arXiv:2008.09470v1 [cs.CL] 19 Aug 2020

TOP2VEC: DISTRIBUTED REPRESENTATIONS OF TOPICS

Dimo Angelov
dimo.angelov@gmail.com

ABSTRACT

Topic modeling is used for discovering latent semantic structure, usually referred to as topics, in a large collection of documents. The most widely used methods are Latent Dirichlet Allocation and Probabilistic Latent Semantic Analysis. Despite their popularity they have several weaknesses. In order to achieve optimal results they often require the number of topics to be known, custom stop-word lists, stemming, and lemmatization. Additionally these methods rely on bag-of-words representation of documents which ignore the ordering and semantics of words. Distributed representations of documents and words have gained popularity due to their ability to capture semantics of words and documents. We present `top2vec`, which leverages joint document and word semantic embedding to find *topic vectors*. This model does not require stop-word lists, stemming or lemmatization, and it automatically finds the number of topics. The resulting topic vectors are jointly embedded with the document and word vectors with distance between them representing semantic similarity. Our experiments demonstrate that `top2vec` finds topics which are significantly more informative and representative of the corpus trained on than probabilistic generative models.

1 Introduction

The ability to organize, search and summarize a large volume of text is a ubiquitous problem in natural language processing (NLP). Topic modeling is often used when a large collection of text cannot be reasonably read and sorted through by a person. Given a corpus comprised of many texts, referred to as documents, a topic model will discover the latent semantic structure, or topics, present in the documents. Topics can then be used to find high level summaries of a large collection of documents, search for documents of interest, and group similar documents together.

A topic is the theme, matter or subject of a text; it is thing being discussed. Topics are often thought of as discrete values, such as *politics*, *science*, and *religion*. However, this is not to the case since any of these topics can be further subdivided into many other sub-topics. Additionally, a topic like *politics* can overlap with other topics, such as the topic of *health*, as they can both share the sub-topic of *health care*. Any of these topics, their combinations or variations can be described by some unique set of weighted words. As such, we assume that topics are *continuous*, as there are infinitely many combinations of weighted words which can be used to represent a topic. Additionally, we assume that each document has its own topic with a value in that continuum. In this view, the document's topic is the set of weighted words that are most informative of its unique topic, which can be a combination of the colloquial discrete topics.

A useful topic model should find topics which represent a high-level summary of the information present in the documents. Each topic's set of words should represent information contained in the documents. For example, one can infer from a topic containing the words *warming*, *global*, *temperature*, and *environment*, that the topic is *global warming*. We define topic modeling to be the process of finding topics, as weighted sets of words, that best represent the information of the documents.

In the remainder of this section we discuss related work and introduce distributed representations of topics. In Section 2 we describe the `top2vec` model. Section 3 describes topic information gain and summarizes our experiments, and we conclude in Section 4.

1.1 Traditional Topic Modeling Methods

The most widely used topic modeling method is Latent Dirichlet Allocation (LDA) [1]. It is a generative probabilistic model which describes each document as a mixture of topics and each topic as a distribution of words. LDA generalizes

Probabilistic Latent Semantic Analysis (PLSA) [2] by adding a Dirichlet prior distribution over document-topic and topic-word distributions.

LDA and PLSA *discretize* the continuous topic space into t topics and model documents as mixtures of those t topics. These models assume the number of topics t to be known. The discretization of topics is *necessary* to model the relationship between documents and words. This is one of the greatest weakness of these models, as the number of topics t or the way to estimate it is rarely known, especially for very large or unfamiliar datasets [3, 4].

Each topic produced by these methods is a distribution of word probabilities. As such, the highest probability words in a topic are usually words such as *the*, *and*, *it* and other common words in the language [4]. These common words, also called stop-words, often need to be filtered out in order to make topics interpretable, and extract the informative topic words. Finding the set of stop-words that must be removed is not a trivial problem since it is both language and corpus specific [5]; a topic model trained on text about *dogs* will likely treat *dog* as a stop-word since it is not very informative.

LDA and PLSA use bag-of-words (BOW) representations of documents as input which ignore word semantics. In BOW representation the words *Canada* and *Canadian* would be treated as different words, despite their semantic similarity. Stemming and lemmatization techniques aim to address this problem but often make topics harder to understand. Moreover, stemming and lemmatization do not recognize the similarity of words like *big* and *large*, which do not share a word stem.

The authors of the LDA paper explicitly state: "We refer to the latent multinomial variables in the LDA model as topics, so as to exploit text-oriented intuitions, but we make no epistemological claims regarding these latent variables beyond their utility in representing probability distributions on sets of words." [1]. The objective of probabilistic generative models like LDA and PLSA is to find topics which can be used to recreate the original document word distributions with minimal error. However, a large proportion of all text contains uninformative words which may not be considered topical. These models do not differentiate between informative and uninformative words as their goal is to simply recreate the document word distributions. Therefore, the high probability words in topics they find do not necessarily correspond to what a user would intuitively think of as being topical.

1.2 Distributed Representations of Words and Documents

In neural networks, a *distributed representation* means each concept learned by the network is represented by many neurons. Each neuron therefore participates in the representation of many concepts. When a neural network's weights are changed to incorporate new knowledge about a concept, the changes affect the knowledge associated with other concepts that are represented by similar patterns [6]. Distributed representation has the advantage of leading to automatic generalization of the concepts learned. Distributed representations are often central to NLP machine learning techniques for learning vector representations of words and documents.

Another key idea behind learning vector representations of words and documents is the *distributional hypothesis*. The essence of the idea is captured by John Rupert Firth who famously said "You shall know a word by the company it keeps" [7]. This statement implies that words with similar meanings are used in similar contexts.

The continuous skip-gram and BOW models [8, 9] known as `word2vec`, introduced *distributed word representations* that capture syntactic and semantic word relationships. The `word2vec` neural network learns word similarity by predicting which adjacent words should be present to a given context word in a sliding window over each document. The learning task of `word2vec` embraces the idea of distributional semantics, as it learns similar word vectors for words used in similar contexts. It also learns distributed representation of words, in the form of vectors, which facilitates generalization of word representation. The `word2vec` model generated word vectors produced state-of-the-art results on many linguistics tasks compared to traditional methods [8, 9, 10, 11].

There has been interest in methods of finding distributed word vectors that do not rely on neural networks. It has been shown that the skip-gram version of `word2vec` is implicitly factorizing a word-context pointwise mutual information (PMI) matrix [12], based on this finding the authors proposed *Shifted Positive* PMI word-context representation of words. This has inspired other methods such as `GloVe` [13], which learn context and word vectors by factorizing a global word-word co-occurrence matrix. Although `word2vec` implicitly factorizes a word-context PMI matrix, what it *explicitly* does is maximize the dot product between word vectors for words which co-occur while minimizing dot product between words which do not co-occur. Additionally it uses a neural network which takes advantage of its learned distributed representation of words. This allows the model to learn about all words simultaneously from a single training step on a context word [14]. The ability of `word2vec` word vectors to capture syntactic and semantic regularities of language that other methods try to recreate is a result of the former points, as is its ability to scale to large corpora [8, 15]. As shown in [10], quantitative comparisons between neural and non-neural word vectors show that neural learned vectors consistently perform better. Results from [12, 16] show that at best non-neural methods achieve

results on certain tasks that are on-par with neural methods by replicating hyper-parameters of neural methods like word2vec. These methods, however, lack the ability to scale to large corpora.

With the goal of overcoming the weaknesses of BOW representations of documents, the *distributed paragraph vector* was proposed with doc2vec [17]. This model extends word2vec by adding a paragraph vector to the learning task of the neural network. In addition to the context window of words, a paragraph vector is also used to predict which adjacent words should be present. The paragraph vector acts as a memory of the topic of the document; it informs each context window of what information is missing [17]. The doc2vec model can learn distributed representations of varying lengths of text, from sentences to documents. The doc2vec model outperforms BOW models and produces state-of-the-art results on many linguistics tasks compared to traditional methods [17, 18]. The doc2vec model was followed by many works on general language models [19, 20, 21].

1.3 Distributed Representations of Topics

A *semantic space* is a spatial representation in which distance represents semantic association [22]. A lot of attention has been given to semantic embedding of words. Specifically, distributed word vectors generated by models like word2vec which have been shown to capture syntactic and semantic regularities of language [8, 23].

The doc2vec model is capable of learning document and word vectors that are jointly embedded in the same space. It has been observed that doing so, or using pre-trained word vectors, improves the quality of the learned document vectors [18]. These jointly embedded document and word vectors are learned such that document vectors are close to word vectors which are semantically similar. This property can be used for information retrieval as word vectors can be used to query for similar documents. It can also be used to find which words are most similar to a document, or most representative of a document. As mentioned in [17], the paragraph or document vector acts as a memory of the topic of the document. Thus the most similar word vectors to a document vector are likely the most representative of the document's topic. This joint document and word embedding is a *semantic embedding*, since distance in the embedded space measures semantic similarity between the documents and words.

In contrast to traditional BOW topic modeling methods, the semantic embedding has the advantage of learning the semantic association between words and documents. We argue that the semantic space itself is a *continuous representation of topics*, in which each point is a different topic best summarized by its nearest words. In the jointly embedded document and word semantic space, a dense area of documents can be interpreted as many documents that have a similar topic. We use this assumption to propose top2vec, a distributed topic vector which is calculated from dense areas of document vectors. The number of dense areas of documents found in the semantic space is assumed to be the number of prominent topics. The topic vectors are calculated as the centroids of each dense area of document vectors. A dense area is an area of very similar documents, and the centroid, or topic vector, can be thought of as the average document most representative of that area. We leverage the semantic embedding to find the words which are most representative of each topic vector by finding the closest word vectors to each topic vector.

The top2vec model produces jointly embedded topic, document, and word vectors such that distance between them represents semantic similarity. Removing stop-words, lemmatization, stemming, and a priori knowledge of the number of topics are not required for top2vec to learn good topic vectors. This gives top2vec a major advantage over traditional methods. The topic vector can be used to find similar documents and words can be used to find similar topics. The same vector algebra demonstrated with word2vec [8, 9] can be used between the word, document and topic vectors. The topic vectors allow for topic sizes to be calculated based on each document vector's nearest topic vector. Additionally topic reduction can be performed on the topic vectors to hierarchically group similar topics and reduce the number of topics discovered.

