Analysis and Implementation of Sense Embedding

Giada Simionato, 1822614
DIAG, Sapienza University of Rome – Via Ariosto, 25, 00185 Rome, Italy

I. Dataset Preprocessing and Implementation Choices

In this work, to create reliable sense embeddings the *EuroSense* dataset was used, at least in the beginning, in the *high precision* version. Through the parsing of the whole file all the sentences were extracted and all the annotations were organized respectively into a 1D array, where each element is the text and a 3D array where each element is a list of tuples in the form `(anchor, lemma, idSynset)` as to exploit all the information extracted while parsing the EuroSense corpus.

As to build the tensor for the input of the model a customized structure was used: each row of the tensor corresponds to an annotation of the dataset. It has length of $2 \times windowSize + 1$ since it is centered in the `lemma_idSynset` and the other windowSize elements that precede and follow it are the ones, punctuation excluded, of the sentence split by spaces. Whenever the senses were too close to the boundaries of the corresponding sentence, the row was padded as to obtain arrays of the same length. In addition, each row of the tensor contained only one sense among words, to provide a better tool for future analyses and a more focused model into relevant information. Such tensor was then used to feed a *Gensim* model. All the hyperparameters choices and results are deeply explained in the following sections.

In order to assess the performances of the different models the *Spearman correlation*, based on the *cosine similarity* measure, was used. In this work all the couples of words where at least one of them wasn't encountered before, hence deprived of an embedding, were considered as completely opposed with a score of -1. This solution influenced the performances but was chosen as to set a lower bound in the performances as to make objective the evaluation and the comparisons of different implementations.

II. Baseline Model and Hyperparameters Tuning

The model used in this work is provided by *Gensim*. The first trained version was *Continuous Bag Of Words*, by setting the hyperparameter *sg* to *0*. Instead of using a *greedy approach*, hence tuning each parameter at a time and then using the value that gives the best result for subsequent analyses, a *grid search* was employed to provide more consistent results without threatening the monotonicity of the performances. Values has been tested from one to six for the *window size*, and *50, 100, 200, 300, 400, 500, 600, 700, 1000* for the *embedding* one. All these training were performed for *five* epochs, since this value showed to be the most performing one and the obtained results are reported in Table 1. The model was then changed: using *Skipgram* was possible to train, again for five epochs, using *hierarchical softmax* or *negative sampling* that in turn allowed to change the *negative* hyperparameter among the suggested boundaries, hence with the values *5, 10, 15, 20*. The results collected with the former technique are showed in Table 2, the ones with the negative sampling in Tables 3, 4, 5, 6 respectively.

From these results arises that the *Skipgram* model obtained overall greater performances reaching a peak of 0.3729 with a *negative sampling* of *15*, *embedding size 200* and *window size 4* where *CBOW* had the maximum value of 0.3216 with *embedding size 200* and *window size 2*. Even if the CBOW proved to be faster than Skipgram the distribution drawn from the tables states that the latter provides generally higher scores. After have been set the best performing model as the baseline, in order to analyze the how the model reacts to changes in the input, instead of using the customized and more efficient method that shaped the input tensor around the senses without feeding the model with useless information, it was used a version that substituted for each anchor its sense without removing exceeding words and without prevent overlapping in the window-sized spaces of each sense. This way to handle information slow-down at least by three times the whole process of building the tensor and training the model and the results obtained are considerably lower than the one of the original method obtaining a Spearman score of 0.2017, hence a reduction of 45.91% with respect to the current baseline.

