Alignment-Based Topic Extraction Using Word Embedding

Tyler Newman[0000−0003−3108−8407] and Paul Anderson[0000−0002−8408−3944]

College of Charleston, Charleston SC 29424, USA
newmantp@g.cofc.edu
andersonpe2@cofc.edu

Abstract. Being able to extract targeted topics from text can be a useful tool for understanding the large amount of textual data that exists in various domains. Many methods have surfaced for building frameworks that can successfully extract this topic data. However, it is often the case that a large number of training samples must be labeled properly, which requires both time and domain knowledge. This paper introduces new alignment-based methods for predicting topics within textual data that minimizes the dependence upon a large, properly-labeled training set. Leveraging Word2Vec word embeddings trained using unlabeled data in a semi-supervised approach, we are able to reduce the amount of labeled data necessary during the text annotation process. This allows for higher prediction levels to be attained in a more time-efficient manner with a smaller sample size. Our method is evaluated on both a publicly available Twitter sentiment classification dataset and on a real estate text classification dataset with 30 topics.

Keywords: Topic extraction · Text annotation · Text classification · Word vectors · Text tagging

1 Introduction

Finding specific topics within textual data is an important task for many domains. A multitude of approaches for achieving this task have appeared in recent years [14], no doubt due to the ever growing amount of textual data available to organizations and researchers. In the most straightforward case, topic labels are known *a priori* and non-domain experts can be trained to manually label examples using systems such as Mechanical Turk [4]. In a second case, topic labels are not known prior to annotation and only after a domain expert has defined them can non-experts be used to label examples. Conversely, once these topic labels have been defined, many domains require an expert throughout the entire annotation process [1, 13]. The fourth case, and the one that motivated our work specifically, is the case in which a domain expert must concurrently evolve the set of topic labels through manual annotation of examples.

In all of these cases, once an appropriate number of training samples have been acquired, many different machine learning algorithms have successfully

been used to automatically identify topics in new examples [14]. Further, reducing the number of training samples needed to produce an accurate predictive model is beneficial in all instances, although the fourth case in particular benefits from a flexible predictive modeling approach that can quickly be retrained on a small number of examples. This is because the domain expert may revise or modify the current topic labels while exploring their problem domain. An example of this scenario arose during a data science project in the real estate domain where a sentence-level topic annotator was desired. Experts in this domain could not specify a final or draft set of topic labels, preventing us from utilizing other approaches that are built to expect a stable set of topics. Further, domain experts required a lightweight, web-based interface where topics could be easily defined, modified, and applied at the sentence level while minimizing the number of training examples needed to produce an automated prediction due to the large number of topics in their domain.

This paper describes a novel method and application for predicting topics at the sentence level in order to reach a high level of accuracy with a limited number of training samples. We evaluate our algorithm on its ability to predict over 30 different sentence-level topic labels within real estate data. We also provide an evaluation of our algorithm on a standard and publicly available twitter sentiment-based prediction dataset. In the real-estate domain, expert knowledge is required for the annotation of the important topics of interest. The topics were not available or known *a priori*, further limiting the number of possible expert annotators. We show that our algorithm achieved a higher prediction accuracy with a very small number of examples, resulting in a significant improvement over standard methods for topic identification with small sample sizes. Finally, our method maintains its ability to predict well in small sample sizes, even in the absence of negative training examples which would further reduce the burden on the domain expert. The rest of this paper is structured as follows: in Section 2 we present a review of related work; in Section 3 we provide descriptions of relevant machine learning algorithms and introduce our architecture and novel methods; and we conclude with Section 6 where we present our experimental results and discussion.

2 Related Work

Kim *et al.* proposes a method for categorizing text at the sentence and document level using word vector distances [6]. Their algorithm helps deal with sparsity issues in word data, specifically caused by words looking very different when actually meaning the same thing. These sparsity issues are often caused by short documents or a small amount of training data. One of the datasets that they used was the SemEval 2013 Task B dataset (Twitter), which contains 12,348 tweets that are labeled as positive, negative, or neutral. They found that their algorithm produced good results quickly and with a relatively small number of training samples (requiring at least sample sizes in the hundreds).