The greatest difference between top2vec and probabilistic generative models is how each models a topic. LDA and PLSA model topics as distributions of words, which are used to recreate the original document word distributions with minimal error. This often necessitates uninformative words which are not topical to have high probabilities in the topics since they make up a large proportion of all text. In contrast a top2vec topic vector in the semantic embedding represents a prominent topic shared among documents. The nearest words to a topic vector best describe the topic and its surrounding documents. This is due to the joint document and word embedding learning task, which is to predict which words are most indicative of a document, which necessitates documents, and therefore topic vectors, to be closest to their most informative words. Our results show that topics found by top2vec are significantly more informative and representative of the corpus trained on than those found by LDA and PLSA.

2 Model Description

2.1 Create Semantic Embedding

In order to be able to extract topics, jointly embedded document and word vectors with certain properties are required. Specifically, we need an embedding where the distance between document vectors and word vectors represents semantic association. Semantically similar documents should be placed close together in the embedding space, and dissimilar documents should be placed further from each other. Additionally words should be close to documents which they best describe. With jointly embedded document and word vectors with these properties, topic vectors can be calculated. This spatial representation of words and documents is called a semantic space [22]. We argue that a semantic space with the outlined properties is a *continuous representation of topics*. Figure 1 shows an example of a semantic space.

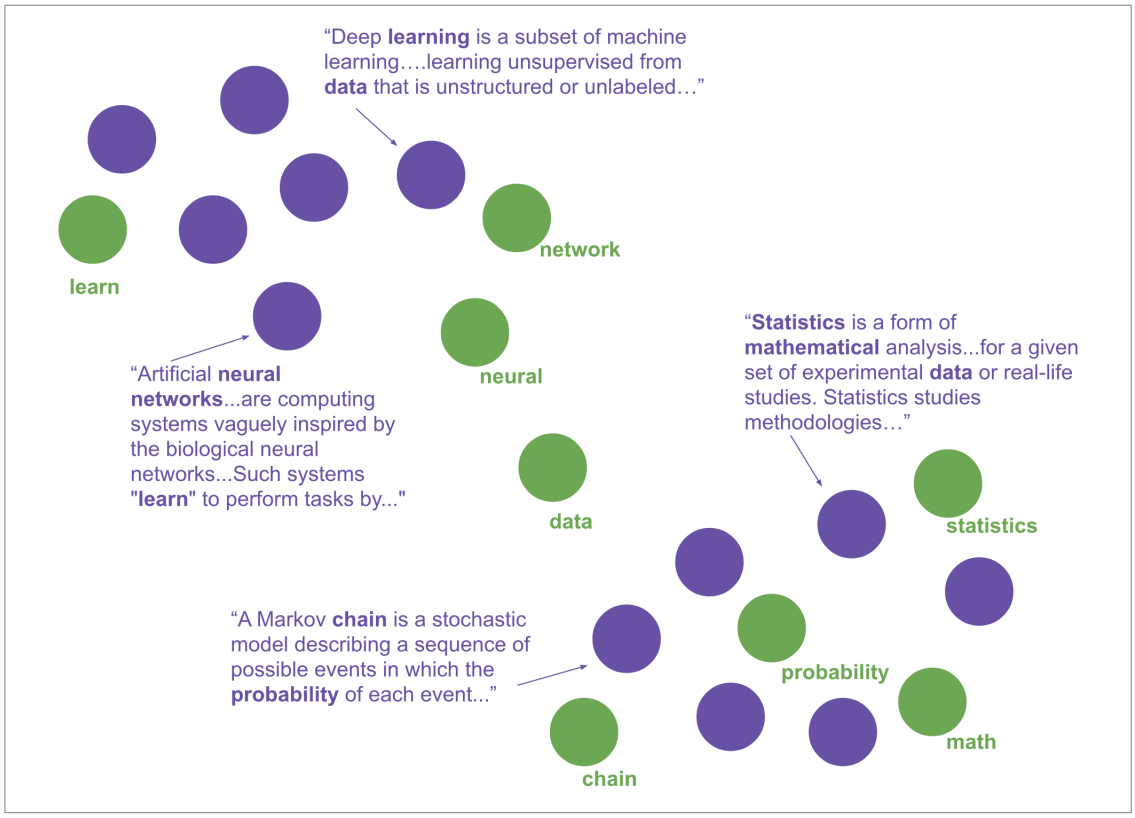

Figure 1: An example of a semantic space. The purple points are documents and the green points are words. Words are closest to documents they best represent and similar documents are close together.

To learn jointly embedded document and word vectors we use `doc2vec` [17, 24]. There are two versions of the model: the Paragraph Vector with Distributed Memory (DM) and Distributed Bag of Words (DBOW). The DM model uses context words and a document vector to predict the target word within context window. The DBOW model uses the document vector to predict words within a context window in the document. Despite DBOW being a simpler model it has been shown to perform better [18]. Our experiments confirm these results and consequently we use the DBOW version of `doc2vec`.

The `doc2vec` DBOW architecture is very similar to the `word2vec` skip-gram model which uses the context word to predict surrounding words in the context window. The only difference is that DBOW swaps the context word for the document vector, which is used to predict the surrounding words in the context window. This similarity allows for the training of the two to be interleaved, thus simultaneously learning document and word vectors which are jointly embedded.

The key insight into how `doc2vec` and `word2vec` learn these vectors is understanding how the prediction task works specifically for DBOW and skip-gram models. The `word2vec` skip-gram model learns an input word and context word vector for each word in the vocabulary. The `word2vec` model consists of a matrix $W_{n,d}$ for input word vectors

and $W'_{n,d}$ for context word vectors, where n is the size of the corpus vocabulary, and d is the size of the vectors to be learned for each word. Each row of $W_{n,d}$ contains a word vector $\vec{w} \in \mathbb{R}^d$ and each row of $W'_{n,d}$ contains a context word vector $\vec{w_c} \in \mathbb{R}^d$. For a given context window of size k, there will be k words to the left and k words to the right of the context word. For each of the $2k$ surrounding words w, their input word vector $\vec{w} \in W_{n,d}$ will be used to predict the context vector $\vec{w_c} \in W'_{n,d}$ of the context word w_c. For each surrounding word w the prediction is $softmax(\vec{w} \cdot W'_{n,d})$. This generates a probability distribution over the vocabulary, for *each* word being the context word w_c. The learning consists of using back propagation and stochastic gradient descent to update *each* context word vector in $W'_{n,d}$, and $\vec{w}$ from $W_{n,d}$, such that the probability of the context vector given the surrounding word, $P(\vec{w_c}|\vec{w})$, is greatest in the probability distribution over the vocabulary. This process is repeated for every context window for all n words. This learning process necessitates semantically similar words to have context word vectors which are close together while making dissimilar words have context word vectors which are distant. This is because in order to maximize the probability $P(\vec{w_c}|\vec{w})$, the value of $\vec{w} \cdot \vec{w_c}$ must be the maximum value in $\vec{w} \cdot W'_{n,d}$. This value is maximized when $\vec{w}$ is closest to $\vec{w_c}$ from word all context vectors in $W'_{n,d}$. Therefore the learning process updates $\vec{w}$ and $W'_{n,d}$ so that $\vec{w}$ and $\vec{w_c}$ are closer together. This can be interpreted as each context word pulling all similar context words towards it in the embedding space, while pushing away all dissimilar words. This results in a semantic space, represented by the context vectors W', where all semantically similar words are close together and all dissimilar words are far apart.

The way the DBOW `doc2vec` model learns document vectors is similar to the `word2vec` skip-gram model. The model consists of a matrix $D_{c,d}$, where c is the number of documents in the corpus and d is the size of the vectors to be learned for each document. Each row of $D_{c,d}$ contains a document vector $\vec{d} \in \mathbb{R}^d$. The model also requires a context word matrix $W'_{n,d}$, which can be pre-trained, randomly initialized, or learned in parallel. For simplicity of the explanation, we will assume a scenario where matrix $W'_{n,d}$ has been pre-trained by a `word2vec` model on the same vocabulary of n words. For each document d in the corpus, the context vector $\vec{w_c} \in W'_{n,d}$ of each word in the document is used to predict the document's vector $\vec{d} \in D_{c,d}$. The prediction is $softmax(\vec{w_c} \cdot D_{c,d})$, which generates a probability distribution over the corpus for *each* document being the document the word is from. The learning consists of using back propagation and stochastic gradient descent to update *each* document vector in $D_{c,d}$ and $\vec{w_c}$ from $W'_{n,d}$, such that the probability of the document given the word, $P(\vec{d}|\vec{w})$, is greatest in the probability distribution over the corpus of documents. This learning process necessitates that document vectors be close to word vectors of words that occur in them and making them distant from word vectors of words that do not occur in them. This can be interpreted as each word attracting documents that are similar to them while repelling documents which are dissimilar to them. This results in a semantic space where documents are closest to the words that best describe them and far from words that are dissimilar to them. Similar documents will be close together in this space as they will be pulled into the same region by similar words. Dissimilar documents will be far apart as they will be attracted into different regions of the semantic space by different words.

We argue that the semantic space generated by `word2vec` and `doc2vec` is a *continuous representation of topics.* This claim can be justified by observing what the learned vector space generated by `word2vec` represents. This model learns a matrix $W'_{n,d}$, which contains context word vectors of dimension d for all n words it is trained on. Each word vector in this matrix alone has no meaning; it only gives relative similarity to other word vectors in the matrix. We argue that this d dimensional embedding space is a continuous representation of topics defined by the matrix $W'_{n,d}$. The matrix $W'_{n,d}$ can be seen as a linear transformation that when applied to a d dimensional vector from the embedding space generates an n dimensional vector. This vector itself is some measure of the strengths of each of the n words in the vocabulary corresponding to the point in the d dimensional space. However what this model has actually learned is how to transform points in the d dimensional space into probability distributions over the n words. Therefore any point $\vec{p}$, in the d dimensional space can be transformed into a probability distribution over the n word vocabulary using $softmax(\vec{p} \cdot W'_{n,d})$. Thus, any point in the d dimensional space represents a different topic. Each word vector, $\vec{w_c} \in W'_{n,d}$, corresponds to the topic in the d dimension space which has the greatest probability of word w_c. In general any point $\vec{p}$ in the d dimensional space can be best described semantically by the nearest word vectors, since those are the words that would have the highest probability in its corresponding topic distribution over the n words in the vocabulary.

