NLP HW 2
Sense Embeddings

Karim Ghonim - Matricola: 1774086

05 June 2019

1 Dataset

The EuroSense - high precision corpus served as the base of the experiments. The EuroSense corpus contained ≈ 1.9M sentences. The SEW corpus was later used in order to improve the sense embeddings to great results. The SEW corporus contains ≈ 4, 2M wikipedia pages. The conservative version of SEW was used as it includes only one sense annotation per tagged mention and no overlap. Both datasets are multilingual sense-annotated with only the english sentences taken into consideration throughout the project, and automatically built which lead to many annotation problems that are discussed below. The iterparse function from the *lxml.etree* library was used in order to iterate through and parse the dataset as it was too big to load in memory. Both datasets were preprocessed in the same manner for consistency in how the sense embeddings and their context will be created.

1.1 Preprocessing

- Removing all punctuations, stop-words, two-letter or less words, and non-ASCII as they add no information to the context surrounding the target word with regards to this application and solely act as noise in the dataset, specially when adding more data (SEW),
- Replacing all numbers by a single token (!), thus removing the sparcity caused by having a vector for every date, hour, price, but still incorporating the information that there is a number present in the context of the sense in question
- Converting all words to lowercase in order to have a single vector for each sense regardless of where its anchor occurs in the sentence. The problem that for example ”Bush” and ”bush” shouldn't be represented by a single vector will later be handled when adding the synset

1.2 Sense representation

Each sense was represented as <**lemma**>_<**synset**> with the lemma converted to lowercase, with the anchor acting as the key in order to retrieve its respective <**lemma**>_<**synset**> pair from the annotations. In order to constrain the size of the embedding matrix, only BabelNet synsets that occur (have a match) in WordNet were taken into consideration. This resulted in an increase in training speed and in the same time not losing any important or frequently occuring senses.
As the dataset was automatically annotated, several mistakes in the annotations were present. Per the advice of the creators of EuroSense, for the high-precision corpus, priority was given to longer anchors, consisting of up to 4 words, the spaces or dashes between tokens in the anchor were replaced by ”_”, e.g ”European union” becomes ”european_union_bn:1”, and ”European-union” becomes ”european_union_bn:2”. Also, when an anchor is repeated in the same sentence with each occurance having a different sense was handled, as the corpus was annotated based on order of occurance, therefore it was possible to track which sense belongs to which anchor.

2 Network architecture

2.1 Data-streaming

Since the entire dataset would saturate the RAM, an iterator was created in order to stream data (the sentences) to the model, therefore being able to make use of the entire corpus at hand.

2.2 Word2Vec

The word2vec context based models Skip-gram and CBOW implemented in the gensim framework were used due to the many features the framework offers. A grid-search was conducted to obtain the best parameters for each experimentation phase. The Skip-gram model takes much longer time to train, but since the task at hand only required learning the sense embeddings, I tried to train only the sense embeddings and disregarding any word embeddings during the training phase, which increased the speed of the training process. However, CBOW consistently provided improved results for all the experiments conducted. Also hierarchical softmax or negative sampling were used as alternatives to the softmax function in order to overcome the problem caused by its computational complexity. Hierarchical softmax showed a dramatical change in computational complexity and the number of operations needed for the algorithm. This is done through the usage of the binary tree, where leaves represent probabilities of words, thus bringing down the evaluation from O(V) to O(log_2V). It proved to be faster and achieved improved and more stable results than the negative sampling which instead of changing all of the weights each time, taking into account the thousands of observations possible, uses only n of them. For all experiments the frequent word sub-sampling used was 10^{-3}, they were trained for 50 epoches with a learning rate of 0.25, as they provided the best results through the grid-search.

2.3 Experiment Phase A - EuroSense

For the first phase of experimentation, only the EuroSense corpus was considered. I used a window size of 10 as the dataset was already quite small so I wanted to incorporate as much contextual information as possible without taking into consideration words which may have been out of context pre-preprocessing of the data. The highest correlation was achieved with closest cosine similarity, being 0.33 with closest cosine for clean data and 0.30 when keeping the stop-words and not substituting the numbers with a single token. This was obtained with embedding size 500 and negative sampling set to 5.

2.4 Experiment Phase B - EuroSense SEW

For this phase, I integrated both datasets which improved the correlation significantly, reaching a correlation of 0.601. For this phase, I decreased the window size to 5 as SEW was extremely noisy so I preferred to consider only closer words. An embedding size of 400 was used with hierarchical softmax in order to achieve the highest correlation, with the same parameters reaching only 0.54 when trained on noisy data (similar to phase A).

2.5 Test

In order to test the trained sense embeddings, two similarity measurement strategies were implemented, being the weighted and the closest similarity. For both, I tested using the Tanimoto distance as well as the cosine distance as shown in the figures below. The scores were then compared to the gold standard set by humans in the WordSimilarity-353 dataset via the Spearman correlation.