III. Inconsistencies Analysis

In order to exploit the EuroSense dataset at most and to enhance the performances, in this work was performed an analysis likely to discover, and eventually eliminate, all the sources of inconsistencies (S.o.I.s) found in the dataset. All those that turned out not to be solvable automatically, were also considered as a good starting point for further enhancement of the dataset. The first reason why data was discarded from the training of the model was that `anchors` in the annotations, drawn from the XML file, weren't referring to any word in the corresponding sentence. As it's possible to notice in Fig. 1a this category contains the 11.772% of the annotations. Different motivations for this were analyzed as showed

in Fig. 1b: only the 1.282% of the anchors that are not in the sentences are due to an upper-lower mismatch between them. Another reason is that there was clearly a mistake in the annotation process since the 20.689% of the anchors that are not in the sentences are anyway part of them[1]. After have been sampled these inconsistencies it was evident that even if all the annotations were labeled as pertained to the English language, they belonged to other languages as pictured in Fig. 1c. Most of them were Spanish, and this lead to the conclusion that most annotations were swapped due to the resemblance between the ISO abbreviations[2].

Among the valid annotations, as showed in Fig. 1a, the 53.5% have the lemma that is annotated with a wrong synset id. To overcome this problem were individuated several sources of inconsistencies: a 10.827% was due to an upper-lower mismatch between the synset and the lemma, a 13.398% was due to the missing underscore symbol as the separator, instead of spaces, between multiple words composing a lemma. An interesting 1.3759% of the lemmas where annotated with the synset id of another lemma in the same sentence. Some words were sampled from the remaining category discovering that nevertheless the lemma and the synset where different, they were generally as it should be, representative of the same meaning[3] and for this reason no further modification were introduced in the model.

In order to fix all these S.o.I.s the customized method for tensor building was enhanced by converting all the sentences, anchors and lemmas to lower case, substituting the spaces with underscores as to match the correct format and all the elements candidates for being part of the rows were tested for the validity, hence not to be punctuation symbols, digits, null words, normal and long stopwords and to be valid one or two character length words. However, at least in the first place, all these changes decrease the performances over the baseline as collected in Fig. 2a. Taken this new version, more respectful of the standards, as the new baseline, it was trained for different numbers of epochs as to find the correct stopping point in the training and the results obtained are showed in Fig. 2b: training for five epochs still confirm all the previous tests and legitimate again the usage of this value for the training performed as explained in Section II.

In order to highlight which changes influenced the most the model in Fig. 2b are collected the results of the baseline model without handling of some features: in the first row there is the model with the same parameters of the baseline without capital letters reduction, nor in the model, nor in the words used for evaluation purpose while in the second the latter was handled. In the last row its possible to notice the impact of not checking whether a word is a stopword as the performances decreased with respect to the baseline.

[1]For example the anchor was *problem* but in the sentence there was *problems*.

[2]*ES* for the Spanish language and *EN* for the English one.

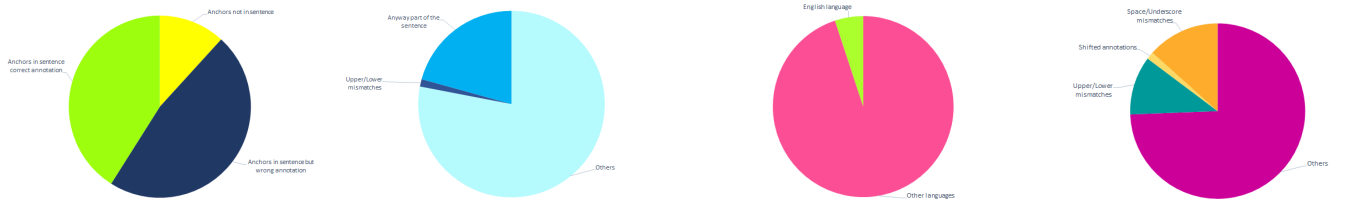
[3]For example *stay* and *remain*, *crime* and *illegal action*, *dispute* and *conflict* etc.