There are several hyper-parameters that have a large impact on the performance of `doc2vec` [18]. The *window size* is the number of words left and right of the context word. A *window size* size of 15 has been found to produce the best results [18], which our experiments support. The `doc2vec` model can use negative sampling or hierarchical softmax as its output layer. These are both meant to be efficient approximations of the full softmax [9]. We found that in our experiments the *hierarchical softmax* produces better document vectors. According to [18], the most important hyper-parameter is the *sub-sampling threshold*, which determines the probability of high frequency words being discarded from a given context window. The suggested *sub-sampling threshold* value is 10^5. The smaller this

number is, the more likely it is for a high frequency word to be discarded from the context window [9, 18]. A related hyper-parameter is *minimum count*, which discards all words that have a total frequency that is less than that value from the model all together. This gets rid of extremely rare words which would not contribute to learning the document vectors. In our experiments we found a *minimum count* of 50 to work best, however this value largely depends on corpus size and its vocabulary. The *vector size* is the size of the document and word vectors that will be learned, the larger they are the more complex information they can encode. In general, the suggested *vector size* is 300 [18], with larger data sets larger values will lead to better results, at greater computational cost. The number of *training epochs* suggested by [18] is 20 to 400, with the higher values for smaller data sets. We found 40 to 400 *training epochs* to be a good range.

2.2 Find Number of Topics

The semantic embedding has the advantage of learning a continuous representation of topics. In the jointly embedded document and word vector space, with the properties outlined in 2.1, documents and words are represented as positions in the semantic space. In this space each document vector can be seen as representing the topic of the document [17]. The word vectors that are nearest to a document vector, are the most semantically descriptive of the document's topic.

In the semantic space, a dense area of documents can be interpreted as an area of highly similar documents. This dense area of documents is indicative of an underlying topic that is common to the documents. Since the document vectors represent the topics of the documents, the centroid or average of those vectors can be calculated. This centroid is the *topic vector* which is most representative of the the dense area of documents it was calculated from. The words that are closest to this topic vector are the words that best describe it semantically. The main assumption behind `top2vec` is that the number of dense areas of document vectors equals the number of prominent topics. This is a natural way to discretize topics, since a topic is found for each group of documents sharing a prominent topic.

In order find the dense areas of documents in the semantic space, density based clustering is used on the document vectors, specifically Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [25, 26, 27]. However, the "curse of dimensionality" which results from the high-dimensional document vectors introduces two main problems. In the high-dimensional semantic embedding space, regularly of 300 dimensions, the document vectors are very sparse. The document vector sparsity makes it difficult to find dense clusters and doing so comes at a high computational cost [28]. In order to alleviate these two problems, we perform dimension reduction on the document vectors with the algorithm Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [29, 30]. In the dimension-reduced space, HDBSCAN can then be used to find dense clusters of documents.

2.2.1 Low Dimensional Document Embedding

Dimension reduction allows for dense clusters of documents to be found more efficiently and accurately in the reduced space. UMAP is a manifold learning technique for dimension reduction with strong theoretical foundations [29, 30]. T-distributed Stochastic Neighbor Embedding (t-SNE) [31] is another popular dimensional reduction technique. We found that t-SNE does not preserve global structure as well as UMAP and it does not scale well to large datasets. Hence, UMAP is chosen for dimension reduction in `top2vec`, as it preserves local and global structure, and is able to scale to very large datasets. Figure 2 shows UMAP-reduced document vectors; it can be seen that a lot of global and local structure is preserved in the embedding.

UMAP has several hyper-parameters that determine how it performs dimension reduction. Perhaps the most important parameter is the *number of nearest neighbours*, which controls the balance between preserving global structure versus local structure in the low dimensional embedding. Larger values put more emphasis on global over local structure preservation. Since the goal is to find dense areas of documents which would be close to each other in the high dimensional space, local structure is more important in this application. We find that setting *number of nearest neighbours* to 15 gives the best results, as this value gives more emphasis on local structure. Another related parameter is the *distance metric*, which is used to measure the distance between points in the high dimensional space. The often used *distance metric* for the document vectors is *cosine similarity* [8, 9], because it measures similarity of documents irrespective of their size. Lastly the *embedding dimension* must be chosen; we find 5 dimensions to give the best results for the downstream task of density based clustering.

2.2.2 Find Dense Clusters of Documents

The goal of density based clustering is to find areas of highly similar documents in the semantic space, which indicate an underlying topic. This is performed on the UMAP reduced document vectors. The challenge is that the document vectors will have varying density throughout the semantic space. Additionally there will be sparse areas where documents are highly dissimilar. This can be seen as noise, as there is no prominent underlying topic. In order to overcome these

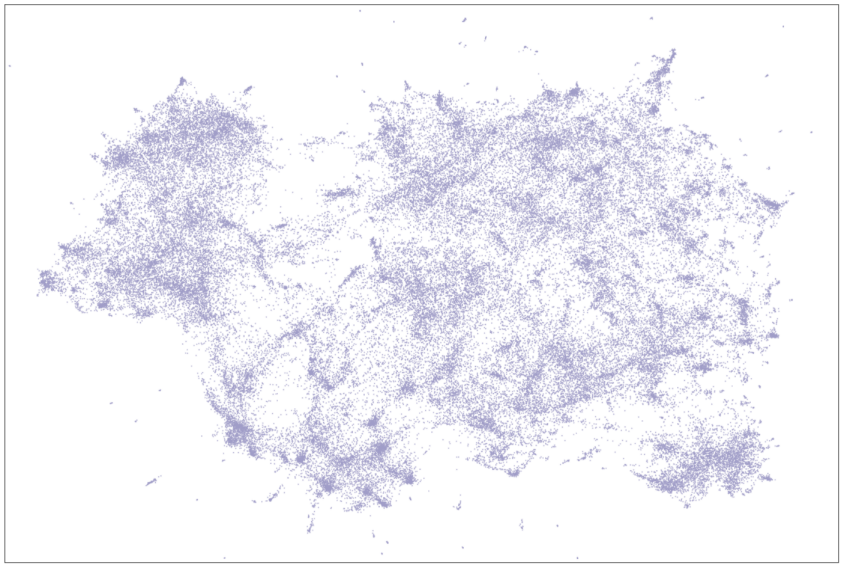

Figure 2: 300 dimensional document vectors from the *20 news groups* dataset that are embedded into 2 dimensions using UMAP.

challenges, HDBSCAN is used to find the dense areas of document vectors, as it was designed to handle both noise and variable density clusters [26]. HDBSCAN assigns a label to each dense cluster of document vectors and assigns a noise label to all document vectors that are not in a dense cluster. The dense areas of identified document vectors will be used to calculate the topic vectors. Documents that are classified as noise can be seen as not being descriptive of a prominent topic. Figure 3 shows an example of dense areas of documents identified by HDBSCAN.

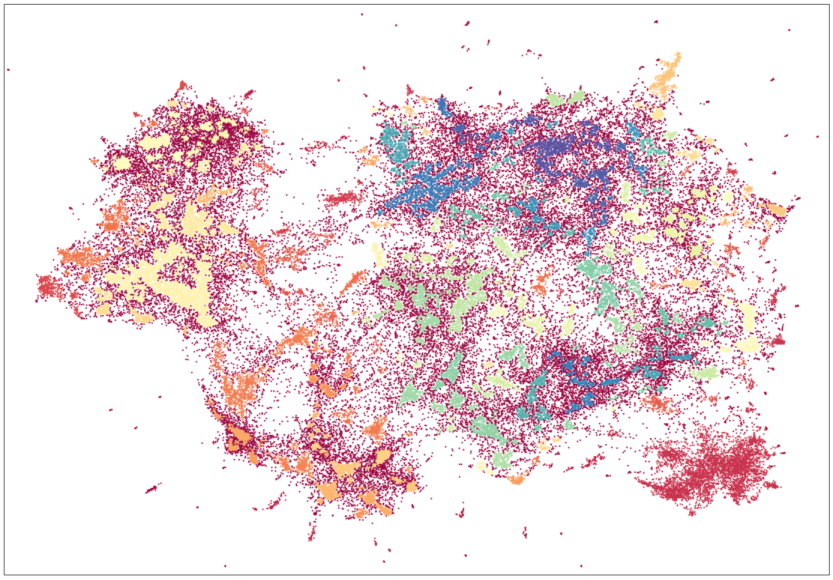

Figure 3: UMAP-reduced document vectors from the *20 news groups* dataset. Each colored area of points is a dense area of documents identified by HDBSCAN, red points are documents HDBSCAN has labeled as noise.

The main hyper-parameter that needs be chosen for HDBSCAN is *minimum cluster size*; this parameter is at the core of how the algorithm finds clusters of varying density [26]. This parameter represents the smallest size that should be considered a cluster by the algorithm. We find that a *minimum cluster size* of 15 gives the best results in our experiments, as larger values have a higher chance of merging unrelated document clusters.

2.3 Calculate Topic Vectors

2.3.1 Calculate Centroids in Original Dimensional Space

The dense clusters of documents and noise documents identified by HDBSCAN in the UMAP-reduced dimension, correspond to locations in the original semantic embedding space. The use of UMAP and HDBSCAN can be seen as a process which labels each document in the semantic embedding space with either a noise label or a label for the dense cluster to which it belongs.

Given labels for each cluster of dense documents in the semantic embedding space, topic vectors can be calculated. There are a number of ways that the topic vector can be calculated from the document vectors. The simplest method is to calculate the centroid, i.e. the arithmetic mean of all the document vectors in the same dense cluster. There are other reasonable options such as the geometric mean or using probabilities from the confidence of clusters created by HDBSCAN. We experimented with these techniques and found that they resulted in very similar topic vectors, with almost identical nearest-neighbour word vectors. We speculate that this is mainly due to the sparsity of the high dimensional space. Therefore, we decided to use the simple method of calculating the centroid. Figure 4 shows a visual example of a topic vector being calculated from a dense area of documents.

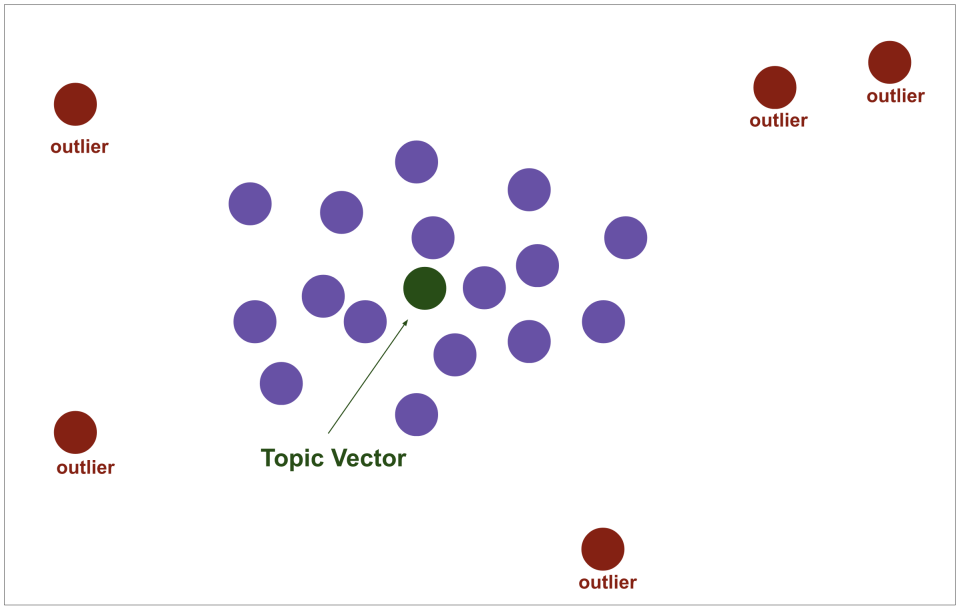

Figure 4: The topic vector is the centroid of the dense are of documents identified by HDBSCAN, which are the purple points. The outliers identified by HDBSCAN are not used to calculate the centroid.

