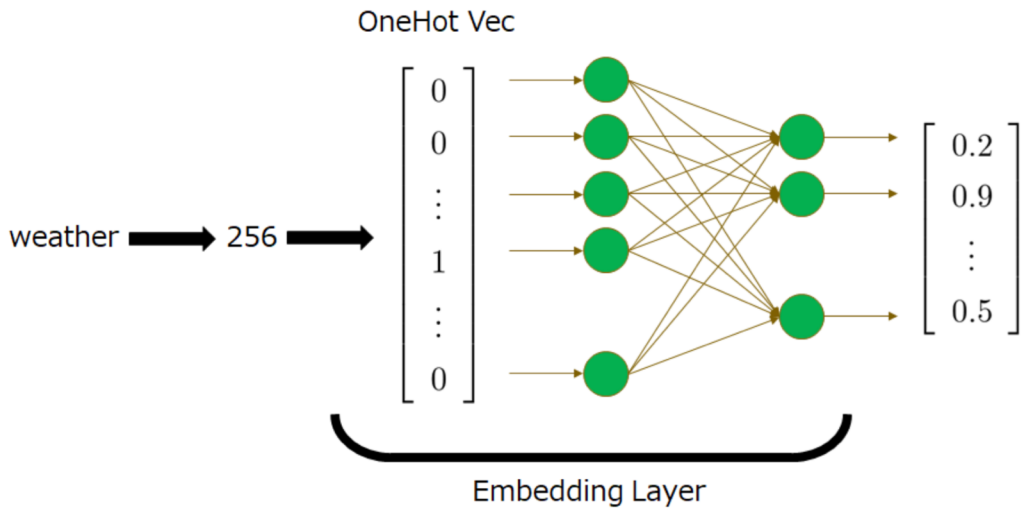
- w2vのお気持ちはテキスト同士がどんな感じで出現するか(**同時に出てきやすさ**)を学習させることで**関連をうまくとらえよう**というもの
- **テキストに限らず使用可能！！（後述）**
- テキスト同士の関連を表現できたベクトルを獲得すれば

“King”-“Man”=”Queen”-“Woman”みたいなことが可能



手法2：Embedding層をもちいる

- テキストなどをLabel encordingしたものを入力として「Onehotに→重み行列をかけて密なベクトルを得る」という操作をしている、重み行列が学習可能



手法3：BERTを用いる

- BERTの事前学習モデルを使用して文章、単語を埋め込んでしまう
- テキストの埋め込みの初手はこれらしい(元NLPer談)
- Transformersを使用、様々な言語、ドメインに対して特化した重みも探せばある(医療系もpubmedで事前学習したものが存在)

手法4：Universal Sentence Encoder

- テキストに限る
- Tfで使用可能
- 多言語でもお構いなく埋め込みをすることが可能

…同一ベクトル空間に全てのテキストを埋め込むことができる　のがよい(next)

その他にはswem,tfidf,LDA,SWEM_max,BM25,SCDV,テキストに対して学習済みモデルでSentiment Classificationして予測結果のlogitsをとる…

これらで獲得したembeddingを次元圧縮してLGBMの入力とする、次元圧縮せずNNに突っ込むなど可能

テキストを埋め込む時、言語ごとに埋め込むべき？

- BERT_baseは英語で学習されているもので別言語ではうまく機能しない
- 多言語の場合は❶英語に全て翻訳❷言語ごとに分けて、各言語のBERT で学習❸多言語対応のBERTで学習
- が考えられるが、❷はよくなさげ

…atma10では言語ごとに分けていたが多言語対応BERTにへんこうしたlate subによりCV,LB共に大きく改善した

すごいぞw2v…非言語にも使用可！！

- Atma6ではやりとりが時系列データとして与えられた

…「やりとり」はグラフ構造を示すキーワード、時系列とw2vは相性が良い　などからkeyとなったのはやりとりした側された側のidをw2vで埋め込むこと　であった

→[やりとりした側についてgroupby,された側を時系列順に並べて重複除去]をtextとしてw2vで学習

→[やりとりした側、された側]でtextとしてグラフ構造を埋め込み

グラフ埋め込み

- 1つ前のスライドでは時系列性を考慮した埋め込み、とグラフ構造を考慮した埋め込み　を1つずつ紹介した

- グラフ埋め込みはもう少し色々存在
- ex)deep walk,node2vec…
- Deep walk:各ノードからk歩だけ移動した時の遷移経路の履歴を“text”としてword2vecに入力(前のスライドではk=1)
- Node2vec:次ノードへの遷移確率を完全ランダムからパラメータを使って調整できるように

参考

- https://agirobots.com/word2vec-and-embeddinglayer/
- https://qiita.com/kenta1984/items/9613da23766a2578a27a
- https://www.guruguru.science/competitions/16/discussions/fb792c87-6bad-445d-aa34-b4118fc378c1/
- https://recruit.gmo.jp/engineer/jisedai/blog/node2vec/
- https://www.guruguru.science/competitions/16/
- https://radimrehurek.com/gensim/models/ldamodel.html
- https://nykergoto.hatenablog.jp/entry/2019/02/24/%E6%96%87%E7%AB%A0%E3%81%AE%E5%9F%8B%E3[…]se_Composite_Document_Vectors_%E3%81%AE%E5%AE%9F%E8%A3%85
- https://github.com/upura/nlp-recipes-ja/tree/master/examples