IV. Resources Augmentation and Best Model

In order to improve the quality of the sense embedding model in addition to the EuroSense was added the *Semantically Enriched Wikipedia* (Sew) dataset in the *conservative* version. Due to the huge dimension of this resource only a part was used: after the parsing of the XML files and the collection of the sentences and the annotations like was done with ES, these latter were analyzed as to discard incomplete or wrong annotations and then used for the building of the structured tensor also respecting the format and the solutions of the S.o.I.s found in Section III. In order to obtain better results due to the usage of more data the model trained was switched to *CBOW*, taking also advantage of its faster behaviour while maintaining the embedding size at 200 and the window size at 4. Using only the first 30 folders of Sew and training the model only with this dataset was obtained a score of 0.39085, with the addition of other 10 folders and the EuroSense it grew at 0.4008 and at the end adding others 20 folders, reaching up a total of 60s (each of them containing a high number of XML files), the same model obtained 0.4534. Even if it was proved the increasing trend in the performances with the addition of more data, this was the maximum that was possible to reach with the resources available for this work.

In order to visualize the behaviour of the model in assigning the embeddings to the senses, a *Principal Component Analysis* was firstly used to reduce the dimensions from 200, hence the embedding size, to a tridimensional space. This strong decrease led to a distortion of the output of the model, making impossible to show the principle behind this work. According to the most recent techniques, an hybrid approach was performed: in the first step all the 200-sized embeddings where reduced through a *PCA* down to 50 dimensions and then used a stochastic approach, the *t-Distributed Stochastic Neighbor Embedding* (t-SNE), especially indicated when the dimensionality of the dataset is very high and must be reduced into 2D or 3D for visualization. This merged approach gave incredibly better results compared with the ones obtained by the two methods alone. In Fig. 3 three very close sense embeddings are depicted with their positions, while in Fig. 4 two different ones show to be mapped very far from each other. In order to retrieve more information about the behaviour of the model also a 3-Nearest Neighbors analysis was performed: for the sense *internet_bn:00024712n*, *network_bn:00024712n*, *software_bn:00021497n* and *computer_bn:00021464n* were obtained showing that the model reached its goal.

V. Tables and Figures



(a) (b) (c) (d)

Figure 1: Inconsistencies analysis results: (a) composition of the dataset: the percentage of the annotations whose anchor is not in the corr. sentence is 11.772%, while the 53.5% have the anchor that is in the sentence but the lemma is annotated with a wrong synset id. Finally the $34,728\%$ are consistent annotations; (b) reasons why an anchor is not in the corr. sentence: 1.282% for upper-lower mismatch, 20.689% are anyway part of the sentence and the remaining part is due to other reasons; (c) languages of not valid annotations belong to: 5.068% English and 94.932% others; (d) reasons why the lemmas are annotated in a wrong way: 10.827% due to upper-lower mismatch, 13.398% due to the missing underscores to replace spaces in lemma composed by multiple words and 1.3759% for annotations that were shifted inside the same sentence.

	Window Size					
Embedding Size	1	2	3	4	5	6
50	0.2607	0.2645	0.2835	0.2680	0.2261	0.1897
100	0.2487	0.3133	0.3057	0.2518	0.2062	0.1713
200	0.2146	**0.3216**	0.2874	0.2273	0.1766	0.1514
300	0.2203	0.3033	0.2952	0.2278	0.1736	0.1350
400	0.2230	0.3093	0.2980	0.2156	0.1618	0.1271
500	0.2103	0.3183	0.2806	0.2121	0.1655	0.1222
600	0.2136	0.3132	0.2969	0.2113	0.1551	0.1176
700	0.2118	0.3073	0.2907	0.2161	0.1486	0.1231
1000	0.2180	0.3141	0.2848	0.2070	0.1504	0.1176

Table 1: Spearman scores of the Continuous Bag Of Words model trained for 5 epochs on the EuroSense dataset without having fixed all the inconsistencies and with the input tensor centered around the senses.