The centroid is calculated for each set of document vectors that belong do a dense cluster, generating a topic vector for each set. The number of dense areas found is the number of prominent topics identified in the corpus.

2.3.2 Find Topic Words

In the semantic space, every point represents a topic that is best described semantically by its nearest word vectors. Therefore the word vectors that are closest to a topic vector are those that are most representative of it semantically. The distance of each word vector to the topic vector will indicate how semantically similar the word is to the topic. The words closest to the topic vector can be seen as the words that are most similar to all documents into the dense area, as the topic vector is the centroid of that area. These words can be used to summarize the common topic of the documents in the dense area. Figure 5 shows an example of a topic vector and the nearest words.

Common words appear in most documents and, as such, they are often in a region of the semantic space that is equally distant from all documents. As a result the words closest to a topic vector will rarely be stop-words, which has been confirmed in our experiments. Therefore there is no need for stop-word removal.

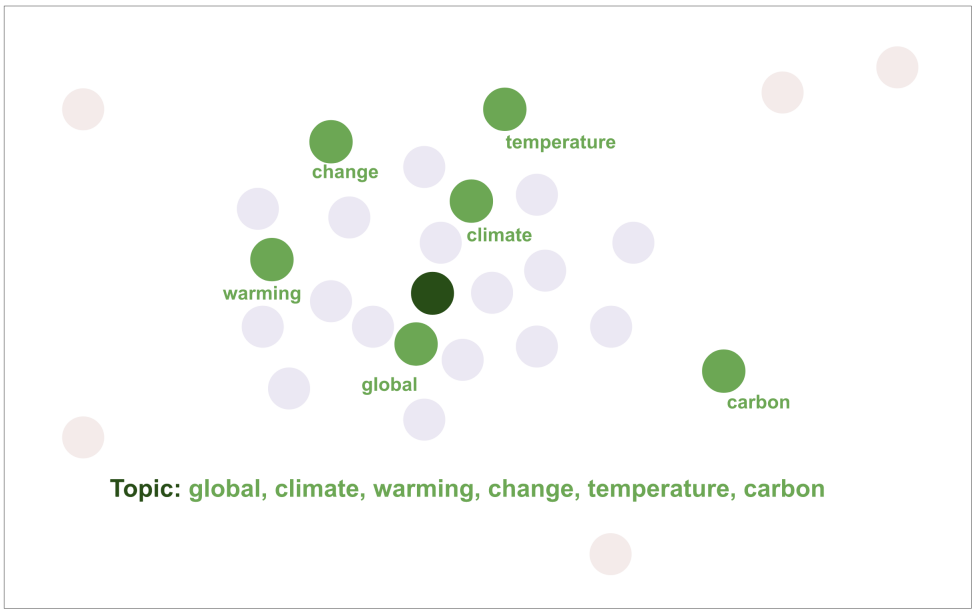

Figure 5: The topic words are the nearest word vectors to the topic vector.

2.4 Topic Size and Hierarchical Topic Reduction

The topic and document vectors allow for the size of topics to be calculated. The topic vectors can be used to partition the document vectors such that each document vector belongs to its nearest topic vector. This associates each document with exactly one topic, the one which is most semantically similar to the document. The size of each topic is measured as the number of documents that belong to it.

An advantage of the topic vectors and the continuous representation of topics in the semantic space is that the number of topics found by `top2vec` can be hierarchically reduced to any number of topics less than the number initially found. This is done by iteratively merging the smallest topic into its most semantically similar topic until the desired number of topics are reached. This is done by taking a weighted arithmetic mean of the topic vector of the smallest topic and its nearest topic vector, each weighted by their topic size. After each merge, the topic sizes are recalculated for each topic. This hierarchical topic reduction has the advantage of finding the topics which are most representative of the corpus, as it biases topics with greater size.

3 Results

3.1 Topic Information Gain

A natural way to evaluate topic models is to score how well the topics describe the documents. This evaluation measures how informative the topics are to a user. We propose using *mutual information* [32] to measure the information gained about the documents when described by their topic words.

Traditional topic modeling methods discretize topic space and describe documents as a mixture of those topics. In order to evaluate a set of these topics T generated from documents D, the total information gained is calculated for each document when described with the proportions of topics given by the topic model.

In contrast, `top2vec` learns a continuous representation of topics and places documents in that space corresponding to their topic. A topic vector found by `top2vec` represents the topic common to a group of documents, or the average of their individual topics. In order to evaluate a set of `top2vec` topics T generated from documents D, the documents are partitioned into sub-sets, with each sub-set corresponding to document vectors with the same nearest topic vector. Thus each document is assigned to exactly one topic. To evaluate these topics, the total information gained is measured for each of the sub-set of documents when described by the words nearest to their topic vector.

The total information gained, or mutual information, about all documents *D* when described by all words *W*, is given by:

$$I(D,W) = \sum_{d \in D} \sum_{w \in W} P(d,w) log \left(\frac{P(d,w)}{P(d)P(w)} \right) \tag{1}$$

The contribution of each co-occurrence between a document *d* and word *w* to the information gain calculation can be seen as the *probability-weighted amount of information* (PWI) *d* and *w* contribute to the total amount of information [33], given by:

$$PWI(d,w) = P(d,w) log \left(\frac{P(d,w)}{P(d)P(w)} \right) \tag{2}$$

Topics are distributions over the entire vocabulary *W*. However, in order to evaluate their usefulness to a user, we evaluate them using the top *n* words of the topic. For evaluation where each document is assigned to only one topic, each topic $t \in T$, will have a set of *n* words $W_t \subset W$ and documents $D_t \subset D$. The information gained about all documents when described by their corresponding topic is given by:

$$\begin{aligned} PWI(T) &= \sum_{t \in T} \sum_{d \in D_t} \sum_{w \in W_t} P(d,w) log \left(\frac{P(d,w)}{P(d)P(w)} \right) \\ &= \sum_{t \in T} \sum_{d \in D_t} \sum_{w \in W_t} P(d|w) P'(w) log \left(\frac{P(d,w)}{P(d)P(w)} \right) \end{aligned} \tag{3}$$

In equation 3, $P(w)$ is the marginal probability of the word w across all documents D. It is used to calculate the *log* term, which is the *pointwise mutual information* [34] between w and d. $P'(w)$ is the probability of topic word w, which is used to calculate the *expected mutual information* [32], or the information gained about document d given topic word w. The quantity we are measuring is the information gained about each document given its corresponding topic words as a prior. Therefore $P'(w)$ is 1 and can be omitted [33], which gives rise to:

$$PWI(T) = \sum_{t \in T} \sum_{d \in D_t} \sum_{w \in W_t} P(d|w) log \left(\frac{P(d,w)}{P(d)P(w)} \right) \tag{4}$$

Alternatively the equation can be generalized for the case that each document is represented by multiple topics. In this case we replace $P'(w)$ with $P(t)$, which is the proportion of words to be used to represent document d by topic t:

$$PWI(T) = \sum_{d \in D} \sum_{t \in T} \sum_{w \in W_t} P(d|w) P(t) log \left(\frac{P(d,w)}{P(d)P(w)} \right) \tag{5}$$

Using Equations 4 and 5, different sets of topics can be compared. A greater quantity of information gain indicates that the topics $t \in T$ are more informative of their corresponding documents. If topics contain words such as *the*, *and*, and *it* or other intuitively uninformative words, they will receive lower information gain values. This is in large part due to the $P(d|w)$ term in the calculation, since the probability of any specific document given a very common word is very low. Therefore, the information gained is also low. Words that are mostly present in the subset of documents corresponding to the topic lead to higher information gain as they are informative of those documents. Additionally, low values of information gain will be obtained if the topic model assigns topics to the wrong documents. Topic information gain measures the quality of the words in the topic and their assignment to documents. Therefore, Equations 4 and 5 give values that correspond with what is intuitively more informative. We argue that due to its information theoretic derivation, topic information gain is a good measure for evaluating topic models.

3.2 LDA, PLSA and Top2Vec Topic Information Gain

In order to evaluate LDA, PLSA and `top2vec` topics we train all models on the same documents D and vocabulary W. Since `top2vec` automatically finds the number of topics, we compare LDA, PLSA and `top2vec` on increasing numbers of topics up to the amount discovered by `top2vec`.

For each comparison between a set of LDA-generated topics, T_{LDA}, PLSA-generated topics, T_{PLSA} and top2vec-generated topics, $T_{top2vec}$, we use the same number of top n topic words and the same number of topics. Thus, for each comparison between T_{LDA}, T_{PLSA}, and $T_{top2vec}$, we ensure the following:

- $|T_{LDA}| = |T_{PLSA}| = |T_{top2vec}| = number\ of\ topics$
- $|W_t| = n, \forall W_t \in T_{LDA}, T_{PLSA}, T_{top2vec} = top\ n\ topic\ words$

3.2.1 20 News Groups Dataset

The *20 News Groups* dataset [35] contains 18,831 posts labelled with the news group they were posted in. We trained top2vec, LDA and PLSA models on this dataset using the same pre-processing steps. LDA and PLSA models were trained with 10 to 100 topics, with intervals of 10. Hierarchical topic reduction was used on the 103 topics discovered by top2vec.

To calculate $PWI(T_{LDA})$, $PWI(T_{PLSA})$, and $PWI(T_{top2vec})$, we use the same W and D. A comparison of the topic information gain for models trained on the *20 news groups* dataset can is shown in Figure 6. The results show that the top n topic words from top2vec consistently provide more information than PLSA and LDA, with varying topic sizes and with up to the top 1000 topic words. Even when stop-words are filtered from LDA and PLSA. For most topic sizes the top 20 words from top2vec convey as much information as the top 100 from LDA and PLSA.