3 Results

Even though there was a significant increase in correlation after adding SEW to the training data, which shows that EuroSense wasn't enough to generalize and overcome the bias within itself (is too domain specific), the results of the embeddings still reached a bottle-neck even though more documents from SEW were added. The best correlation of 0.6 was already reached when using only 1M files from SEW out of the entire 4.2M. This means that other improvements should be implemented in order to overcome the bias in the data and achieve human-like performance regarding this task. An interesting result was that when I subtract man_bn:00053096n from king_bn:00024097n, then add woman_bn:00034016n, the closest vector is queen_bn:00065644n. Also, as see in table 4, tiger and tigers have different vectors even though they have the same synset, meaning that further processing be done on the annotations. Even though the closest cosine similarity achieved the highest correlation during phase B, the weighted cosine similarity got much higher correlation during phase A, showing that with less data it may be better to take into consideration all senses of a word when computing the similarity. Finally, as can be seen from the t-SNE plots provided, the network is able to predict similar senses correctly as well as cluster them together accurately.

	Closest		Weighted	
Embedding Size	COS	Tanimoto	COS	Tanimoto
350	0.271	0.230	0.359	0.340
400	0.280	0.246	0.350	0.326
450	0.288	0.257	0.368	0.329
500	0.301	0.241	0.370	0.376

Table 1: Spearman Correlation - Phase A

	Closest		Weighted	
Embedding Size	COS	Tanimoto	COS	Tanimoto
350	0.584	0.586	0.563	0.566
400	0.600	0.598	0.547	0.536
450	0.585	0.591	0.531	0.544
500	0.593	0.593	0.537	0.35

Table 2: Spearman Correlation - Phase B

Bank	bank_bn:00008363n		**Bank**	bank_bn:00008364n	
Similar Sense	COS	Tanimoto	Similar Sense	COS	Tanimoto
estuary_bn:00031676n	0.368	0.215	banks_bn:00008364n	0.813	0.674
stream_bn:00074588n	0.342	0.195	central_bank$_{b}n : 00017175n$	0.620	0.449
metre_bn:00052501n	0.340	0.197	banking_bn:00008412n	0.616	0.444
erosion_bn:00029547n	0.325	0.192	commercial_bank$_{b}n : 00020991n$	0.598	0.417
channel_bn:00017684n	0.324	0.186	credit_bn:00022100n	0.577	0.399

Table 3: Ambigious words and their closest senses

Tiger	tiger_bn:00060436n		**Italy**	italy_bn:00047705n	
Similar Sense	COS	Tanimoto	Similar Sense	COS	Tanimoto
tigers_bn:00060436n	0.368	0.215	france_bn:00036202n	0.732	0.577
bear_bn:00009342n	0.342	0.195	germany_bn:00026684n	0.691	0.528
lion_bn:00049156n	0.340	0.197	spain_bn:00031605n	0.655	0.487
leopard_bn:00050713n	0.325	0.192	austria_bn:00007266n	0.648	0.469
elephant_bn:00030314n	0.324	0.186	greece_bn:00030401n	0.640	0.460

Table 4: Common words and their closest senses

Psychology	psychology_bn:00065026n		**Drink**	drink_bn:00087321v	
Similar Sense	COS	Tanimoto	Similar Sense	COS	Tanimoto
sociology_bn:00072574n	0.368	0.215	drinking_bn:00012196n	0.390	0.242
social_psychology_bn:00072543n	0.342	0.195	drink_bn:00010183n	0.370	0.226
anthropology_bn:00004584n	0.340	0.197	thirsty_bn:00111852a	0.365	0.121
philosophy_bn:00061984n	0.325	0.192	drinker_bn:00028752n	0.359	0.212
psychology_bn:00065026n	0.324	0.186	alcohol_bn:00002519n	0.357	0.217