	Window Size					
Embedding Size	1	2	3	4	5	6
50	0.2005	0.2960	0.3044	0.3009	0.2946	0.2911
100	0.2603	0.3073	0.3090	0.2982	0.2790	0.2659
200	0.2494	0.3266	0.3454	0.3195	0.3248	0.3164
300	0.2556	0.3483	0.3491	0.3372	0.3252	0.3197
400	0.2580	0.3362	0.3475	0.3387	0.3195	0.3122
500	0.2516	0.3483	0.3478	0.3423	0.3407	0.3358
600	0.2536	0.3367	0.3392	0.3449	0.3426	0.3381
700	0.2615	0.3299	0.3440	0.3515	**0.3496**	0.3402
1000	0.2566	0.3420	0.3284	0.3144	0.3175	0.3162

Table 2: Spearman scores of the skipgram model with **hierarchical sampling**, trained for 5 epochs as CBOW.

	Window Size					
Embedding Size	1	2	3	4	5	6
50	0.2687	0.3205	0.3454	0.3356	0.3315	0.3302
100	0.2696	0.3550	0.3533	0.3555	0.3578	0.3570
200	0.2260	0.3137	0.3607	0.3596	0.3580	0.3568
300	0.2174	0.3070	0.3177	0.3482	0.3612	0.3538
400	0.2208	0.2883	0.3345	0.3624	**0.3636**	0.3512
500	0.2187	0.2888	0.2785	0.2733	0.2841	0.2468
600	0.2183	0.2854	0.3205	0.3498	0.3311	0.3392
700	0.2192	0.2850	0.3018	0.3428	0.3209	0.3195
1000	0.2220	0.2872	0.2966	0.3245	0.3175	0.3078

Table 3: Spearman scores of the skipgram model with **negative sampling** of **5**, trained for 5 epochs as CBOW.

	Window Size					
Embedding Size	1	2	3	4	5	6
50	0.2868	0.3250	0.3469	0.3350	0.3416	0.3394
100	0.2574	0.3299	0.3452	0.3658	0.3372	0.3400
200	0.2346	0.3200	0.3636	0.3682	0.3625	0.3384
300	0.2268	0.3056	0.3216	0.3571	**0.3713**	0.3422
400	0.2240	0.2867	0.3337	0.3474	0.3541	0.3274
500	0.2256	0.2877	0.3234	0.3410	0.3381	0.3125
600	0.2216	0.2821	0.3200	0.3375	0.3198	0.3111
700	0.2242	0.2861	0.3176	0.3373	0.3245	0.3095
1000	0.2236	0.2749	0.2874	0.2869	0.2916	0.2703

Table 4: Spearman scores of the skipgram model with **negative sampling** of **10**, trained for 5 epochs as CBOW.

Embedding Size	Window Size 1	2	3	4	5	6
50	0.2650	0.3236	0.3412	0.3319	0.3233	0.3188
100	0.2379	0.3336	0.3596	0.3559	0.3540	0.3361
200	0.2159	0.3224	0.3554	**0.3729**	0.3516	0.3480
300	0.2087	0.3124	0.3487	0.3621	0.3646	0.3412
400	0.2079	0.2977	0.3330	0.3482	0.3515	0.3122
500	0.2037	0.2821	0.2974	0.3362	0.3319	0.3204
600	0.2037	0.2811	0.3239	0.3255	0.3177	0.3049
700	0.2027	0.2879	0.3102	0.3299	0.3099	0.2906
1000	0.2046	0.2866	0.2641	0.2887	0.2714	0.2233

Table 5: Spearman scores of the skipgram model with **negative sampling** of **15**, trained for 5 epochs as CBOW.

Embedding Size	Window Size 1	2	3	4	5	6
50	0.2900	0.3159	0.3359	0.3372	0.3378	0.3254
100	0.2657	0.3378	0.3595	0.3453	0.3453	0.3398
200	0.2247	0.3160	0.3466	0.3569	0.3466	0.3410
300	0.2119	0.2985	0.3286	**0.3651**	0.3504	0.3476
400	0.2110	0.2892	0.3284	0.3456	0.3344	0.3209
500	0.2104	0.2891	0.3197	0.3425	0.3469	0.3166
600	0.2111	0.2737	0.3261	0.3303	0.3327	0.3100
700	0.2106	0.2812	0.3102	0.3230	0.3240	0.3112
1000	0.2104	0.2771	0.2800	0.2821	0.2691	0.2530

Table 6: Spearman scores of the skipgram model with **negative sampling** of **20**, trained for 5 epochs as CBOW.

