Capstone Project: Machine Learning Engineer Nanodegree

Ryan Skorupski

September 22, 2016

Definition:

Project Overview:

An equity investor can invest either publicly or privately. On the private side, the information available about potential investments can be rather sparse. Of course, there are websites like Crunch Base and professional products like Pitch Book that help angel, private equity, and venture capital investors understand the private (and mainly startup) landscape better [1] [2]. However, if you don't want to spend a lot of money and you want to get a better understanding of the landscape you can scrape sites like Crunch Base or Angel.co.

One important aspect of investing in a startup is understanding its competitive landscape: is it first in this space, are there a lot of other start ups trying something similar, etc.

This project will use company descriptions and labels (tags) scraped from Crunch Base to create a tool that returns a firms competitors (or most similar companies). The project aims to automate how we identify a potential investment's competitors.

Problem Statement:

The goal is to create a tool that when given a firms name returns its ten closest competitors. The tasks involved are:

1. Collect and store the data (the data will have 5000 observations: company name, description, and tags). The collection process uses selenium to scrape the wanted information and stores it as a csv file.
2. Clean the data. More specifically, prepare the data for feature extraction: remove numbers, remove special characters, convert everything to lower case, etc.
3. Extract Features. For this project, features will be extracted a few different ways in an attempt to find the best solution.
4. Implement an unsupervised learning model to return 10 competitors (most similar companies). For this project, three different implementations will be explored.

---

1 https://www.crunchbase.com/app/search/companies
2 http://pitchbook.com/

Metrics:

For this type of problem identifying a quantitative metric is nearly impossible. We have an unsupervised problem and we don't know the answer.

However, I will use a function that hopes to capture some qualitative measure (or better put, leverage a person's ability to recognize a competing company based on description). The function will simply just take a company name and return the queried company with its descriptions and the competitors with their description. By reading the returned information you can get an intuitive feel for how good the model performs.

On a complexity note, I return the training time for each model. In case the speed of training is important to a user of the function.

Analysis:

Data Exploration:

The data is found in a csv called 'companies'. It contains 5000 observations with each observation having a name, description, and list of keywords. The keywords are labels or tags related to the industry the company is in and can be empty (54 observations have no tag). Both name and description have information for all observations.

To explore the data, run the jupyter notebook named 'DataExploration_Visualization'. The notebook performs some basic exploratory NLP (natural language processing) analysis on both the company descriptions and the keywords.

By leveraging the python NLTK package, all company descriptions are combined into a single string for analysis [3]. The description string has 14,282 unique tokens (words, numbers, or punctuation) and 189,521 tokens in it. That gives us a lexical richness measure of approximately 7.5% (lexical richness is: distinct tokens/total number of tokens). Further analysis of the string, which can be seen in the jupyter notebook give us the most common bigrams and the 50 most common tokens.

Exploratory Visualization:

One choice for visualization is a lexical dispersion plot. It used to show the frequency of certain words in a string in relationship to where the word occurs. An example of this is in the jupyter notebook, however, this visualization is not very useful here.

A better visualization is a word cloud on the keywords [4]. A word cloud is a grouping of the most popular words in a structure where the words that appear the most are the largest. Because the

3 http://www.nltk.org/

4 http://peekaboo-vision.blogspot.de/2012/11/a-wordcloud-in-python.html

keywords here may also be bi, tri, or quad grams, I convert all bi, tri, and quad grams to unigrams by removing their spaces.

Above is the word cloud that was generated by the keywords, it can also be found in the jupyter notebook. The word cloud informs us of the more popular keywords (industries). It appears semiconductor, medical device, and health care service companies are more represented in the data.

Algorithms and Techniques:

For this project, I will try three different methods and choose the one that performs the best. For each implementation two things are important:

1. Feature extraction (the way in which we vectorize our string data)
2. The unsupervised technique chosen to return the competitors

Below, I will expound on both of the pieces above for each of the methods. Also, each approach has its own jupyter notebook associated with it. For the code and some more comments, please refer to the specific implementation's notebook.

Approach I: Naive Nearest Neighbor Search

The first approach will be to obtain a baseline for an extremely simple Nearest Neighbor Search [5]. That is we will do a very naive feature extraction and create the list of similar companies by just measuring the proximity of the firms that are closest.