Topic and feature extraction is popular in the world of data mining, and there has been a lot of work for making this process more efficient and more accurate. [18] proposes an algorithm that combines binary classifiers, Conditional Random Fields (CRFs), and Expectation Maximization in order to extract information from business-to-consumer (B2C) emails. Different pieces of the extracted information are annotated as specific features. This is done by utilizing templates that have been predefined based on other similar documents. Their approach is fully unsupervised and requires no manual corrections or fixes to the data. Instead of using word vectors for matching, they run low-accuracy annotators trained on weak features which then feed into their CRF model.

Nguyen *et al.* introduces an algorithm that combines preprocessing, pattern recognition, iterative model development, and active learning to annotate and classify features found within clinical text records [13]. In their algorithm, the textual data is first standardized and normalized to perform tasks such as correcting spelling, expanding abbreviations, and converting to a standard layout. The data then moves to the iterative model development process, where models are trained and evaluated using Support Vector Machines (SVMs) and CRFs. The model is refined using a visual annotator that allows for some manual correction, along with active learning that lets the learner select the most informative data to retrain the model. Their approach requires a relatively large number of training samples. In one of their tests, they ran various active learning algorithms on 100 batches of radiology reports with 10 reports per batch. The F-scores for the active learning algorithms, on average, did not surpass 75% until around 7-10 batches (i.e., 70-100 reports) were run. It also took the algorithms between 30-50 batches to reach a 90% F-score.

Wang *et al.* offers a new approach to modeling targeted subtopics within text. Instead of extracting all larger topics within a corpus, they search for more specific subtopics using a *targeted topic model* (TTM) [17]. To do this, each sentence is treated as its own topic that focuses on only one aspect. These topics are deemed relevant or irrelevant based on a set of specified keywords. Their model was run on five datasets taken from Twitter that range in size from 10k to 50k samples.

Finally, several attempts have been made to create architectures that can be used for new domains. [8] proposes a Python framework to ameliorate the process of feature extraction in various different forms of media such as video, audio, and text. Their framework, Pliers, attempts to package the benefits of multiple other machine learning frameworks and services into one coherent feature extraction toolbox.

3 Methods

3.1 Standard Approaches

We will now present brief descriptions of several widely applied methods used for target identification. This section is broken up into two additional subsections:

feature extraction and machine learning methods. All of the methods described in this work rely on one of two word embedding methods: bag-of-words (BOW) or Word2Vec [9, 10]. We briefly describe how these methods were implemented and incorporated into our work.

Feature Extraction In its simplest form, BOW is an orderless representation of word frequencies in a document. In the context of this sentence-level target identification problem, the word counts from each sentence are normalized into a word frequency matrix prior to classification. The Python natural language toolkit (nltk) and native Python String library [2] were used for this step. Python's String library was used to parse out punctuation and stop words were removed using nltk. This was followed by stemming using nltk's SnowballStemmer [2].

Word2Vec is an NLP system that utilizes neural networks in order to create a distributed representation of words in a corpus [9, 10]. While the BOW pipelines produce word frequency for each document respectively, Word2Vec creates vectors for each word present in a document. These vectors have a smaller distance between them for related words. The words Athens and Greece are examples of this, along with pluralities or tense switches, such as alumnus and alumni or walking and walked [10]. In order to map words to vectors, Word2Vec uses an underlying shallow neural network in addition to techniques seen in deep learning tasks. This unsupervised task takes each individual sentence for a given corpus and, within the neural network, encodes the context of word in the sentence, much like the deep learning autoencoders seen in restricted Boltzmann machines and Stacked Denoising Autoencoders [15, 16]. This is done through the usage of skip-grams, which calculate the probabilities of occurrence for words within a certain distance before and after a given word. Inter-relating these probabilities creates similar word vectors for those with higher probabilities.

For the purposes of evaluation in this paper, two Word2Vec models were used. The first was a model trained on the real estate corpus, and the second was a publicly available Twitter Word2Vec pre-trained model available at `http://yuca.test.iminds.be:8900/fgodin/downloads/word2vec_twitter_model.tar.gz`.

Machine Learning Methods Three standard machine learning methods that are often used in topic identification were selected for comparison: Naïve Bayes, SVM, Random Forest (RF), and K-Nearest Neighbor (KNN). Standard implementations of these algorithms are available in scikit-learn. Naïve Bayes was applied to the BOW features [7] using empirical priors and non-parametric settings. SVMs with BOW features have been shown to perform well on a wide range of text classification applications [14]. SVMs are not prone to error with high-dimensional datasets and have been previously shown to be useful in text-based classification problems [5]. Four standard kernels were tested: linear, polynomial, radial basis, and sigmoid. The penalty parameter (C) was also varied as 0.01, 0.1, 1, and 10. For the presentation of the results, a single entry for SVMs is displayed that corresponds to the best parameter selection for each class. KNN

with two standard distance metrics was tested. Both distance metrics are built upon Word2Vec embeddings as opposed to BOWs. The two metrics used were the mean and maximum cosine-similarity between all pairs of words.