Epochs	Scores
2	0.2439
5	0.3035
15	0.2559
30	0.2074
50	0.1935
100	0.1716

(a)

Versions	Scores
Model capitals + score cap.	0.3729
Model cap. + score no cap.	0.2927
Baseline	0.3035
Baseline no stopwords check	0.2574

(b)

Figure 2: (a) Spearman scores of the skipgram model with **negative sampling** of **20**, trained for different numbers of epochs; (b) First row: the model trained for 5 epochs with the baseline hyperparameters without reducing to lower case it and the score words. In the second row the latter were converted. The third row is the baseline model and the last one shows a decreasing of the performances of 15.19% if the validity check on the element of the tensor rows is not performed.

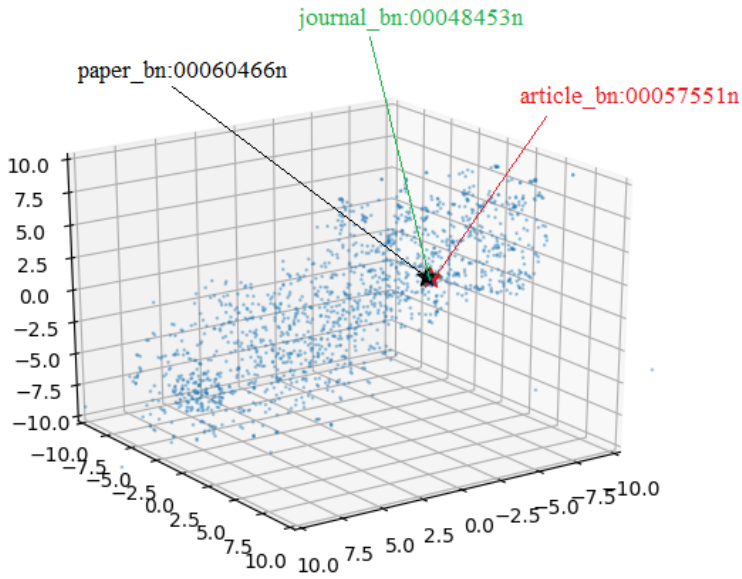


Figure 3: This figure shows the result of the 3D PCA + t-SNE analysis described in Section IV. It proves that the model, when reduced to three dimensions, assigns the closer embeddings to closer senses of words, here it's possible to see that three closest points are strongly correlated. `Paper_bn:00060466n`, `journal_bn:00048453n` and `article_bn:00057551n` are also the three nearest neighbours of the sense `journal_bn:00026887n` .

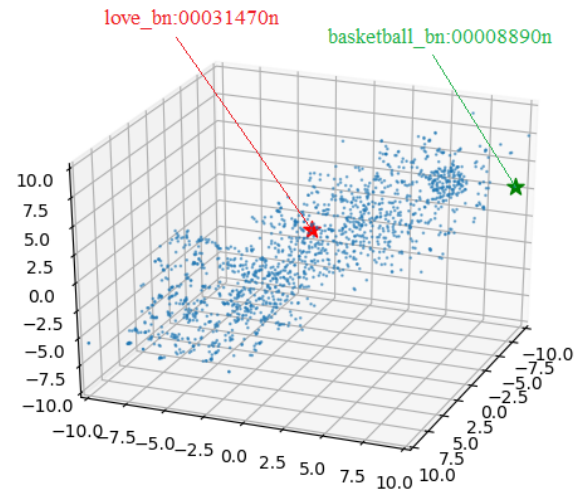


Figure 4: This figure shows how very different senses are embedded into far 3D vectors like in the case of `love_bn:00031470n` and `basketball_bn:00008890n`. This point of view gives the possibility to see clearly the presence of a cluster of points, not too far from `basketball_bn:00008890n`, hence the presence of a concept that can described by a higher number of different senses.