Tables 1 and 2 show the topics for top2vec and LDA models with topic size of 20. LDA was chosen over PLSA as it had higher topic information gain for 20 topics. Topics are ordered by increasing information gain. The topics shown for LDA have stop-words removed, where as the top2vec topics are the exact words discovered by the model. Tables 1 and 2 demonstrate that the topic information gain score corresponds to what is intuitively more informative.

In Table 2, LDA topics 2, 3 and 5 appear to contain nonsensical tokens, yet they have a high information gain. The *20 news groups* data set contains messages that were sent encrypted or contain source code. When the *20 news groups* messages are tokenized, these tokens are treated as words by the models. Thus, LDA has actually found informative tokens for that set of messages. However, that set contains only 23 messages. Therefore, LDA has found 3 different topics out of the requested 20, which only represent 23 messages out of the 18831 total amount the LDA model is trained on. This highlights an advantage of top2vec when finding the number of topics.

Figure 7 shows the semantic embedding of the messages labeled by the news group each message was posted in. This figure shows that the semantic embedding has learned the similarity of messages by visually demonstrating the continuous representation of topics. Messages from similar newsgroups are in similar regions of the semantic space. The small red points on the very right of Figure 7, are the 23 messages which predominantly contain encrypted content or large quantities of source code. Due to the density of that set of messages, top2vec finds a topic for those messages. However, when hierarchical topic reduction is performed to reduce the topic size to 20, due to its small size, the topic of the encrypted and source code containing messages is merged into another topic that is most semantically similar to it. The semantic embedding of the messages labeled with the 20 top2vec topics from Table 1 that each belong to is shown in Figure 8. It demonstrates the assignment of the posts to the 20 topics correspond almost exactly to the 20 news groups and that each topic's top 3 words are very informative of the news group's actual topic. This visually demonstrates that top2vec finds topics which are more representative of the corpus as a whole, as confirmed by the topic information gain score in Figure 6.

Topic information gain measures how informative topic words are of documents. Therefore, low scores are achieved when uninformative topic words are chosen, as well as when topics are assigned either to wrong documents or with incorrect proportions. There are a number of LDA topics in Table 2 that appear to be very coherent and that correspond to specific news groups. However, they have low scores in comparison to similar top2vec topics in Table 1. This is explained, in part, by LDA's modeling of documents as a mixture of topics. It models each document with non-zero probabilities of all topics. Therefore each of the messages will have some non-zero proportion of the topics 2, 3 and 5 that were generated from encrypted or source code containing messages. Figure 9 shows the contribution of each LDA topic from Table 2 to all messages. It demonstrates that the most informative topics are highly localized and that the uninformative topics are spread out over many messages. Topic 15 and 17, which both have low information gain, make up a large proportion of most messages. These are topics with very generic words that are found in most documents.

The goal of LDA is to find topics such that their words recreate the original document word distributions with minimal error. This includes stop-words such as *the*, *and*, *it* and other generic words that would not be considered informative or topical by a user. This explains topic 15 and 17 which are just the generic words that occur in most documents. LDA's goal can also result in extremely specific topics, such as 2, 3, and 5, which necessitate other topics to be more general. Figure 9 visually demonstrates the reason that LDA topics produce lower information gain; it finds many unlocalized and therefore uninformative topics compared to top2vec.

Figure 6 shows that as the number of topics increases, the topic information gain for top2vec is consistently higher than for LDA and PLSA. This is because top2vec topics are more localized in the semantic space and therefore more informative. The number of topics found by top2vec on the *20 News Groups* data set is 103, and are even more localized than the 20 topics in Table 1 which were generated from hierarchichal topic reduction. The original topics discovered in the region of topic 7 and 14 are shown in Figure 10 and Figure 11. These topics are even more localized than the reduced topics and therefore more informative as indicated by the information gain scores in Figure 6.

3.2.2 Yahoo Answers Dataset

The *Yahoo Answers* dataset [36, 37], contains 1.3 million labelled posts. The posts are from 10 different topics, with 130,000 posts per topic. The number of topics top2vec found in this dataset are 2,618. Due to the computational cost of training LDA and PLSA models, we were only able to train the models from 10 to 100 topics with intervals of 10. Hierarchical topic reduction was used on the topics discovered by top2vec.

To calculate $PWI(T_{LDA})$, $PWI(T_{PLSA})$, and $PWI(T_{top2vec})$, we use the same W and D. A comparison of the topic information gain for models trained on the *Yahoo Answers* dataset can be seen in Figure 12. These results are consistent with the results from the *20 News Groups* dataset. They show that the top n topic words from top2vec consistently provide more information than PLSA and LDA, with varying topic sizes and up to the top 1000 topic words. Even when stop-words are filtered from LDA and PLSA. For most topic sizes, the top 20 words from top2vec convey as much information as the top 100 from LDA and PLSA.

Tables 3 and 4 show the topics for top2vec and LDA models with a topic size of 10. The topics are ordered by increasing information gain. LDA was chosen over PLSA because it had higher topic information gain for 10 topics. The topics shown for LDA have stop-words removed, where as the top2vec topics are the exact words discovered by the model. This comparison demonstrates the interpretability of the topics and their associated information gain score, showing that the more informative topics receive higher information gain.

Figure 13 shows the semantic embedding of *Yahoo Answers* posts with their true topic labels. This figure demonstrates that the semantic space has captured the similarity of posts that share a similar topic. Figure 14 shows the posts labelled with the top2vec topics from Table 3. It demonstrates that the assignment of the posts to the 10 topics correspond almost exactly to the 10 topic labels from the *Yahoo Answers* dataset and that the topic's top 3 words are very informative of the true topic. This visually demonstrates that top2vec finds topics that are representative of the corpus as a whole, as confirmed by the topic information gain score in Figure 12.

Figure 15 shows the strengths of each of the 10 LDA topics from Table 4 across all posts. This visually demonstrates that more informative topics are localized in the semantic space, and that LDA discovers topics that are less localized than top2vec topics. Additionally highly unlocalized LDA topics like 9 and 10, which contain the lowest information gain scores, also contain generic words that would not be considered topical or informative by a user. Figure 15 demonstrates visually that, apart from LDA topics containing less informative words, the reason LDA topics receive lower topic information gain is that they are less localized than top2vec topics and therefore less informative.

4 Discussion

We have described top2vec, an unsupervised learning algorithm that finds topic vectors in a semantic space of jointly embedded document and word vectors. We have shown that the semantic space is a continuous representation of topics that allows for the calculation of topic vectors from dense areas of highly similar documents, topic size, and for hierarchical topic reduction. The top2vec model also allows for comparing similarity between words, documents and topics based on distance in the semantic space.

We have proposed a novel method for evaluating topics that uses mutual information to calculate how informative topics are of documents. The topic information gain measures the amount of information gained about the documents when described by their topic words. This measures both the quality of topic words and the assignment of topics to the documents. Our results show that top2vec consistently finds topics that are more informative and representative of the corpus than LDA and PLSA, for varying sizes of topics and number of top topic words.

There are several advantages of top2vec over traditional topic modeling methods like LDA and PLSA. The primary advantages are that it automatically finds the number of topics and finds topics that are more informative and representative of the corpus. As demonstrated, stop-word lists are not required to find informative topic words, making it easy to use on a corpus of any domain or language. The use of distributed representations of words alleviates several challenges of traditional methods that use BOW representations of words, which ignore word semantics.

Traditional topic modeling techniques like LDA and PLSA are generative models; they seek to find topics that recreate the original documents word distributions with minimal loss. This necessitates these models to place uninformative words in topics with high probability, as they make up a large proportion of all documents. Additionally, there is no guarantee that they will find topics that are representative of the corpus. The results show they can find topics that are extremely specific or overly broad.

In contrast, the words closest to `top2vec` topic vectors are the words that are most *informative* of the documents the topic vectors are calculated from. This is due to the learning task that generates joint document and word vectors, which predicts the document a word came from. This learning task necessitates document vectors to be placed close to the words that are most informative of the documents. The continuous representation of topics in the semantic space allows topic vectors to be calculated from dense areas of those documents. Thus `top2vec` topics are the words that are most informative of a document, rather than the set of words that recreate the documents distribution of words with accurate proportions. We suggest that `top2vec` is more appropriate for finding informative and representative topics of a corpus than probabilistic generative models like LDA and PLSA.

The `top2vec` code is available as an open-source project[1].

[1] https://github.com/ddangelov/Top2Vec

Table 1: Topic information gain for the top 10 words from `top2vec` topics trained on the *20 news groups* dataset with 20 topics.

Topic Number	**Topic Words**	**PWI(T)**
1	pitching, pitchers, pitcher, hitter, batting, hit, hitters, baseball, batters, inning	74.2
2	bike, ride, riding, bikes, motorcycle, bikers, helmet, riders, countersteering, passenger	71.9
3	circuit, voltage, circuits, resistor, signal, khz, impedance, analog, diode, resistors	69.1
4	centris, ram, mhz, quadra, nubus, vram, iisi, lciii, cpu, fpu	62.4
5	patient, symptoms, patients, doctor, disease, treatment, jxp, therapy, skepticism, physician	59.1
6	koresh, fbi, compound, batf, davidians, atf, waco, raid, fire, bd	54.7
7	israel, arab, arabs, israeli, jews, palestinians, israelis, war, peace, occupied	54.3
8	orbit, space, launch, orbital, satellites, lunar, shuttle, spacecraft, moon, earth	53.7
9	clipper, nsa, encryption, encrypted, secure, keys, crypto, algorithm, escrow, scheme	51.6
10	controller, drives, drive, ide, scsi, floppy, bios, disk, jumpers, esdi	50.3
11	windows, drivers, ati, cica, driver, exe, card, autoexec, mode, ini	50.1
12	car, engine, cars, ford, brakes, honda, tires, valve, wheel, rear	49.7
13	hockey, playoffs, nhl, game, season, team, playoff, teams, scoring, play	48.0
14	gun, guns, firearms, laws, weapons, handgun, crime, amendment, handguns, firearm	46.6
15	window, application, xlib, manager, openwindows, motif, server, xview, client, clients	43.6
16	jesus, christ, god, bible, church, scripture, christians, scriptures, christian, heaven	38.9
17	postscript, format, printer, fonts, files, formats, font, truetype, bitmap, image	36.7
18	shipping, sale, offer, condition, asking, brand, sell, obo, price, selling	36.0
19	atheists, belief, religion, beliefs, god, christianity, truth, religions, believe, atheist	34.4
20	please, mail, post, email, posting, address, thanks, reply, interested, appreciate	11.3
		996.6

Table 2: Topic information gain for the top 10 words from LDA topics, *after* stop-word removal, trained on the *20 news groups* dataset with 20 topics.