補足　w2vの2通りの使用

- [ベクトル獲得といった手段という観点で言うと大きく分けて 2 つ使用法があります。w2v は 周辺の単語との関係性 (文脈) からその単語の意味を表現するように学習するので (重要) 文書集合が異なれば、同じ単語であっても異なるベクトルを得ることができる。]そうです。
- →事前学習済みモデルを使用 (wikipedia などで学習されてます, 今回未使用)
- スクラッチで学習 (コンペで使用されている語彙のみを使用)

の2通りを考える！！(w2vmodel.wvを渡す関数を持っておく)

補足：グラフ埋め込み

- 何を使えば良いかについてまとまっている論文をまず参照し、どれかを使用
- 前述の2つは実装が比較的楽であった
- Node2vec(著者実装)

```
###  random_warkの遷移の確率をパラメータp,qによって制御する。
### 小q大でbfs(周辺みる)、逆の時dfs(深く繋がりを追う)
def make_node2vec(G, num_walk, length_of_walk,p=1,q=1):
    ## num_walkは何周試行するか,データ数が大きくなる
    ## p,qはnode2vecのパラメータ  https://recruit.gmo.jp/engineer/jisedai/blog/node2vec/
    ##p小q大でbfs(周辺みる)、逆の時dfs(深く繋がりを追う)
    #ランダムウォークで歩いたノードを入れるlistを生成
    paths = list()
    node_list = df_network["col1"].unique()
    for i in range(num_walk):
        for node in node_list:
            now_node = node
            ### スタートとlength_of_walk歩  歩いたとき  到着したノード(length_of_walk+1)がpathに入る==========
            #到達したノードを追加する用のリストを用意する
            path = list()
            path.append(str(now_node))
            for j in range(length_of_walk):
                if j ==0:
                    next_node = random.choice(list(G.neighbors(now_node)))
                    path.append(str(next_node))
                    pre_node = now_node
                    now_node = next_node
                else:
                    pre = list(G.neighbors(pre_node))
                    now = list(G.neighbors(now_node))
                    dtx_0 = list(set(pre_node)& set(now))
                    dtx_1 = list(set(pre)& set(now))
                    dtx_2 = list(set(now) -set(pre)& set(now)-set(pre_node))
                    choice_prob = np.array([*[1/p]*len(dtx_0),*[1]*len(dtx_1),*[1/q]*len(dtx_2)])
                    choice = [*dtx_0,*dtx_2,*dtx_1]
                    next_node = np.random.choice(choice, p=choice_prob/sum(choice_prob))
                    pre_node = now_node
                    now_node = next_node
                    path.append(str(next_node))
            #ランダムウォークしたノードをリストに追加
            paths.append(path)
    return paths
node2vec_wark = make_node2vec(G, 50, 3)
```

補足　w2vの説明可能性

- 浅いNNに入れて隠れ層を取得、なので説明可能性は諦めるべし
- 説明可能性が高いモデルと低いツヨツヨモデル2つ作るのがだいじなのだ！

最後に..自分のコンペ中と後のメモ

```
sub = pd.read_csv(SUB_PATH)
sub["likes"] = y_test_sub
sub.loc[sub.likes <= 0,"likes"] = 0
sub.to_csv(SAVE_TEST_SUB_PATH,index=False)
### ================


###CV:1.060739210025097...araiさんの続き+TE(logとってから)+sub_title_len(010)
#+material(binary) (017)
#+size h wが大事っぽい(019)
#+説明の長さ(021)...長さ0なら人気ないのを拾えてそう
#+場所(複数ある場合は一つ目だけをついか)(022)(LE)
#maeにすれば外れ値拾える？？...CV:1.0640713503344634  わけでもない?...
##tilte dim5:1.0355546292357918:030.csv
## title 5dim:1.039737618192661
## +説明5dim:1.0381281250577157

####
#1:bertは多言語で同じモデル使用...1.0256922026518511
#..固有値＋..1.0270574321525245..遅い
#..USE+svd(20)...1.0288857329459224,dim5:1.0270045436374424,dim2:1.0282328498394
56
#...not work
#2:色はいろんな空間をとる....1.0252187781415212
##tfidf_svd(title)...1.0244204029832693
## +YIQ 1.0226137070791068
## +tfidf(decri) 1.0194040833033788,1.0186121758812734,1.0186248718529725
## +bm25 (title,decri)1.013319066891187
## makerのcoount vec 1.014466336064767
## w2v 1.0126476922441185
## hex_w2v:1.0011117698319736
## historical person 1.0012423903861372
## swem meanとmaxの併用は不要説
## scdvはちょい時間かかる程度、気軽に使えそう(どれか選ぶ  で良いかも)
```

効いた実験についてはdirごとコピーして次実験へ
効いてない実験に関しては前の事件をコピーして次実験へ
を繰り返した。埋葬したアイデアも多少はある