Table 5: Common words and their closest senses

Figure 1: Similars senses

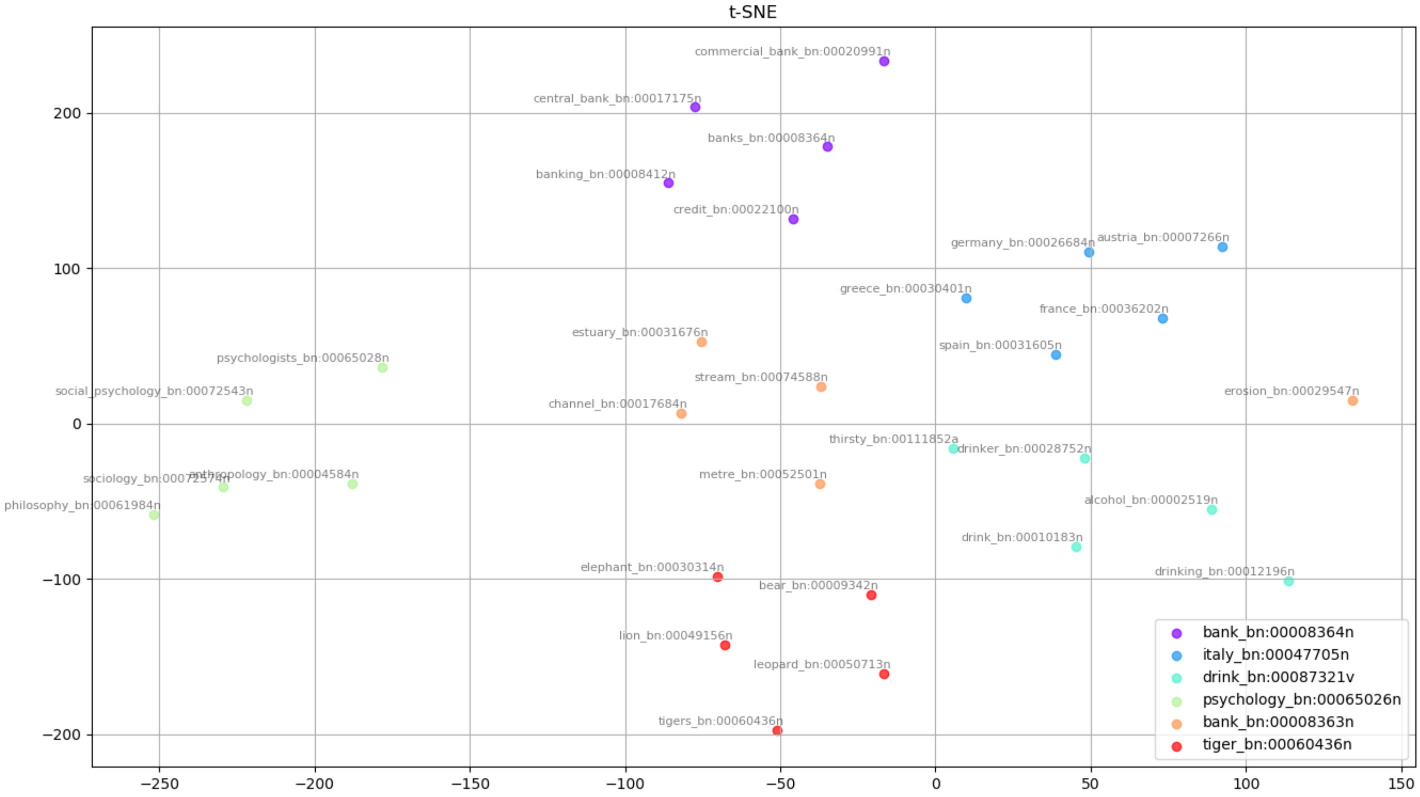


Figure 2: Senses similar to Bank

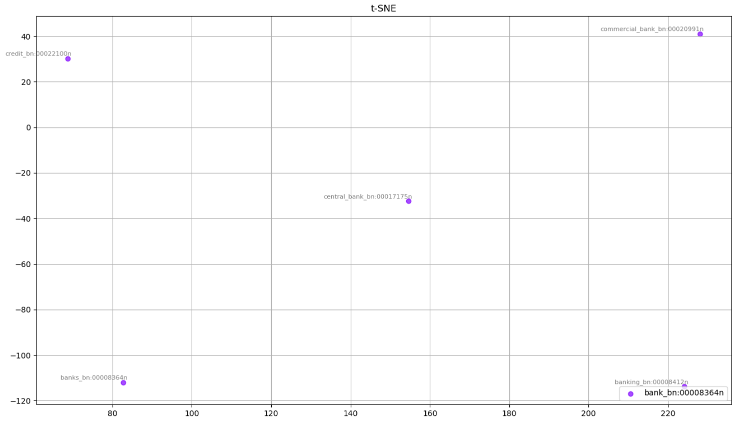
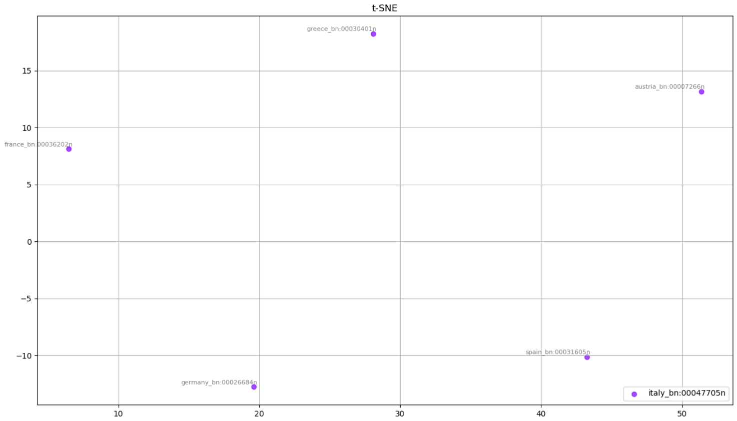


Figure 3: Senses similar to Italy


t-SNE
greece_bn:00030401n
austria_bn:00007266n
france_bn:00036202n
spain_bn:00031605n
germany_bn:00026684n
italy_bn:00047705n