Topic Number	**Topic Words**	**PWI(T)**
1	la, pit, gm, det, bos, tor, pts, chi, vs, min	49.9
2	hz, cx, ww, uw, qs, c_, pl, lk, ck, ah	47.0
3	ax, max, pl, di, tm, ei, giz, wm, bhj, ey	42.6
4	db, period, goal, play, pp, shots, st, power, mov, bh	27.9
5	mk, mm, mp, mh, mu, mr, mj, mo, mq, mx	27.7
6	health, medical, new, study, research, disease, cancer, use, patients, drug	19.0
7	armenian, people, said, one, armenians, turkish, went, us, children, turkey	17.3
8	dos, windows, drive, card, system, disk, mb, scsi, pc, mac	16.7
9	file, program, window, files, image, jpeg, use, windows, display, color	15.7
10	government, president, law, would, mr, israel, state, rights, fbi, states	13.4
11	god, jesus, bible, church, christian, christ, christians, faith, lord, man	12.2
12	game, team, games, hockey, season, teams, league, nhl, new, players	12.0
13	space, nasa, earth, launch, shuttle, orbit, moon, satellite, solar, mission	11.8
14	edu, ftp, graphics, available, pub, image, mail, com, version, also	9.7
15	would, know, anyone, get, thanks, like, one, please, help, could	8.5
16	key, use, data, system, one, information, may, encryption, used, number	7.5
17	people, would, one, think, know, like, say, even, see, way	7.3
18	one, car, would, like, get, time, much, also, back, power	7.1
19	edu, com, please, list, mail, sale, send, email, price, offer	4.0
20	think, year, would, good, time, last, well, get, one, got	3.6
		——
		360.9

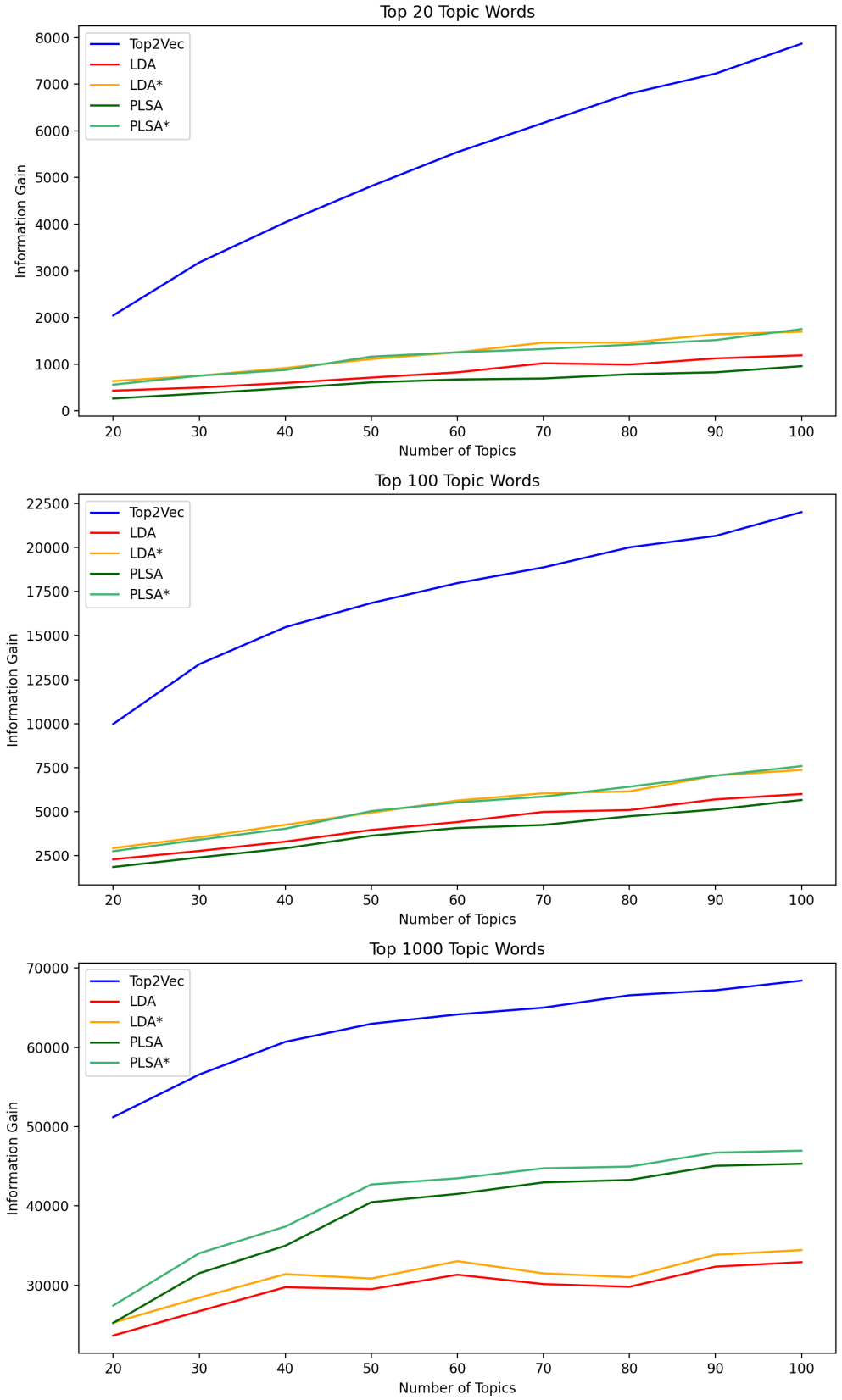

Figure 6: Topic information gain comparison between Top2Vec, PLSA, and LDA trained models on the *20 News Groups* dataset. LDA* and PLSA* have stop-words removed.

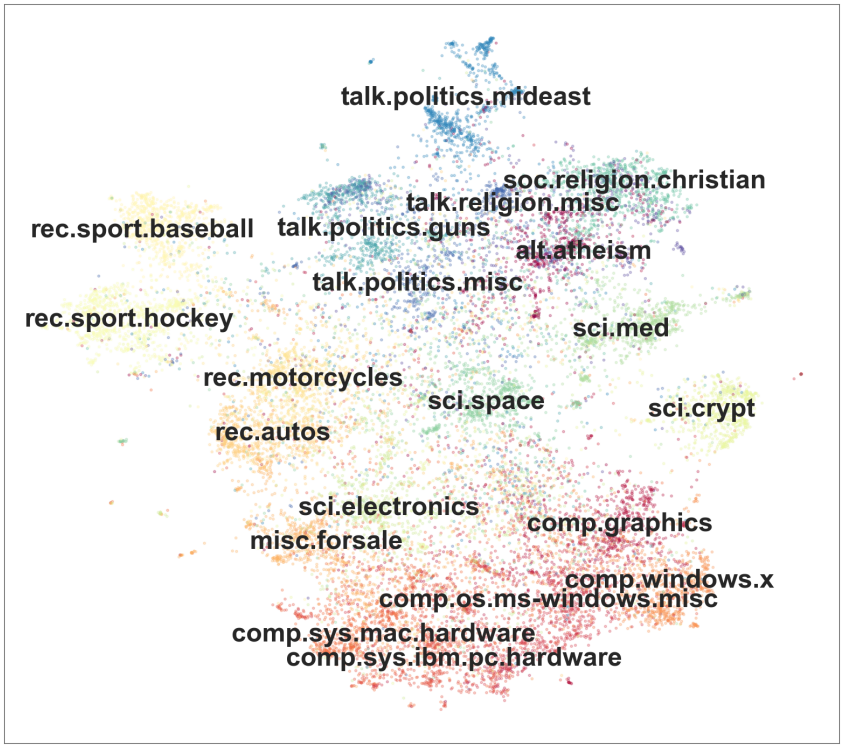

Figure 7: Semantic embedding of *20 news groups* messages labeled by news group. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

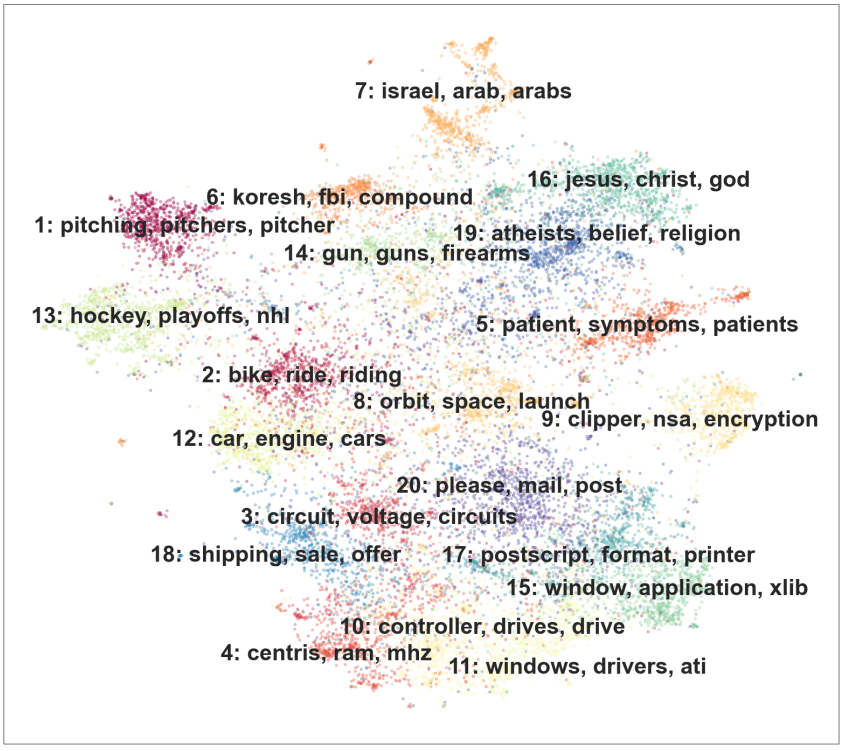

Figure 8: The *20 news groups* messages labeled with the `top2vec` topics from Table 1. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