5 http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html#sklearn.neighbors.NearestNeighbors

Approach I: Feature Extraction:

The features will be extracted from the firms' descriptions. The features will just be a bag of words (just a count of the words in the description). We will not drop any stopwords, or create any n-grams (except of course the unigrams that are the words).

Approach I: The unsupervised technique chosen to return the competitors

I use the default parameters (except for number of neighbors, which we set to 10) for Scikit Learn's unsupervised learner for implementing neighbor searches (sklearn.neighbors.NearestNeighbors).

Note: The distance here is minkowski. Minkowski distance is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance

Approach II: Content-Based Recommendation Engine

This approach will be similar to how sites without user information may recommend similar items based on item content (item descriptions, images, etc.). Because the data I have is a firm's description, I can perform some type of feature extraction and use a recommendation engine to recommend the most similar firms to it.

Approach II: Feature Extraction

For this approach, I will create the features by doing TF-IDF (Term Frequency - Inverse Document Frequency) to parse through the descriptions, identify distinct phrases in each item's description, and then find 'similar' products based on those phrases [6]. In our case TF-IDF will work by looking at all one, two, and three-word phrases (uni-, bi-, and tri-grams) that appear multiple times in a description (the "term frequency") and divides them by the number of times those same phrases appear in all product descriptions. So terms that are 'more distinct' to a particular product get a higher score, and terms that appear often, but also appear often in other products get a lower score. Note: we will also drop all stop words (commom words like: the, we, a).

Approach II: The unsupervised technique chosen to return the competitors

Once the features are obtained, we will measure a similar firm with cosine similarity. We will do this by using SciKit Learn's linear_kernel (which in this case is equivalent to cosine similarity) [7].

Approach III: Doc2Vec Similar Search

This approach will treat each description as a document and convert the words into an embedding and the firms themselves into a vector embedding. Then we will just look at the most similar other firm embeddings using cosine similarity.

---

6 http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
7 http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.linear_kernel.html

Approach III: Feature Extraction

Doc2vec (aka paragraph2vec, aka sentence embeddings) modifies the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents [8]. Note: distributed memory model is the default when running Doc2Vec.

The algorithm runs through the sentences iterator twice: once to build the vocab, and once to train the model on the input data, learning a vector representation for each word and for each label in the dataset. Here the input data is the the firm's description and the label is the the firm's name.

One caveat of the way this algorithm runs is that, since the learning rate decrease over the course of iterating over the data, labels which are only seen in a single LabeledSentence during training will only be trained with a fixed learning rate. This frequently produces less than optimal results. To obtain better results I will iterate over the data several times.

Note on space complexity: With the current implementation, all label vectors are stored separately in RAM. In this case, a unique firm name per description, causes memory usage to grow linearly with the size of the corpus.

Approach III: The unsupervised technique chosen to return the competitors

To measure the similarity of the firm, we use the Gensim's Doc2Vec built in 'most similar' function. This method computes cosine similarity between a simple mean of the projection weight vectors of the given docs. The default number of most similar returned by this function is 10 (topn=10).

Benchmark:

The benchmark for this project is going to be the first implementation, which is the naive nearest neighbor search. This implementation is chosen as the benchmark because it is one of the simplest ways to attempt to create a competitor search tool.

Taking the the benchmarking a bit further, it would be very time consuming to use the qualitative tool/function outlined in the metric section on every company. Therefore, I chose three companies at random to use as a benchmarking set for the qualitative tool. That is, the same three companies will be checked with each implementation to gauge performance between them.

Methodology:

Data Preprocessing:

Each implementation has a slightly different feature extraction procedure. Because of this, the data is

8 https://radimrehurek.com/gensim/models/doc2vec.html

preprocessed slightly different for each case. The exact code and additional comments for the feature extraction can be found in the implementation's respective jupyter notebook, under the train function.

Approach I: Data Preprocessing

The data preprocessing for the naive case is very simple. The data only needs to be converted to lower case and then converted to a bag of words. This is accomplished by using Scikit Learns CountVectorizer.

Approach II: Data Preprocessing

For the content recommendation engine case, words are converted to lower case, punctuation and numbers are dropped, stop words are dropped (stop words are common words like a, the, he, etc), and bi and tri grams are also created. Then instead of a simple frequency for each word, a TF-IDF score is calculated. TF-IDF is explained in the 'Algorithms and Technique' section. This is all easily accomplished using Scikit Learns TfidfVectorizer.

