thesis

agert

June 2018

On the web there is a large volume of raw text data, e.g. Reviews of products, movies, anime, books, music, social posts by individuals, self-descriptive

text about a website or product, and so on. These are categorized into domains; each domain has its own quirks, knowledge, and method of being brought about. Although a movie review may sound similar to a book review, they typically differ hugely in the distribution of words used.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

One approach to making sense of these domains is to produce rules from expert knowledge. An expert in movies would tell you that if the review talks about it being a "cannibal horror film", we can understand that it is likely a scary movie and is related to the original 'Cannibal Holocaust' movie. Encoding this kind of knowledge is difficult, time-consuming, and hard to automate reliably.

Most successful approaches in recent times, like vector-spaces, word-vectors, and others, rely on the distributional model of semantics. This model relies on encoding unstructured text e.g. of a movie review, as a vector, where each dimension corresponds to how frequent each word is, we are able to calculate how similar the entities are, e.g. we know that if two movies have a similar distribution of words in their reviews, like frequent use of the word 'scary', or 'horror', then they would have a higher similarity value. These models, also known as 'semantic spaces' encode this similarity information spatially.

A domain-specific semantic space contains a lot of domain knowledge that is not made explicit. One approach towards understanding how these spaces represent information is a conceptual space [**?**]. In this space, we can understand domain entities, e.g. movies in a domain of IMDB movie reviews, to be represented as points, and domain properties to be in regions around these points. The relationships between these regions have been formalized in terms of similarity reasoning and commonsense reasoning.

In this chapter, we focus on further experimenting with one relationship that was formalized in [**?**]: a ranking of entities on properties. In particular, we use this method of building a representation of entities as a way to convert a vector space into an interpretable representation, for use in an interpretable classifier. The reason that we chose this representation to expand on is because by representing each entity e with a vector v that corresponds to a ranking r, the meaning of each dimension is distinct, and we are able to find labels composed of clusters of words for these dimensions. Here, we make the distinction between a property and a word, a property is a natural property of the space that exists in terms of a ranking of entities, and words are the labels we use to describe this property.

Having distinct, labelled dimensions are beneficial when producing an interpretable classifier, e.g. a Decision Tree classifier that uses this representation: As each ranking being distinct, each node of the tree acts as a step in the reasoning of why an entity was classified a certain way, and that node is appropriately labelled so that we are able to understand it. We show an example tree in **??** Interestingly, we find that if a property is relevant enough to a classification task, classifying using a single ranking performs better than using multiple rankings - as these rankings generalize to new entities.

The main focus of our quantitative experimentation is to demonstrate that the rankings our representation is composed of correspond to domain knowledge.

To do so, we use Decision Trees as described in section 2.3.1, which we constrain by depth, forcing the classifier to use only a limited amount of dimensions. We can understand our approach to be suitable if the classifier performs well with use of only a few properties, as we know that these properties are able to represent important concepts in the domain. We compare these to the original representations, which are typically less disentangled, as well as to topic models, another approach towards representing the domain.

We also evaluate the method qualitatively and in terms of interpretability, specifically evaluating the ability of classifiers to appropriately describe how they concluded that a particular entity belongs to a class. The qualitative investigation focuses on demonstrating how different parameters of the method affect the representation through use of examples and analysis.

This chapter continues as follows: We begin by describing the work related to this method, giving valuable context for the utility and potential of our approach. This is followed by an explanation of the method, including the variations we have adopted for our experimental work. We follow this with our qualitative experimentation, explaining how these variations affect the results, as well as the interpretability of the method, and we end with a quantitative analysis on how well we can represent domain knowledge using decision trees constrained to a limited depth.

3.2 Related Work

There are many approaches towards improving document representations.

Probabilistic approaches like topic models.

Matrix factorization approaches.

3.3 Method

The goal of this method is to obtain a representation composed of salient properties, starting with a domain-specific vector space S_e and its associated bag-of-words (BOW) representation B_w. To obtain these properties, we use a variant of the unsupervised method proposed in [**?**], which we explain in this section.

3.3.1 Rankings entities on words

We can understand that only some words will be properties, as only some correspond to domain knowledge, e.g. in a domain of IMDB movies, the word "the" does not correspond to a property of the domain, but the word "horror" does. Initially, we obtain rankings of entities for each word in the space.

As an initial filtering step, we remove words that do not meet a frequency threshold, with the understanding that words that do not occur in a minimum amount of documents are unlikely to correspond to properties as they are too specific to a subset of movies, which would make them difficult to learn. This leaves us with w_n words. We show the kind of words that would be poorly represented in **??**.