3.2 Novel Approaches

Two novel approaches were developed and evaluated in this work that stem from an alignment-based distance metric. The first approach is a standard KNN classifier utilizing the novel alignment-based distance metric in place of traditional BOW distance metrics. To classify a new unknown sample, the alignment-based KNN calculates the distance between the target sample and all labeled data. The k-nearest neighbors are then found and the majority class is returned. The second approach is an alignment-based threshold classifier that only requires positively labeled data and a predefined threshold. This threshold-based classifier measures the distance from a target unknown sample to only the positively annotated samples. If the score is above the threshold, a positive class prediction is returned. The success of both methods is dependent on the alignment-based distance metric described below.

Alignment Distance Metric All alignment-based approaches were implemented using the NeedlemanWunsch algorithm that has been made famous for its use in aligning biological sequences [12]. We have adapted the scoring and gap penalties for the target identification classification problem. The algorithm produces a numerical score based on the aggregation of misalignment penalties between words (cosine-similarity) combined with the penalties for skipping a word in either the labeled or unlabeled sentence. The cosine similarity misalignment score is the dot product between the vector representation of each word. There is no gap penalty for skipping a word in the unlabeled sentence; however, skipping a target word carries a high penalty and is therefore avoided by the algorithm. This forces the algorithm to match all target words that have been identified by a domain expert. An example alignment of two sentences is shown in Table 1. For all alignment-based methods, the Word2Vec implementation described in [10] was used.

Standard Distance Metrics In addition to these two novel approaches, we implemented and tested two standard Word2Vec distance metrics. The first metric was the maximum cosine-similarity between pairs of words in annotated and unknown sentences. The second was the average cosine-similarity score between pairs of scores.

4 Use Cases

4.1 Real Estate

The domain we designed our original system for involved property descriptions from real estate data. This data was created by realtors and contains information

Table 1: Example alignment of two sentences with an aggregate alignment score of 0.91. The first sentence has been annotated as a positive example of the Master Bedroom Downstairs target class. The second sentence has no known annotation. The optimal alignment shown results in a score of 0.91 with misalignment occurring between bedroom and bedrooms. Gaps in the annotated sentence matched to words in the unknown sentence are not penalized.

Annotated:	— master bedroom — ——- downstairs
Unknown:	two master bedrooms are located downstairs

about various real estate listings. Each listing has a large amount of metadata (e.g., location, images, basic features); however, the main piece of information we use in our system is the description written for each listing. This description is usually no more than a short paragraph and is created by the real estate agent to summarize the listing as a whole. It can include any information that the agent wishes to convey to the potential buyer, such as property features, kitchen appliances, etc. By analyzing a listing's description, we attempt to extract specific features about the listing itself.

We created a web application using the Angular JavaScript framework. The main interface for this application provides a way to easily and quickly annotate real estate listings. This interface pulls listings that have not yet been annotated from our server, along with the sentence and word data associated with that listing. It also retrieves the topic information so that the listing can be properly annotated.

Once a listing has been retrieved, the annotator can select a sentence from the listing description to annotate. They can then toggle the individual words in the sentence that match a specific topic. If they believe that a pattern should be mapped to a topic that does not currently exist, then they have the option to create it. Once a topic has been defined, it can then be used by future annotators. When an annotator is satisfied with an annotation, they simply click submit and it is stored inside of the annotations database to be used in future alignment predictions.

Along with this annotation interface, we also included other pages within the web application that provide statistical information regarding the database. Some of these pages display simple information, such as the specific progress of different annotators. Other pages give annotators more control over the database itself, allowing them to take actions such as correcting mistakes in previous annotations or modifying topic names.