Approach III: Data Preprocessing

For the Doc2Vec case, words are converted to lower case, all non letters are removed, and anything more than a single space is converted to a single space. For this case, this is done with a lower function, a regex function, and a replace function. Once the data is cleaned, a word embedding and document embedding is learned for the words and for every observation's description. The document embedding is a description embedding.

These embeddings are learned by using Gensim's Doc2Vec function. The learning takes place over 20 iterations where during each iteration the learning rate is slightly decreased. The learning is done on a neural network where the network is trying to learn unique word embeddings. The networks tries to learn the words around a given word so then it can make a prediction of the next word in a sentence. The better it performs on this neural net, the better the word embeddings should be in theory. Once the word embeddings are learned, a document vector is constructed by the word embeddings present in that document. That document embedding is the feature for this case.

Implementation:

The implementation of each approach are contained within two functions. These functions can be viewed explicitly in their corresponding notebooks under: train and _train. Below, I will walk you through each function for each approach.

Approach I: Implementation

- **Train()**
  - creates a variable to hold when the function is initaited
  - reads data from csv
  - performs any needed cleaning of the data
  - prints ingestion time
  - rewrite variable to hold start time, to calculate training time
  - calls _train()
  - prints training time
  - returns a data frame with 10 most similar companies for each company
- **_Train()**
  - creates a data frame to hold output
  - converts data frame columns into arrays
  - stores a CountVectorizer object
  - learns the vocab then tokenized each description with simple frequency count
  - stores tokenized features
  - Then performs a nearest neighbor search for each company returning the 10 nearest neighbors based on Minkowski distance. Stores each of the ten neighbors in an array.
  - Then adds a row to the out data frame: names, 10 Nns, and features

Approach II: Implementation

- **Train()**
  - creates a variable to hold when the function is initaited
  - reads data from csv
  - prints ingestion time
  - rewrite variable to hold start time, to calculate training time
  - calls _train()
  - prints training time

    - returns a data frame with 10 most similar companies for each company
- **_Train()**
    - creates a data frame to hold output
    - converts data frame columns into arrays
    - stores a TfidfVectorizer object
    - learns the uni, bi, and trigrams, drops stopwords, and calculates TF-IDF score for every gram
    - creates a matrix that holds the cosine similarity between all companies using a linear kernel on the TF-IDF features
    - performs and argsort on each row of the matrix holding the cosine similarites to obtain the 10 closest. Stores the firm name and cosine similarity for the closet ten in an array. Adds a row to the output data frame: name and array mentioned in previous sentence.

Approach III: Implementation

- **Train()**
    - creates a variable to hold when the function is initiated
    - reads data from csv
    - performs some cleaning on the data
    - prints ingestion time
    - rewrite variable to hold start time, to calculate training time
    - calls _train()
    - prints training time
    - returns a data frame with 10 most similar companies for each company
- **_Train()**
    - creates a data frame to hold output
    - stores each word except stopwords in description into an array and stores company name for each word in a parallel array, and then puts both into an array.
    - Trains the Doc2Vec model over 20 iterations, each time dropping the the learning rate by a small amount
    - saves the Doc2Vec model (for future use)

- uses Doc2Vec's function most_similar to return the closest companies in cosine similarity on a description embedding. Stores the closest ten in an array with company name and difference in cosine similarity. Adds a row to the output data frame: name and array mentioned in previous sentence.

Refinement:

Because this problem is unsupervised, and the benchmark is just a simple method being applied, refinement is dealt with differently than it would be in a supervised case. In a supervised case, I could tune hyper-parameters to refine my model or play with feature extraction (choosing or changing features to improve the model).

Here, the refinement is seen in the differences between each approach. (Also, I could add even more refinements, but since approach II performs well, I decided it wasn't needed)

The first approach is naive in its feature extraction and in its distance measure for the nearest neighbor search. All it does is perform a frequency count on each word in the description and leaves zeros for words in the entire lexicon, but not in that particular description. Because of this both approach 2 and 3, attempt refinements in feature extraction. Approach 2 uses and TF-IDF weight instead of a simple frequency count for its tokenization of the words in description. Approach 3 uses a neural net to train word embeddings and description embeddings.

Also, both approach 2 and 3 make a refinement over approach 1's distance measure. Approach 2 and 3 use cosine similarity, which is touted as one of the best performing distance metrics for NLP [9].

