Lbl2Vec : An embedding based approach for Unsupervised document retrieval on Predefined topics

why is this important? What is the key difference between other methods?
Annotation (labelling -> high cost) , have to annotate even document without any correalation

Context : We have a large number of news articles extracted from sports sections of different newspapers, but the metadata of the articles contain no information about their content. We would have to annotate our articles at a high cost, and we might not be interested in any sports apart from the previously specified ones.

How? Uses cosine similarity to measure the similarity between label and document vectors that are embedded in the same feature space. => determine semantic similarity to assign a document to the topic or not

Previous approaches

K-nearest neighbor classification to assign the most likey label to each document / Key word enrichiment + Latent semantic analysis (LSA) / BERT / Term document frequency scores / ad-hoc document retrieval : based on user queries -> user submit a search query and the model ranks set of documents that are most likely to be relevant for user information needs

Approach of Lbl2Vec

Word vectors W and document vectors D and label embeddings L within the same feature space => the distance can be considered their semantic similarity.

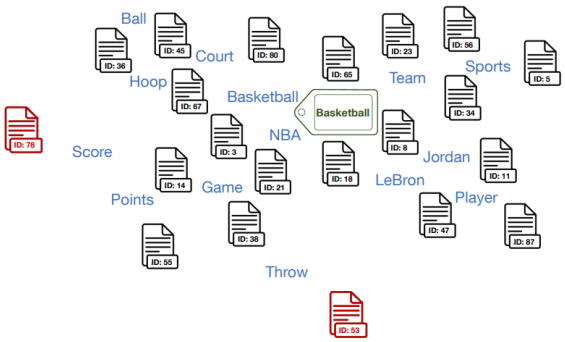

Figure 1: Example illustration of a semantic feature space related to Basketball. Blue: Descriptive keyword embeddings. Black: Document embeddings that are semantically similar to the keywords and each other. Red: Outlier document embeddings. Green: Label embedding.

To train

1. Uses PV-DBOW (Paragraph vector) which only trains document embeddings but not word embeddings. Therefore, uses Skip-gram for word embedding training (predicting word vectors) Iterative training on the interleaved PV-DBOW and Skip-gram architectures enable us to simultaneously learn word and document embedding that share the same feature space.

2. Topic keywords for label embedding. For each topic manually define at least one keyword that describes the topic properly.

3. We calculate the cosine similarity of keyword embedding to each document embeddings and sort the document embeddings in descending order.

4. To only include documents that has high cosine similarities we need to set params additionally. **Similarity threshold** / dmin: minimum number of document embeddings that have to be added to Dci (set of candidate documents) since it can be too restrictive with only small selection / dmax for max number of docs

5. Uses LOF Local Outlier Factor cleaning to identify document embedding with lower local density that their neighbors and remove from Dci.

1. Words, Documents, and Topics: We have a collection of words, documents (texts), and topics. Topics are like categories or themes that group related documents together.

2. Document Embeddings: Each document in the collection is converted into a numerical vector (embedding). This vector represents the semantic meaning of the document, capturing the important information in it.

3. Centroid of Document Embeddings: The centroid is like the **center point of all the document embeddings belonging to a particular topic**. It's calculated by taking the average of all the document embeddings in that topic.

4. Label Embeddings: We **define a label embedding for each topi**c (t(i)). The label embedding represents the **overall meaning of that topic**, combining the meanings of all the documents related to it.

5. Jointly Embedded Semantic Representations: By following this process, we now have embeddings for words (representing their meanings), embeddings for documents (representing their contents), and embeddings for topic labels (representing the overall meaning of the topics).

In summary, this process aims to create a cohesive way of understanding the meaning of words, documents, and topics in a collection. The embeddings allow us to capture and compare the semantic similarities between different elements, which is useful for various natural language processing tasks like topic modeling, document clustering, and information retrieval.

Model training

We convert all document words and topic keywords to lowercase, **tokenize** the documents, **assign IDs** to them, and train an individual model on each dataset. We conduct a short manual hyperparameter optimization prior to training and use the given standard hyperparameters for completely unlabeled datasets.

Also, By adjusting the topic similarity thresholds, we can reduce the number of documents that are truly related to a topic. By doing so, it can sample a small dataset with high precision from a large corpus of documents, and then use this dataset for semi-supervised classification.

r

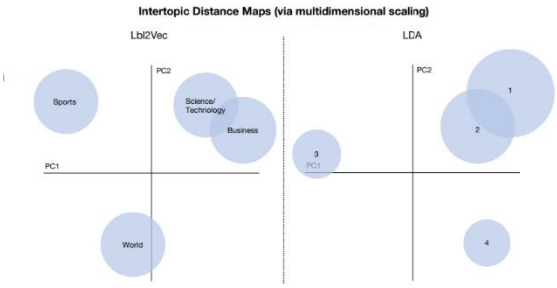

Figure 2: Visualization of Lbl2Vec and LDA topic representation capabilities based on AG's Corpus. Each circle represents a topic, whereas each topic, in turn, consists of

LDA	
Topic 1	oil; crude; prices; microsoft; windows; dollar; reuters; barrel; stocks; yukos;
Topic 2	ccia; thunderbird; generali; macau; cheetham; backman; hauritz; pizarro; rituxan; abdicate;
Topic 3	orton; mashburn; bender; kwame; pippen; attanasio; elliss; icelandair; lefors; stottlemyre;
Topic 4	wiltord; perrigo; quetta; dione; mattick; olympiad; panis; agis; bago; cracknell;

Table 2: Top 10 most relevant terms for each topic of the LDA model; we use the LDAvis relevance with $\lambda = 0.1$.

low us to relate the modeled topics to the AG's Corpus classes. However, from Table 3 we can conclude that

Lbl2Vec	
World	iraq; killed; minister; prime; military; palestinian; minister; israeli; troops; darfur;
Sports	cup; coach; sox; league; championship; yankees; champions; win; season; scored;
Business	stocks; fullquote; profit; prices; aspx; quickinfo; shares; earnings; investor; oil;
Science/ Technology	microsoft; windows; users; desktop; music; linux; version; apple; search; browser;

Table 3: Top 10 most relevant terms for each topic of the `Lbl2Vec` model; we use the LDAvis relevance with $\lambda = 0.1$.

Method	AG's Corpus			20Newsgroups		
	F1	Prec.	Rec.	F1	Prec.	Rec.
KE + LSA	76.6	76.8	76.6	61.0	71.1	57.8
Lbl2Vec	82.7	82.7	82.7	75.1	75.1	75.1
Supervised Naïve Bayes	89.8	89.8	89.9	85.0	87.1	85.4

Table 4: Performance of our Lbl2Vec models when classify

However, it couldn't beat the supervised document classification.