4.2 Twitter

Twitter has grown quite large in the past decade, along with the amount of textual data it has created. This data has proven to be a popular source for test-

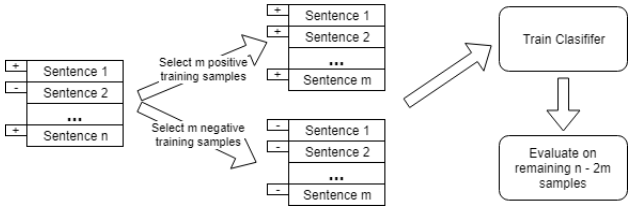


Fig. 1: One iteration of cross-validation. This was repeated 20 times in total for training set sizes 2 to (n/2 - 2)

ing and implementing text-based machine learning algorithms [3, 6, 11, 17]. We tested our alignment algorithm on the Twitter dataset used in [6]. This dataset includes the text data for 12,348 tweets and annotations that have been made for each tweet. These annotations are based on the entire tweet and represent the overall sentiments conveyed therein. The possible sentiment values include positive, neutral, negative, objective, and objective-OR-neutral. In our experiment and the experiment done by [6], only positive, neutral, and negative sentiments are used.

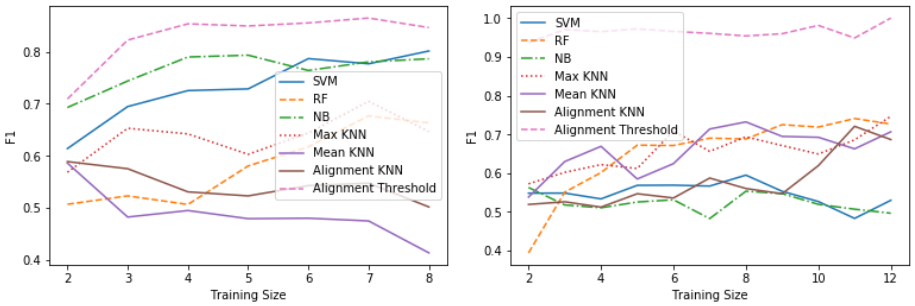
In order to annotate this dataset, we created a simple interactive widget that runs inside of a Python Jupyter Notebook. This publicly available widget is very similar to the annotation interface we created for the real estate data and can be seen in supplemental Figure 4. We store the tweets in a text file, which the interface uses to select tweets at random. Once a tweet is selected, the annotator can toggle the words in the tweet that they believe match the sentiment that was predicted. For example, if a tweet was labeled as having a positive sentiment, then only words that are relevant to that sentiment will be selected and stored in the annotation. This allows us to store annotations in a very similar format to the ones that were stored for our real estate data.

5 Evaluation

Each classifier was subjected to iterative cross-validation as a function of the training size for each category. This procedure is summarized in Figure 1. The average F_1 score across all iterations for each training set size was calculated and plotted as a function of training set size. Two examples are shown in Figure 2. The area under this curve was then calculated to measure the ability of each classifier to perform well for small as well as larger training set sizes. The 95% confidence interval was calculated for the area under the F_1 curve as a function of the training set size.

6 Results and Discussion

We compared our alignment-based algorithms to bag-of-words derived SVM, Naïve Bayes, random forest, and KNN classifiers on 30 real estate categories



(a) C. Fenced Backyard (b) D. Eat-in Kitchen

Fig. 2: Two sample categories showing the F_1 score versus the training set size for the five different classifiers tested.

and positive/negative sentiment tweet prediction. Four algorithms based on Word2Vec word embedding were evaluated. Two standard distance metrics (mean and max) were evaluated as baseline classifiers in addition to two novel alignment-based classifiers (Alignment KNN and Alignment Threshold). These results are summarized in Table 2 which shows the iterative cross-validation confidence intervals of the area under the F_1 curve versus training set size. The table is sorted by the average score for the Alignment Threshold method. The first column in the table represents a summary of how this method performed in comparison to the others. If the Alignment Threshold method had a higher and non-overlapping confidence interval when compared to all other methods, this is indicated with a *W+*. If the lower bound on the Alignment Threshold method was higher than any other method and *does* overlap, this is indicated with a *W*. A *T* was used if the Alignment Threshold confidence interval was not higher than any other method but still overlapped the best method. All other cases are indicated with an *L*. In total, there are 29% *W+*, 35% *W*, 19% *T*, and 16% *L*, meaning that the Alignment Threshold method was as good or better in 84% of the classes. These results demonstrate how the alignment-based algorithms are able to perform better on fewer samples or equivalent to standard approaches. This is significant even in the case of equivalent accuracy as the alignment-based threshold algorithm does not require negative training samples which reduces the number of samples an expert must annotate. Further, all alignment-based methods are built upon a semi-supervised learning approach where large amounts of unlabeled data is used to reduce the need for labeled data.