Results:

Model Evaluation and Validation:

As discussed in the 'Metric' section, a quantitative metric for evaluation is hard to define. Instead a qualitative tool is used on the approaches' outputs to aid in judging performance. The clear winner out of the three approaches is approach 2: the content based recommendation system.

In the 'Justification' section below, I will show using the qualitative function that approach 2 outperforms approaches 1 and 3. I will also show it is trained the fastest. In this section, however, I will discuss approach 2 in detail and explain the sensitiveness analysis that can be found in the jupyter notebook called 'A2SensitivityAnalysis'.

Approach 2 is a content based recommendation engine. That means once all data is collected you can extract features and then store the cosine similarity between the feature space in a matrix. Any time time you want to search for a n-number of similar companies you can just argsort the matrix on the row

9 http://www-nlp.stanford.edu/IR-book/

that represents the company you are searching for similar companies to (you return the back end of the argsort).

To see how the method performs under different circumstances, I run the model under three separate scenarios:

1. Bag of Words instead of TF-IDF
2. A decreased data set: 3000 instead of 5000
3. Taking a subset of description to extract features (Basically, testing effect of smaller description space)

Bag of Words instead of TF-IDF:

Using BOW instead of TF-IDF reduces the quality of the competitors returned. By looking at the checker function in the Sensitivity Analysis notebook under the BOW section, we can see competitors that are definitely not competing in the same industry. For the Octagen firm, which is in bio-pharmaceuticals, we are given Intuit, HP, and a power systems company as competitors, when obviously they are not. One possible reason for this, is that those predicted competitors have larger descriptions than most of the firms in the data set. Because of this, they wind up having more words that match with smaller descriptions especially stop words and generic words. That is why TF-IDF is a superior feature extraction method for NLP data. It helps deal with the size and scale of the descriptions.

For the other two companies, we get similarly poor results. You can refer to the notebook for the function calls.

A decreased data set: 3000 instead of 5000

Decreasing the size of the data set, doesn’t appear to destroy the approaches results. Yantra was randomly dropped from the data, but the checker function on the other two seems to perform well. The competitors are very similar in scope and industry. See the exact results in the Sensitivity Analysis notebook.

Taking a subset of description to extract features

For this robustness check on approach 2, I only keep the first 15 words of each description. The idea is to see how the method performs with more less data and more sparse vectors. The results are good; the checker function returns firms that are in the same industry. The only downside is its hard to understand how close these firms are in exact product scope. The reduced descriptions are mainly just industry information. Therefore, more information definitely improves competitor search. To see exact checker function results, see the Sensitivity Analysis notebook.

Justification:

Approach II performs the best out of the three approaches. Below is a table that shows the time it takes for each model to be trained on the data and shows the top 5 competitors for each of of the approaches. By reading the descriptions of each of the returned competitors you can get a sense of how good the method performs. If you want to see even more competitors returned, go to the approaches respective jupyter notebook.

By exploring the table below, we notice that approach two provides the best similar companies out of the three methods. Approach one is not horrible as it does give some firms that are in the same industry and close in product, but also throws in firms that are not even close (i.e. hairdressing operator and data networking company are listed as similar to a bio-pharmacuetical). Approach 3, is just terrible; It barely returns any firm that looks like a competitor to the queried firm (My initial thought is there is not enough information to train the word embeddings well or I didn't train the embeddings enough).