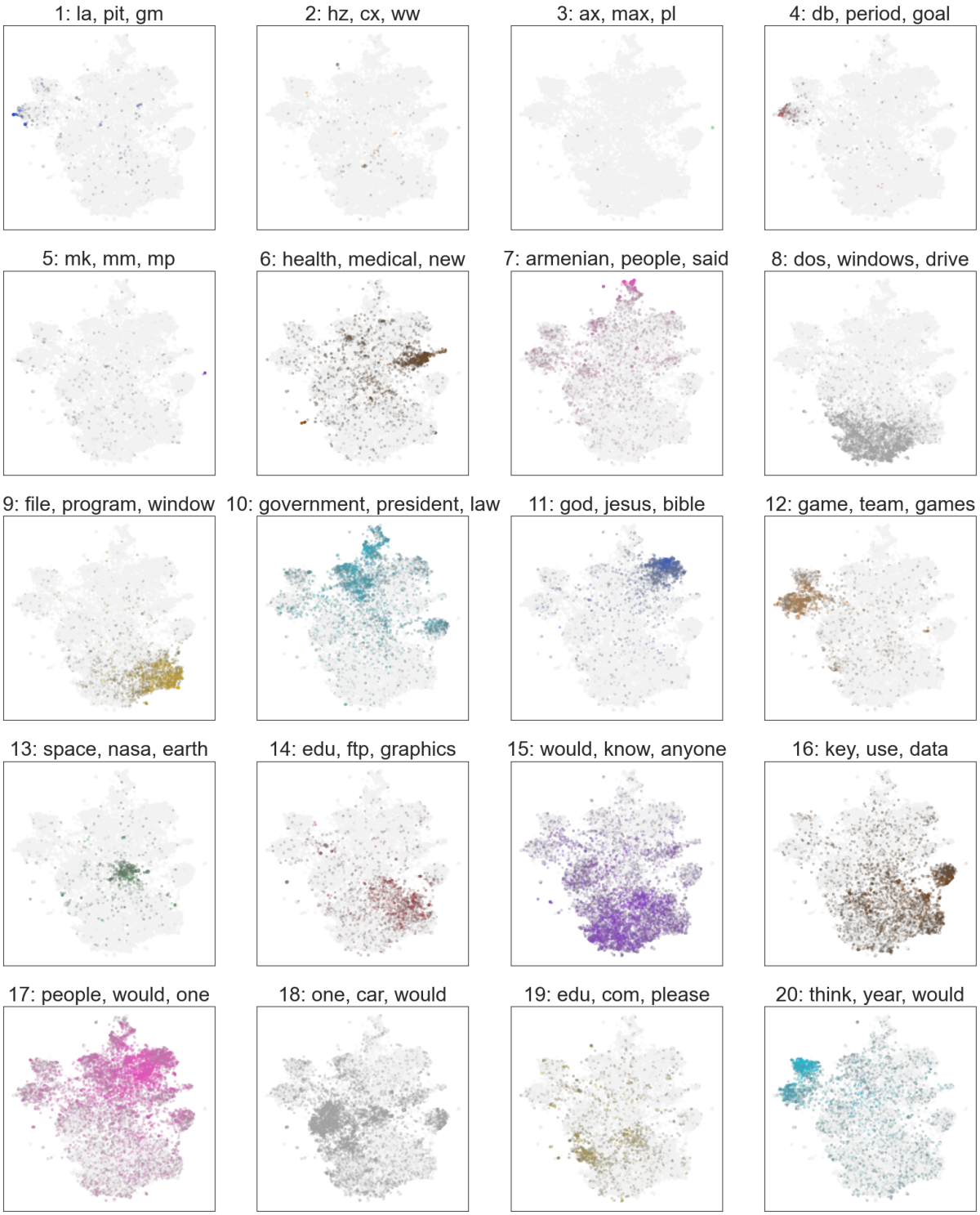

Figure 9: Topic proportion of each LDA topic from Table 2 across all *20 news groups* messages in the semantic embedding. The topics are ordered by decreasing information gain. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

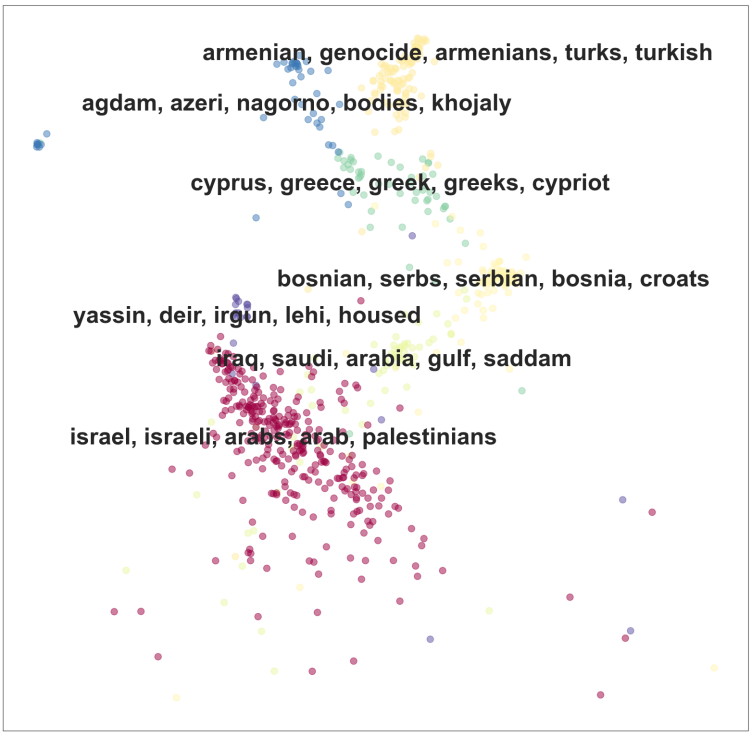

Figure 10: Zoom in of `top2vec` original topics found in region of topic 7 from Table 1. This region of the semantic space corresponds to the *talk.politics.mideast* news group. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

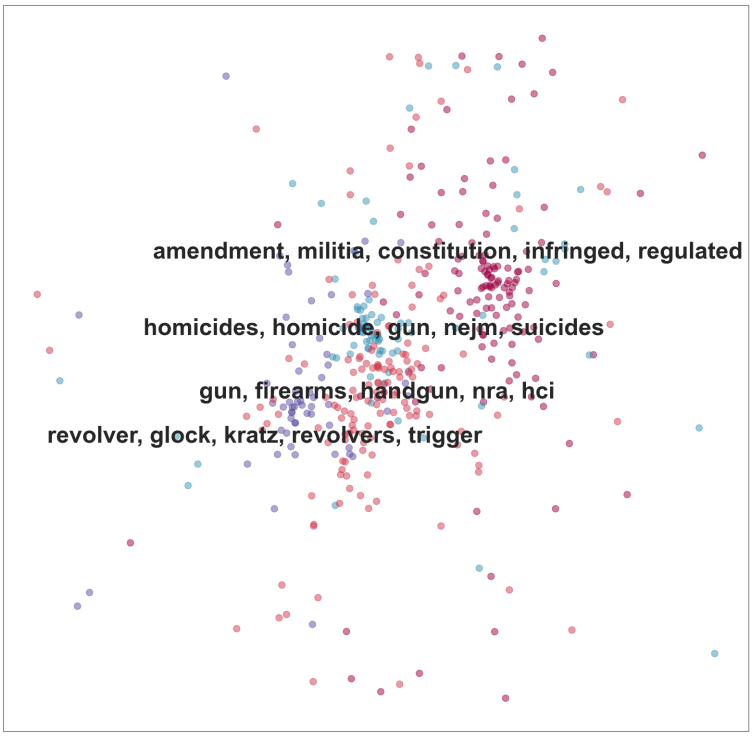

Figure 11: Zoom in of `top2vec` original topics found in region of topic 14 from Table 1. This region of the semantic space corresponds to the *talk.politics.guns* news group. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

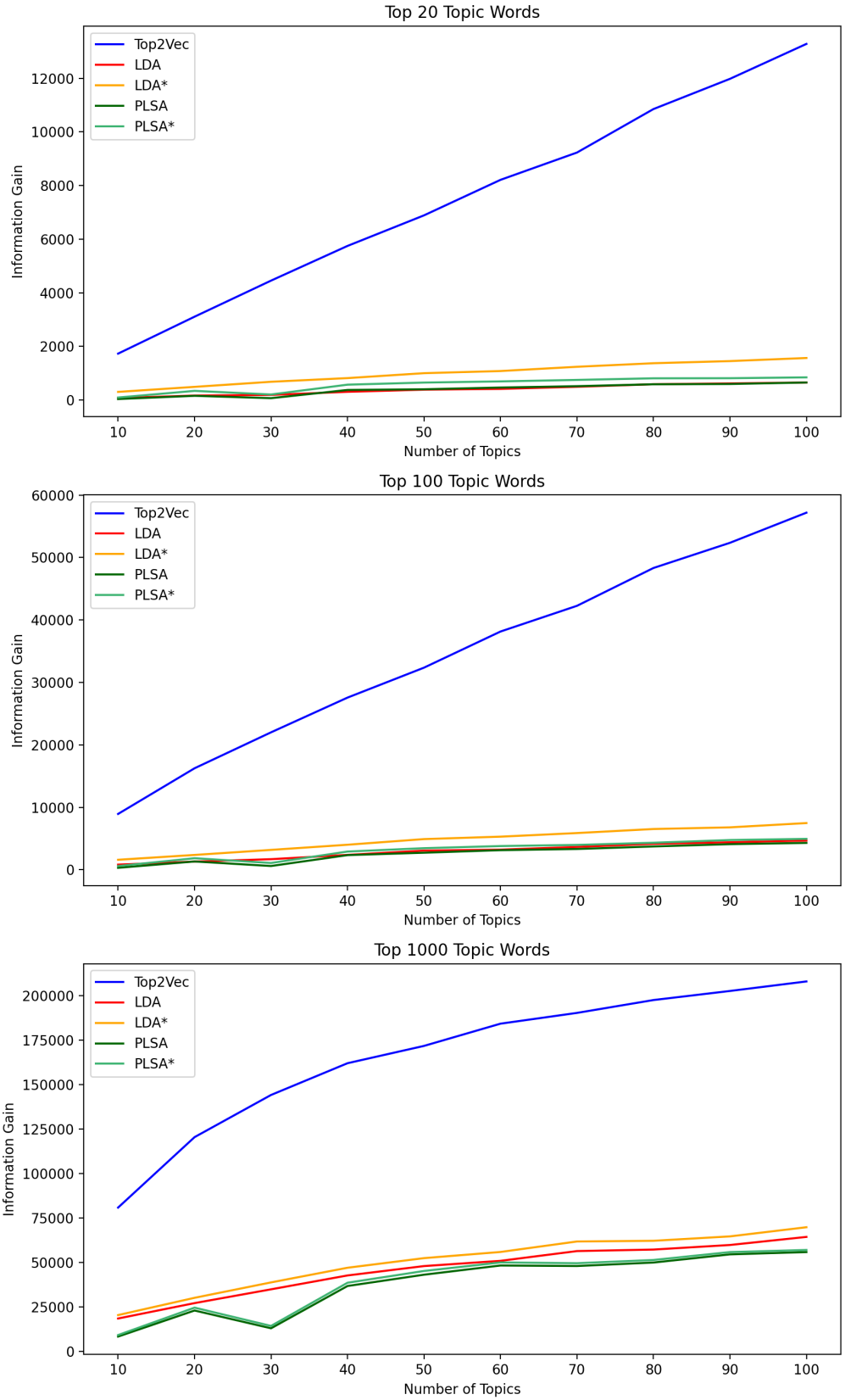

Figure 12: Topic information gain comparison between Top2Vec, PLSA, and LDA trained models on the *Yahoo Answers* dataset. LDA* and PLSA* have stop-words removed.