It was our desire to test our algorithm against the multi-level kernel system developed by Kim *et al.* discussed in section 2; however, we were unable to find a public implementation of this algorithm, and our attempts to reach the authors were not successful. In their paper, the authors evaluated their algorithm using the Twitter dataset also used in this paper [11], which includes 12,348 tweets that are each matched with an overall sentiment value. The multi-level kernel system

was able to achieve accuracy between around 80% and 90% using samples sizes ranging from the low hundreds to the mid thousands, with their best accuracy result being 0.808 (with a standard test data split of 25%). No results were presented for sample sizes less than 50.

Further inspection of Table 2 shows that in some cases the Alignment Threshold classifier performs poorly while the Alignment KNN classifier, which uses both positive and negative training examples, performs better or equivalent to standard approaches. We believe this is due to the underlying word embedding vectors for the specific targets. Words such as those found in positive and negative tweets are relatively ubiquitous, while those found when mentioning a walk-in closet are relatively rare. It is reasonable to assume that hard coded rules could also be developed in some cases, but that the approach presented in this paper would be preferred as it is easily extended to additional categories without the need to maintain a rule management system.

The original annotation system was researched and built specifically for a company specializing in real estate technology. Because this data is proprietary, our algorithms are also tested our algorithm on a publicly available Twitter sentiment prediction dataset. The original Javascript Angular based system is proprietary, but we have implemented the core functionality of our system using Jupyter Notebook widgets. The widget loads directly inside of the Notebook and displays a tweet's text, its overall sentiment value, and buttons that represent each word in the tweet. These buttons can be toggled to create the annotation pattern. All of this data is stored in a Pandas DataFrame and serialized to its own file, which can be used to train and evaluate our algorithm. We have made this version of the annotation interface open source in addition to all alignment-based implementations and evaluation code (`https://github.com/Anderson-Lab/sentence-annotation`).

7 Conclusions

Being able to extract topics from text is an ever-growing problem for many domains given the large amount of textual data that is constantly being created. Therefore, it is necessary to minimize the amount of annotating needed to achieve high levels of accuracy. To accomplish this, we introduced a novel topic prediction algorithm that requires only a small amount of human annotation. Our results show that this approach can provide significant performance benefits when the target labels are not known *a priori* or when the sample size is small. Future directions of this work include experiments to determine if these alignment-based distance metrics continue to provide non-redundant benefits as the sample size grows significantly. For community and reproducibility purposes, our methods are available in a public repository that includes a Jupyter Notebook annotation widget that allows for annotation to easily be carried out on other real world datasets.

Table 2: Area under the F_1 vs. training set size curve. 95% confidence intervals are calculated for each method. Categories are ordered by the average F_1 Alignment Threshold score.