	A2 Content Based Recommendation Engine	A1 Naive Nearest Neighbor Search	A3 Doc2Vec Similar Search
Training Time	13 seconds	17 seconds	57 seconds
Octagen	Operator of biopharmaceutical company. The company develops drugs for hemophilia, other genetic disorders and variations of recombinant B domain to avoid inactivation by flying below the radar screen of the immune system.		
Octagen Comp 1	Flex Pharma: Operator of a biopharmaceutical company. The company develops clinically proven products and treatments for muscle cramps and spasms.	Twinstrand Therapeutics: Operator of biopharmaceutical company. The company engages in the discovery, development and commercialization of biological drugs for the treatment of life threatening diseases.	Destination Kiruna: Operator of a tourism company in the region of Kiruna. The company organizes events, guided tours, self-drive tours and offers holiday and tourism packages.
Octagen Comp 2	Alkermes: Operator of a biopharmaceutical company. The company develops products based on drug-delivery technologies to enhance therapeutic outcomes in major diseases.	Axovan: Operator of a biopharmaceutical research company. The company is involved in the discovery of drugs linked to G protein-coupled receptors.	Periness: Operator of a biotechnology company focusing on addressing the problem of male infertility. The company's product is a systemic protein-based drug for treatment of male sub-fertility.
Octagen Comp 3	Twinstrand Therapeutics: Operator of biopharmaceutical company. The company engages in the discovery, development and commercialization of biological drugs for the treatment of life threatening diseases.	ZetaRx Biosciences: Operator of biotechnology company. The company engages in the development of therapies for cancer and other diseases using genetically engineered lymphocites.	Edge Therapeutics: Provider of therapeutic products for acute, fatal and debilitating medical conditions. The company develops implantable technology for direct delivery of therapeutic compounds to the site of brain injury.
Octagen Comp 4	Konova: Developer of anti-obesity drugs. The company develops drugs that are used for the treatment of obesity.	Frisørkjeden Tango Norge: Operator of a hairdressing chain. The company was established because of the growing demand for hair saloons.	BMDSys Production: Developer of cardiac diagnostic imaging systems. The company uses magnetic field imaging system in medical diagnosis. It also helps to identify persons with a risk for sudden cardiac problem by cardiac electrophysiology.
Octagen Comp 5	Ansata Therapeutics: Operator of a biopharmaceutical company focused on dermatologic treatments. The company develops peptide-based topical antibiotics for skin issues.	Loran International Technologies: Operator of a data networking company. The company is a system integrator of high speed networks.	Carolus Therapeutics: Developer of biopharmaceuticals aimed at treating acute and chronic inflammation. The company develops new drugs that are aimed to provide relief to patients suffering from a host of disorders triggered by acute and chronic inflammation.
Yantra	Provider of distributed order management and supply chain fulfillment solutions for retail, distribution, logistics, and manufacturing industries. The company focuses on distributed order management, supply collaboration, inventory synchronization, reverse logistics, logistics management, networked warehouse management, and delivery and service scheduling. It also provides consulting and support services. It offers Yantra 7x products, a comprehensive group of software applications, which enable organizations to manage their fulfillment processes across customers, operations, suppliers, and partners.		
Yantra Comp 1	Valdero: Provider of supply chain performance management and analytical collaboration software applications. The company provides inventory visibility and order fulfillment software applications.	Factory Logic: Supplier of software for lean scheduling and supply synchronization. The company provides manufacturing and supply chain management solutions to the automotive, electronics, industrial equipment and aerospace industries.	Hewlett-Packard: HP Inc, formerly Hewlett-Packard Company was incorporated in 1947 under the laws of the State of California as the successor to a partnership founded in 1939 by William R. Hewlett and David Packard. Effective in May 1998, it changed its state of incorporation from California to Delaware. The Company is a provider of products…
Yantra Comp 2	Sandlot: Provider of subscription management software and services for publishing industry. The company\'s product EclipseNet, manages order fulfillment processes within the virtual enterprise. The company provides a in-house and hosted services for management of recurring order transactions, integrated order process management, order fulfillment and customer service.	Valdero: Provider of supply chain performance management and analytical collaboration software applications. The company provides inventory visibility and order fulfillment software applications.	Angiotech Pharmaceuticals: Angiotech Pharmaceuticals Inc is a pharmaceutical and medical device company that develops, manufactures and markets medical device products and technologies, primarily within the areas of interventional oncology, wound closure and ophthalmology. The Company currently operates in two segments: Medical Device Technologies…
Yantra Comp 3	Factory Logic: Supplier of software for lean scheduling and supply synchronization. The company provides manufacturing and supply chain management solutions to the automotive, electronics, industrial equipment and aerospace industries.	GEOCOMtms: Provider of integrated fleet management software to local and short-haul pickup and delivery operations. The company also provides implementation, training and support services.	Phurnace Software: Provider of Java application deployment. The company\'s products include Phurnace Deliver, which eliminates configuration-related errors and reduces dependency on scripting. It's products also include Phurnace WebSphere Portal Deliver, …
Yantra Comp 4	Intelligent Markets: Provider of comprehensive order management software and services to broker dealers. The company offers ALTA, an order management system, which offers services in multiple asset classes, including convertible bonds, corporate bonds, OTC equities and derivatives.	Mercari Technologies: Provider of supply-chain management and e-fulfillment software company. The company\'s merchandising solutions bringing retailers and manufacturers closer to their customers by integrating consumer behavior data with merchandise planning, supply chain optimization, category management and financial analysis.	Vipshop: Provider of flash sales website. The company provides new sales events with foreign and domestic brand name clothing, accessories, cosmetics and designer household items at discounted prices in a limited time sales format.
Yantra Comp 5	Mercari Technologies: Provider of supply-chain management and e-fulfillment software company. The company\'s merchandising solutions bringing retailers and manufacturers closer to their customers by integrating consumer behavior data with merchandise planning, supply chain optimization, category management and financial analysis.	Aspective: Provider of e-commerce and WAP-based mobile applications and services for customer relationship management, business intelligence, data management and field service management. The company also provides managed and support services, which include application support, support desk, infrastructure, monitoring and outsourcing services.	Harman International Industries: Harman International Industries Inc was incorporated in 1980. The Company is engaged in the design and engineering of connected products and solutions for automakers, consumers and enterprises, including audio systems, visual products, enterprise automation solutions and connected services. Its manufacturing facilities are located in North America…
Discovery Engine	Developer of an internet search engine. The company offers an interaction model of search engine that also can also compile information from multiple sources.		
Discovery Engine Comp 1	Peryskop.pl: Provider of an online search engine. The company provides semantic search engine for products and product\'s reviews in Polish and English.	Zebido.com: Operator of an auction website.	Eldat Communication: Developer of Electronic Shelf Label (ESL) systems. The company develops electronic defense systems and integrated circuit designs.
Discovery Engine Comp 2	JustSpotted: Provider of real time search engine. The company\'s search engine aggregates and organizes content being shared on the internet. It offers search options on entertainment, technology, sports, world and business, science, gaming, politics and lifestyle topics.	ES Enterprise Solutions: Provider of an online software service.	Photonics Applications: Manufacturer of fiber-optic transmission technology for cable and wireless communications networks. The company also provide design, development and consulting services in fiber-optic transmission technology systems.
Discovery Engine Comp 3	Goshme Solucoes Para a Internet: Developer and provider of search engine. The company assists users by providing a list of all search engines and databases appropriate to the query, ranked by relevance, divided by categories and sub-categories, and with a brief description about each search engine.	Realtime Worlds: Developer of video games.	Genesis Teleserv: Provider of integrated customer contact services. The company also integrates information management systems.
Discovery Engine Comp 4	WiseNut: Provider of search engine and Web-browsing services. The company is the developer of a crawler-based search engine and database of indexed Web pages.	La La Media: Operator of an online music store.	Afferent Corporation: Developer of medical devices to treat chronic neurological dysfunction. The company\'s lead technology enhances the function of mechanoreceptor cells involved in sensory perception as a means of restoring brain function following stroke, improving elderly balance and addressing complications resulting from diabetic neuropathy.
Discovery Engine Comp 5	DealAngel: Provider of a travel deal search engine. The company enables users to find travel and hotel deals.	Peryskop.pl: Provider of an online search engine. The company provides semantic search engine for products and product\'s reviews in Polish and English.	Dispatching: Operator of a promotional logistics company. The company also offers its clients various field support with services like storage, handling and distribution. It also offers promotional services including brand development, sales promotion and market research.