Then, for each considered word w, a logistic regression classifier is trained to find a hyperplane H_w in the space that separates entities e which contain w in their BOW B_e representation from those that do not. The vector v_w perpendicular to this hyperplane is then taken as a direction that models the word w. In **??**, we show an example of this in a toy domain. To rank the objects on the entity, if e is the representation of an entity in the given vector space S_e then we can think of the dot product $v_w \cdot e$ as the value $r_e w$ of object e for vector v_w, and in particular, we take $r_e 1 < r_e 2$ to mean that e_2 has the property labelled with the word w to a greater extent than e_1. The result of this is shown in **??**. Example entities, with their associated highest and lowest ranking properties, are shown in **??**.

3.3.2 Filtering directions to obtain salient properties

With the rankings R_r, we could create a representation of each entity Se, composed of w_n dimensions, where each dimension is a ranking of the entity e on that word $w_r e$. However, many of the words do not correspond to salient properties. In-order to filter these words out, we evaluate them using a scoring metric, and remove the words that are not sufficiently well scored. We use three different metrics:

Classification accuracy. Evaluating the quality in terms of the accuracy of the logistic regression classifier: if this classifier is sufficiently accurate, it must mean that whether word w relates to object o (i.e. whether it is used in the description of o) is important enough to affect the semantic space representation of o. In such a case, it seems reasonable to assume that w describes a salient property for the given domain.

Cohen's Kappa. One problem with accuracy as a scoring function is that these classification problems are often very imbalanced. In particular, for very rare words, a high accuracy might not necessarily imply that the corresponding direction is accurate. For this reason, X proposed to use Cohen's Kappa score instead. In our experiments, however, we found that accuracy sometimes yields better results, so we keep this as an alternative metric.

Normalized Discounted Cumulative Gain This is a standard metric in information retrieval which evaluates the quality of a ranking w.r.t. some given relevance scores [**?**]. In our case, the rankings r_e of the entity e are those induced by the dot products $v_w \cdot e$ and the relevance scores are determined by the Pointwise Positive Mutual Information (PPMI) score $ppmi(w, e)$, of the word w in the BoW representation of entity e where $ppmi(w, e) = \max\left(0, \log\left(\frac{p_{we}}{p_{w*} \cdot p_{*o}}\right)\right)$, and

$$p_{wo} = \frac{n(w, o)}{\sum_{w'} \sum_{o'} n(w', o')}$$

where $n(w, e)$ is the number of occurrences of w in the BoW representation of object e, $p_{w*} = \sum_{e'} p_{we'}$ and $p_{*e} = \sum_{w'} p_{w'e}$.

In principle, we may expect that accuracy and Kappa are best suited for binary features, as they rely on a hard separation in the space between objects

that have the word in their BoW representation and those that do not, while NDCG should be better suited for gradual features. In practice, however, we could not find such a clear pattern in the differences between the words chosen by these metrics despite often finding different words. In Table **??**, we show examples of properties scored highly for each domain.

3.3.3 Clustering salient properties

Although we are able to find the words that are most salient, the properties in the domain may not correspond directly to words. Further, the properties may not be well described by their associated word. In-order to find better representations of properties, we cluster together similar vectors v_w, following the assumption that those vectors which are similar are representing some property more general than their individual words, and we can find it between them. As the final step, we cluster the best-scoring candidate feature directions v_w. Each of these clusters will then define one of the feature directions to be used in applications. The purpose of this clustering step is three-fold: it will ensure that the feature directions are sufficiently different (e.g. in a space of movies there is little point in having *funny* and *hilarious* as separate features), it will make the features easier to interpret (as a cluster of terms is more descriptive than an individual term), and it will alleviate sparsity issues when we want to relate features with the BoW representation, which will play an important role for the fine-tuning method described in the next section.

As input to the clustering algorithm, we consider the N best-scoring candidate feature directions v_w, where N is a hyperparameter. To cluster these N vectors, we have followed the approach proposed in [**?**], which we found to perform slightly better than K-means. The main idea underlying their approach is to select the cluster centers such that (i) they are among the top-scoring candidate feature directions, and (ii) are as close to being orthogonal to each other as possible. We refer to [**?**] for more details. The output of this step is a set of clusters $C_1, ..., C_K$, where we will identify each cluster C_j with a set of words. We will furthermore write v_{C_j} to denote the centroid of the directions corresponding to the words in the cluster C_j, which can be computed as $v_{C_j} = \frac{1}{|C_j|} \sum_{w_l \in C_j} v_l$ provided that the vectors v_w are all normalized. These centroids $v_{C_1}, ..., v_{C_k}$ are the feature directions that are identified by our method.