	Category	# Samples	SVM	RF	NB	Max KNN	Mean KNN	Align. KNN	Align. Threshold
W+	Screened Porch	21	0.6 - 0.68	0.74 - 0.83	0.63 - 0.71	0.67 - 0.76	0.69 - 0.76	0.72 - 0.79	0.88 - 0.9
W	Walk in closet	25	0.78 - 0.83	0.83 - 0.88	0.75 - 0.81	0.67 - 0.76	0.56 - 0.64	0.68 - 0.75	0.88 - 0.9
W+	Granite Countertops	31	0.62 - 0.69	0.64 - 0.73	0.64 - 0.71	0.6 - 0.7	0.7 - 0.79	0.57 - 0.67	0.86 - 0.9
W	Stainless Steel Appliances	20	0.7 - 0.76	0.77 - 0.85	0.74 - 0.8	0.62 - 0.71	0.51 - 0.59	0.7 - 0.77	0.81 - 0.85
W+	Eat-in Kitchen	11	0.41 - 0.5	0.5 - 0.61	0.39 - 0.48	0.5 - 0.59	0.49 - 0.61	0.43 - 0.53	0.79 - 0.82
T	Near beach	17	0.73 - 0.79	0.69 - 0.77	0.75 - 0.81	0.63 - 0.72	0.8 - 0.84	0.77 - 0.83	0.76 - 0.82
W+	Sunroom	10	0.46 - 0.55	0.51 - 0.63	0.52 - 0.62	0.53 - 0.62	0.48 - 0.62	0.52 - 0.63	0.77 - 0.8
W	Open Floor Plan	22	0.73 - 0.8	0.73 - 0.82	0.66 - 0.73	0.62 - 0.72	0.56 - 0.65	0.75 - 0.83	0.76 - 0.81
W+	Dual Vanities	10	0.6 - 0.69	0.58 - 0.69	0.58 - 0.68	0.45 - 0.56	0.51 - 0.6	0.48 - 0.6	0.77 - 0.8
W	Crown molding	13	0.63 - 0.7	0.45 - 0.56	0.49 - 0.59	0.58 - 0.68	0.71 - 0.79	0.58 - 0.68	0.75 - 0.81
W+	Heart of Pine Flooring	12	0.59 - 0.67	0.58 - 0.7	0.59 - 0.67	0.54 - 0.65	0.62 - 0.71	0.61 - 0.71	0.73 - 0.81
W+	Breakfast Bar	8	0.57 - 0.65	0.52 - 0.63	0.58 - 0.66	0.45 - 0.56	0.43 - 0.55	0.47 - 0.59	0.75 - 0.77
W	Built-In Cabinets	16	0.64 - 0.72	0.64 - 0.74	0.61 - 0.69	0.6 - 0.69	0.63 - 0.73	0.66 - 0.74	0.72 - 0.79
T	Tennis Courts	9	0.62 - 0.7	0.57 - 0.7	0.53 - 0.63	0.75 - 0.79	0.68 - 0.74	0.75 - 0.79	0.7 - 0.77
T	Pool	10	0.62 - 0.7	0.58 - 0.68	0.69 - 0.75	0.68 - 0.74	0.61 - 0.69	0.62 - 0.7	0.69 - 0.75
W	Garden Tub	7	0.6 - 0.67	0.6 - 0.69	0.38 - 0.49	0.44 - 0.55	0.28 - 0.37	0.38 - 0.49	0.69 - 0.73
W	StructureUnit Brick Exterior	7	0.62 - 0.7	0.48 - 0.6	0.58 - 0.65	0.55 - 0.63	0.43 - 0.53	0.49 - 0.59	0.69 - 0.73
T	Cul-De-Sac	8	0.62 - 0.69	0.63 - 0.71	0.59 - 0.68	0.68 - 0.74	0.63 - 0.71	0.67 - 0.72	0.68 - 0.74
W+	Vaulted Ceiling	8	0.43 - 0.52	0.5 - 0.61	0.48 - 0.58	0.48 - 0.58	0.55 - 0.63	0.45 - 0.56	0.68 - 0.73
L	Near Shopping	23	0.84 - 0.88	0.63 - 0.73	0.78 - 0.83	0.73 - 0.8	0.86 - 0.89	0.76 - 0.82	0.66 - 0.74
W	Open Kitchen	10	0.46 - 0.55	0.55 - 0.67	0.5 - 0.61	0.44 - 0.55	0.36 - 0.47	0.36 - 0.45	0.65 - 0.73
W	Gas Log Fireplace	9	0.57 - 0.66	0.58 - 0.69	0.46 - 0.56	0.47 - 0.59	0.43 - 0.58	0.53 - 0.63	0.61 - 0.71
W	Master Bedroom Downstairs	8	0.48 - 0.59	0.46 - 0.6	0.35 - 0.46	0.44 - 0.56	0.42 - 0.52	0.54 - 0.63	0.61 - 0.7
T	Natural Light	15	0.63 - 0.71	0.5 - 0.6	0.59 - 0.68	0.53 - 0.64	0.62 - 0.7	0.62 - 0.72	0.6 - 0.68
L	Near Downtown	15	0.67 - 0.74	0.69 - 0.78	0.82 - 0.85	0.71 - 0.78	0.68 - 0.75	0.74 - 0.8	0.59 - 0.68
W+	StructureUnit Deck	8	0.43 - 0.54	0.39 - 0.5	0.43 - 0.52	0.42 - 0.53	0.37 - 0.5	0.42 - 0.54	0.59 - 0.67
W	Fenced Backyard	7	0.51 - 0.6	0.37 - 0.5	0.53 - 0.62	0.43 - 0.53	0.32 - 0.41	0.35 - 0.47	0.59 - 0.67
L	Gas Range	9	0.5 - 0.59	0.47 - 0.6	0.55 - 0.63	0.54 - 0.65	0.68 - 0.75	0.42 - 0.52	0.57 - 0.66
T	tweets	23	0.53 - 0.61	0.39 - 0.48	0.49 - 0.58	0.43 - 0.52	0.45 - 0.54	0.52 - 0.6	0.52 - 0.6
L	Near Restaurants	7	0.61 - 0.67	0.49 - 0.62	0.5 - 0.59	0.61 - 0.67	0.59 - 0.65	0.6 - 0.67	0.49 - 0.61
L	Professionally Landscaped	7	0.45 - 0.55	0.32 - 0.41	0.41 - 0.51	0.49 - 0.59	0.37 - 0.47	0.41 - 0.51	0.38 - 0.47