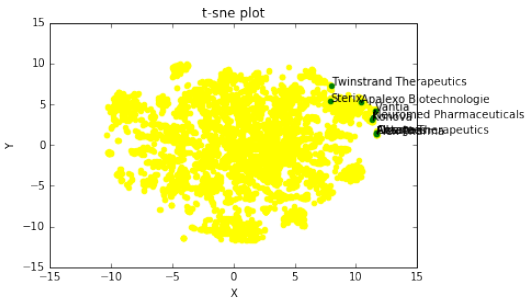
Conclusion:

Free Form Visualization:

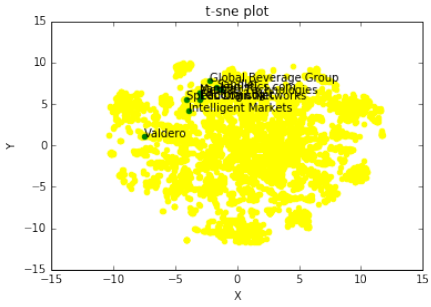
The essence of this problem is clustering. We ultimately just want to know the firms that are closest to a firm in question. Therefore, a visualization that makes sense is a 2 dimensional graph that plots the firms (hopefully, showing those closest as competitors). However, this is is a challenge. Approach 2 has a sparse feature space of 178894 (made up of uni, bi, and trigrams).