Table **??** displays some examples of clusters that have been obtained for three of the datasets that will be used in the experiments, modelling respectively movies, place-types and newsgroup postings. For each dataset, we used the scoring function that led to the best performance on development data(see Section **??**). Only the first four words whose direction is closest to the centroid v_C are shown. **K-Means Derrac's K-Means Variation Mean-shift Hdbscan**

3.4 Quantitative Results

We demonstrate the effectiveness of our approach on five datasets, each with their associated tasks.

- The IMDB Movie Dataset: 15,000 movies represented by aggregated reviews. On the tasks of Movie Genres, 100 IMDB Keywords, and UK + US Age Certificates.
- Flickr Place-types: 1,383 place-types. On the tasks of three different place-types, Foursquare, Geonames and OpenCYC.
- The 20-Newsgroups dataset: 18,846 newsgroup posting in 20 different categories. On the task of identifying which of the 20 categories the posting is from.
- The IMDB Sentiment Dataset: 50,000 movie reviews, with binary tags for either positive or negative. On the task of identifying if the review is positive or negative.
- The Reuters Dataset: 10655 News articles. On the task of identifying the category of the article.

To test the ability of the identified directions to accurately represent domain concepts in a ranking, we use low-depth decision-trees. Although these classifiers are not intended to be competitive with more complex classifiers like unbounded decision trees or SVM's, we find that our rankings are sometimes able to outperform these approaches using only a single decision node (equivalent to finding the best cutoff in a single ranking for classification). We use the F1 metric for our experiments, as almost all classes in each dataset are unbalanced.

We obtain the unsupervised representations as follows:

- To obtain the PPMI representation, we first remove terms from the corpus that do not appear at least (length of the corpus * 0.01) documents, and then trim all entities that were exclusively composed of these terms. We then we apply standard PPMI based on the implementation from svdmi [1].
- We obtain the MDS spaces for the movies, place-types and wines datasets from the data made available by [**?**], to obtain the MDS spaces for the other datasets, we use the same method as [**?**] and using default parameters for the MDSJ library. For each domain, we obtain 20, 50, 100 and 200 dimensional spaces.
- So that we can use the sparse PPMI matrices when obtaining the space, we the TruncatedSVD method from scikit-learn method[2] with default parameters. For each domain, we obtain 20, 50, 100 and 200 dimensional spaces.

[1] https://github.com/Bollegala/svdmi/blob/master/src/svdmi.py

[2] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.htm

- For the averaged word-vectors (AWV) and the weighted averaged word vectors (AWVw), we average the glove 6B word-vectors[3] obtained from the Wikipedia 2014 + Gigaword 5 corpuses. As these are only available in size 50, 100 and 200, and there are not many other commonly used pre-trained word-vectors that offer multiple dimension sizes, differing from other methods we only obtain AWV and AWVw representations of size 50, 100 and 200. As these dimension sizes are hyper-parameters, we can consider average word vectors to be disadvantaged on some tasks, but as it is unlikely that there is too much benefit in training our own word-vectors from the relatively small domains, we opted to simplify the process and simply remove this as a hyper-parameter for this method, as well as the averaged method. The averaged word vectors are obtained by multiplying the vectors by the PPMI values, and finding the weighted average of all vectors multiplied in this way.

- For the Doc2Vec vectors, we use hyperparameter optimization to select the appropriate parameters, as the quality of the end space is typically reliant on well-tuned hyperparameters for the dataset. We use [**?**] as a guideline for which parameters to optimize, re-using the parameters that stayed constant for both their datasets in their tests, specificially the dbow method, glove6B pre-trained 300-dimensional word-vectors, training those word vectors while training the representation, a sub-sampling of 10(-5), and a negative sample of 5. We tune and select between the values of the window size 5, 15, 50, the minimum frequency count 1, 5, 20, and epoch size 20, 40, 100, and as in the other methods, we obtain vectors of size 20, 50, 100, 200, but the hyperparameters for each of these is found individually, as the different space sizes are later evaluated on how well they can produce good directions. We evaluate the quality of the space using a Gaussian SVM on a selected task for each dataset, in the case of Reuters, Newsgroups and Sentiment, we use their associated tasks, for Movies we use the Genres task and Place-types the Foursquare task, as these tasks represent essential concepts in the domain.

Table 1 shows how well unsupervised representations perform. Topic models are included to demonstrate the difference between other simple and interpretable approaches, and Random Forest's are included to demonstrate the difference between our simple but interpretable approach and a model that typically performs well at the task [**?**], but is difficult to interpret.

Table 2 demonstrates the difference between unsupervised representations and salient properties, and Table 3 demonstrates the difference between salient properties and clustered salient properties.

[3]https://nlp.stanford.edu/projects/glove/