References

1. Andrzejewski, D., Zhu, X., Craven, M.: Incorporating domain knowledge into topic modeling via Dirichlet Forest priors. In: Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09. pp. 1–8 (2009). https://doi.org/10.1145/1553374.1553378, http://portal.acm.org/citation.cfm?doid=1553374.1553378
2. Bird, S., Klein, E., Loper, E.: Natural language processing with Python. No. January 2009, O'Reilly Media, Inc. (2009). https://doi.org/10.17509/ijal.v1i1.106, https://books.google.com/books?id=KGIbfiiP1i4C{\&}pg=PR5
3. Boella, G., Caro, L.D., Ruggeri, A., Robaldo, L.: Learning from syntax generalizations for automatic semantic annotation. Journal of Intelligent Information Systems **43**(2), 231–246 (2014). https://doi.org/10.1007/s10844-014-0320-9
4. Buhrmester, M., Kwang, T., Gosling, S.D.: Amazon's mechanical Turk: A new source of inexpensive, yet high-quality, data? Perspectives on Psychological Science **6**(1), 3–5 (2011). https://doi.org/10.1177/1745691610393980
5. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 1398, pp. 137–142 (1998). https://doi.org/10.1007/s13928716
6. Kim, J., Rousseau, F., Vazirgiannis, M.: Convolutional Sentence Kernel from Word Embeddings for Short Text Categorization. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (September), 775–780 (2015)
7. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval pp. 4–15 (1998). https://doi.org/10.1007/BFb0026666, http://link.springer.com/10.1007/BFb0026666
8. McNamara, Q., de la Vega, A., Yarkoni, T.: Developing a comprehensive framework for multimodal feature extraction. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2017). https://doi.org/10.1145/3097983.3098075
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. CoRR **abs/1301.3**, 1–12 (2013). https://doi.org/10.1162/153244303322533223, http://arxiv.org/abs/1301.3781
10. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. Proceedings of NAACL-HLT (June), 746–751 (2013). https://doi.org/10.3109/10826089109058901, http://www.aclweb.org/anthology/N13-1090
11. Nakov, P., Rosenthal, S., Ritter, A., Wilson, T.: SemEval-2013 Task 2: Sentiment Analysis in Twitter. Proceedings of the International Workshop on Semantic Evaluation (SemEval-2013) **2**(SemEval), 312–320 (2013)
12. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48**(3), 443–453 (1970). https://doi.org/10.1016/0022-2836(70)90057-4
13. Nguyen, H., Patrick, J.: Text Mining in Clinical Domain. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16 pp. 549–558 (2016). https://doi.org/10.1145/2939672.2939720, http://dl.acm.org/citation.cfm?doid=2939672.2939720
14. Pawar, P., Gawande, S.: A comparative study on different types of approaches to text categorization. International Journal of Machine Learning and Computing **2**(4), 423–426 (2012). https://doi.org/10.7763/IJMLC.2012.V2.158

15. Srivastava, Salakhutdinov: Multimodal Learning with Deep Boltzmann Machines. Advances in neural information processing systems (NIPS) **15**, 2222–2230 (2012). https://doi.org/10.1109/CVPR.2013.49
16. Vincent PASCALVINCENT, P., Larochelle LAROCHEH, H.: Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol. Journal of Machine Learning Research **11**, 3371–3408 (2010). https://doi.org/10.1111/1467-8535.00290
17. Wang, S., Chen, Z., Fei, G., Liu, B., Emery, S.: Targeted Topic Modeling for Focused Analysis. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16 (2016). https://doi.org/10.1145/2939672.2939743
18. Zhang, W., Ahmed, A., Yang, J., Josifovski, V., Smola, A.J.: Annotating Needles in the Haystack without Looking. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15 pp. 2257–2266 (2015). https://doi.org/10.1145/2783258.2788580, `http://dl.acm.org/citation.cfm?doid=2783258.2788580`

8 Supplemental

Our work was directly motivated to reduce the burden of interaction with human supervisors, and therefore, we present an overview of our architecture that can be broken up into four distinct phases: preprocessing, annotation, learning, and evaluation. Figure 3 illustrates these phases and the steps that are taken in each.