Therefore, to create a 2d visualization, we need to reduce the feature space drastically. PCA is terrible for NLP. An alternative that is very popular in NLP is t-SNE [10]. t-SNE is a tool for data visualization. It reduces the dimensionality of data to 2 or 3 dimensions so that it can be plotted easily.

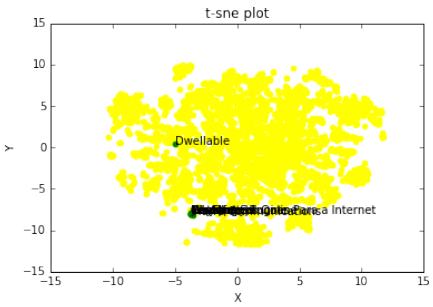
However, with very high dimensional sparse data it can be helpful to first reduce the dimensions (for this example , I reduce the dim to 50) with TruncatedSVD and then perform t-SNE. This will usually improve the visualization. Below, are three plots. One for each of the benchmarking firms: Octagen, Yantra, Discovery Engine.



Ocatgen Plot



Yantra Plot



Discovery Enigne Plot

The green dots are the queried and closest cosine similarity firms. The yellow dots are all other firms. Ideally, we would want the green dots to be the closest to each other and yellow outside of the green clustering. However, that does not exactly happen. The problem here is the information lost while reducing the feature space to 2.

It may be possible to get better looking 2 dimensional plots by playing with the dimension reduction tools and parameters, but because of the cost -its very time consuming to run- I will not try to get better looking plots. Also, the code for the dimensional reduction and plots is in jupyter notebook ‘A2ContentBasedRecommendationEngine’ under the free form visualization section.

10 http://alexanderfabisch.github.io/t-sne-in-scikit-learn.html

Reflection:

The overall aim of the project is to create a method to return competitor (similar) companies to a queried firm. For start-ups or young companies, competitor lists from online sites like Yahoo Finance or CrunchBase may be sparse or non existent at the time you want to know. Therefore, having a tool that can just used scraped descriptions of startups may be useful in quickly identifying the competitive landscape. The alternative is doing the research by hand.

For this project, I believe a content based recommender engine performs well.

1. It takes the data: firm name and firm descriptions
2. Converts the descriptions into TF-IDF vectors (creates uni, bi, and trigrams and drops stop words)
3. Calculates a cosine similarity matrix using a linear kernel
4. Returns top 10 similar companies by performing an argsort on the cosine similarity matrix (return back end of argsort)

Note: If I added more firms, I would need to recalculate the cosine similarity matrix

One thing struck me as interesting during the project.

The poor performance of Doc2Vec surprised me. I have never used it before, but have used the Word2Vec embeddings (which this is built on top of) before and have normally seen very strong results. This leads me to ruminate on two possible issues.

1. I just didn't have enough data to train the model or I under trained it- I am leaning towards this as the problem.
2. The document embedding loses valuable information. So the idea of Doc2Vec is it learns the word embeddings in some dimensional space and then takes all the words in a document and maps them to some embedding. You can think about it like a dimension reduction almost. For each document you have this large word embedding matrix and then you reduce this to some document embedding, in this case a description embedding. I guess its possible that the description embedding lost valuable information that leads to the poor results

Improvement:

Approach 2 uses TF-IDF for feature extraction, leverages phrases as well as uni-grams, drops stop words, and uses a linear kernel to obtain cosine similarity instead of using a nearest neighbor search. Possible improvements are:

- Append the keyword list as a feature: possibly adding the keywords to the description both in phrase form and as uni-grams, then apply TF-IDF.
- Add higher order grams, like a quadgram
- Use a different method for feature extraction: some word vectorization (like word2vec)
- Mine more data: could create a webscraper with Selenium to collect more information about the firms; one thought is use CrunchBase to get a list of competitors where available. Another thought is get articles about firms where available and use them as features as well.

In conclusion, I do believe an even better competitor tool can be created, but I am not sure if it could be created with just this data. I think Approach 2 does about as well as any approach can with the starting data.