Table 3: Topic information gain for the top 10 words from `top2vec` topics trained on the *Yahoo Answers* dataset with 10 topics.

Topic Number	Topic Words	PWI(T)
1	overwrite, rebooting, debug, debugging, reboot, executable, compiler, winxp, xp, winnt	112.2
2	securities, unpaid, equity, purchaser, payment, broker, underwriting, issuer, payable, underwriter	104.4
3	regimen, discomfort, inflammation, swelling, psoriasis, puffiness, inflammatory, irritation, edema, hypertension	98.1
4	realtionship, realationship, insecurities, confide, hurtful, inlove, clingy, friendship, bestfriend, friendships	92.5
5	song, sings, singer, sang, artist, duet, album, lyrics, ballad, vocalist	91.9
6	scripture, believers, righteousness, righteous, pious, spiritual, spirituality, sinful, worldly, discernment	82.2
7	team, players, game, teams, scoring, league, teammate, scorers, playoff, defensively	80.0
8	courses, subjects, curriculum, students, teaching, faculty, syllabus, academic, undergraduate, baccalaureate	78.6
9	war, leaders, politicians, government, democracy, political, terrorists, terrorism, partisan, policies	64.6
10	thus, constant, hence, surface, resulting, greater, therefore, becomes, occurs, larger	33.1
		837.6

Table 4: Topic information gain for the top 10 words from LDA topics, *after* stop-word removal, trained on the *Yahoo Answers* dataset with 10 topics.

Topic Number	Topic Words	PWI(T)
1	team, game, world, win, cup, play, de, football, best, player	23.2
2	computer, yahoo, use, get, click, internet, free, com, need, windows	22.3
3	people, us, country, war, world, would, american, bush, government, america	18.0
4	one, water, two, would, light, number, energy, used, earth, use	16.9
5	body, weight, also, doctor, eat, blood, may, day, get, pain	15.7
6	www, com, http, find, song, name, know, anyone, org, music	14.2
7	get, money, school, would, need, work, pay, good, business, job	13.1
8	god, people, one, life, believe, jesus, word, many, would, us	12.5
9	like, know, get, think, would, want, people, good, really, go	9.1
10	time, like, friend, said, guy, back, would, one, years, got	8.3
		153.3

Figure 13: Semantic embedding of *Yahoo Answers* posts with true labels. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

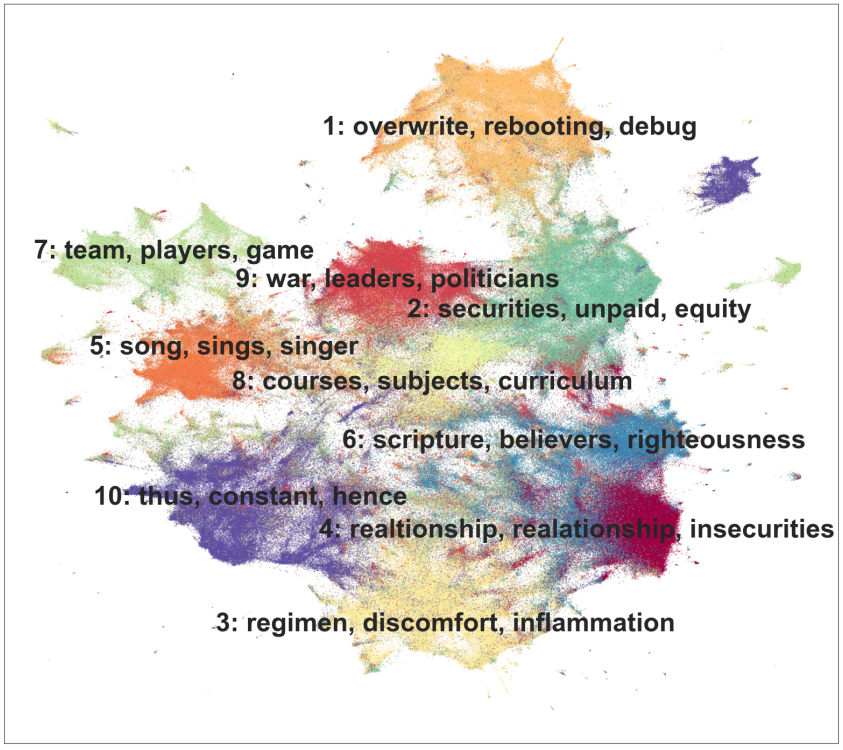

Figure 14: *Yahoo Answers* posts labeled with the `top2vec` topics from Table 3. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

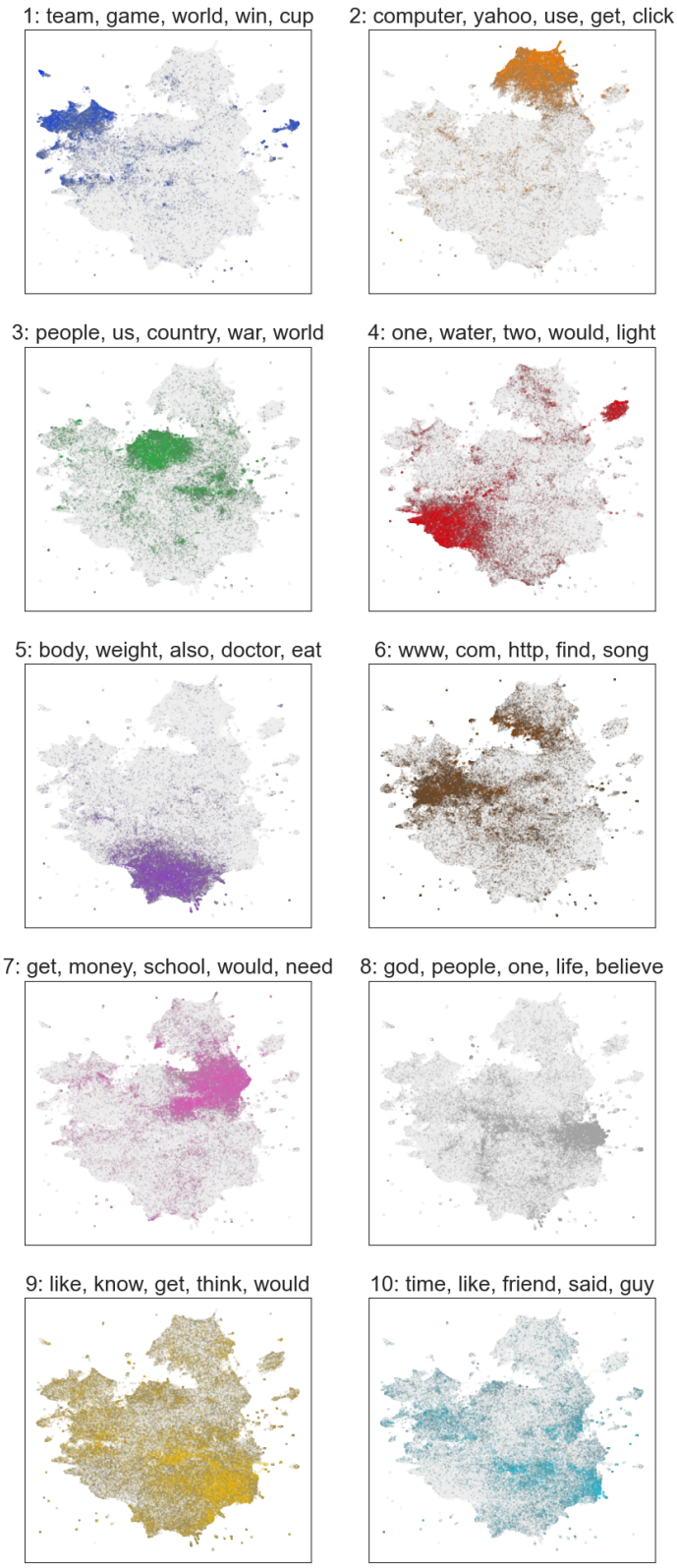

Figure 15: Topic proportion of each LDA topic from Table 4 across all *Yahoo Answers* posts in the semantic embedding. The topics are ordered by decreasing information gain. The 300 dimension document vectors are embedded into 2 dimensions using UMAP.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[2] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[3] Jordan L Boyd-Graber and David M Blei. Syntactic topic models. In *Advances in neural information processing systems*, pages 185–192, 2009.

[4] S. Syed and M. Spruit. Full-text or abstract? examining topic coherence scores using latent dirichlet allocation. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 165–174, 2017.

[5] Kai Yang, Yi Cai, Zhenhong Chen, Ho-fung Leung, and Raymond Lau. Exploring topic discriminating power of words in latent dirichlet allocation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2238–2247, 2016.

[6] Geoffrey E Hinton et al. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[7] Henry Widdowson. J.R. Firth, 1957, papers in linguistics 1934–51. *International Journal of Applied Linguistics*, 17:402 – 413, 10 2007.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[10] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[11] Zhuang Bairong, Wang Wenbo, Li Zhiyu, Zheng Chonghui, and Takahiro Shinozaki. Comparative analysis of word embedding methods for dstc6 end-to-end conversation modeling track.

[12] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[15] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[16] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[17] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *ArXiv*, abs/1405.4053, 2014.

[18] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany, August 2016. Association for Computational Linguistics.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[20] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[21] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[22] Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. Topics in semantic representation. *Psychological review*, 114(2):211, 2007.

[23] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[24] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[25] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[26] Leland McInnes and John Healy. Accelerated hierarchical density based clustering. *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov 2017.

[27] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

[28] RB Marimont and MB Shapiro. Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70, 1979.

[29] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[30] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[32] M Cover Thomas and A Thomas Joy. Elements of information theory. *New York: Wiley*, 3:37–38, 1991.

[33] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[34] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[36] Jamaal Hay Wenpeng Yin and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *EMNLP*, 2019.

[37] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.