Preprocessing & Cleaning The first step in our pipeline involves processing the raw listing description data and converting it into a consistent format. Each description is stripped of any extra whitespace, transformed into all lowercase letters, and decoded using the UTF-8 character encoding. Once this has been done, the description is separated into its constituent sentences. These sentences are then broken up and become lists of words that can be used in our alignment algorithm.

Annotation In order to expedite the annotation process, we made two simple annotation interfaces; one for each of the datasets that we tested our algorithm on. A screenshot of the Jupyter widget is shown in Figure 4.

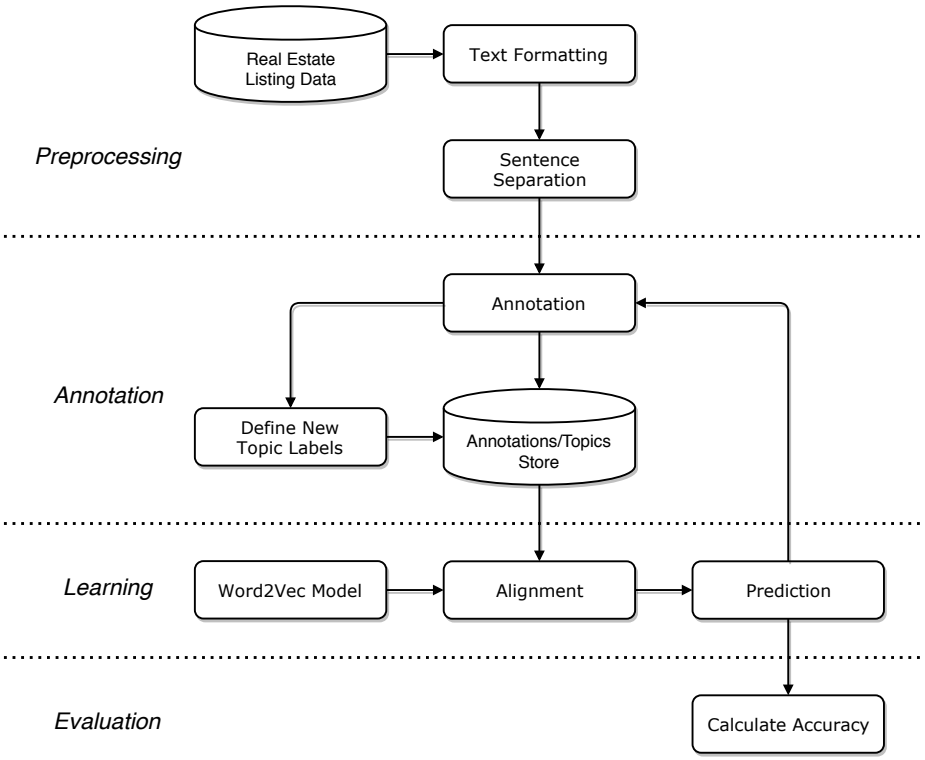


Fig. 3: Architecture Pipeline

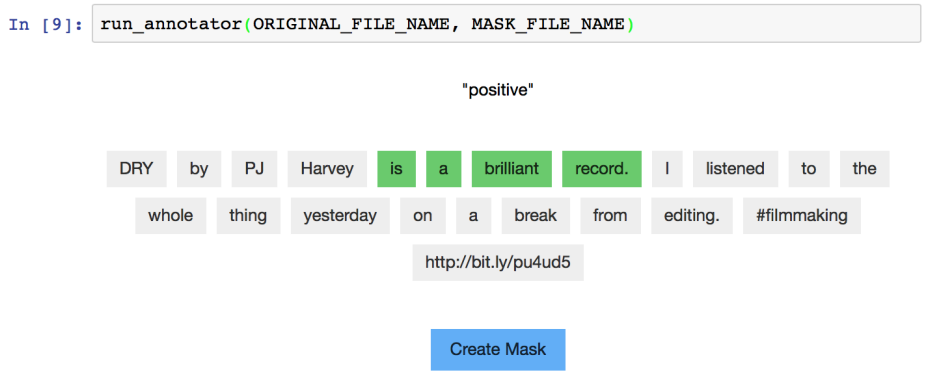


Fig. 4: Python Tweet Annotation Tool