Generating a Lexicon for Named Entity Co-occurrences

Kevin Jin
New York University
kj764@nyu.edu

Abstract

Methods commonly used for ranking bigram collocations can be adapted to rank pairs of named entities that are frequently found in the same sentences together over a corpus of news stories. By representing entities as nodes in a graph, the frequency of co-occurrences can be used to weigh the edges. After filtering out infrequent co-occurrences, methods from network analysis can cluster mutually co-occurring entities into "sectors". This paper explores the significance of using such a method by validating the detected sectors using context and intuition.

1 Introduction

This paper describes the initial stages of a lexicon project that seeks to map related organizations to each other. The purpose of this stage is to test the assumption that the most related pairs of organizations are those that are most frequently mentioned together in news articles. If the resulting clusters generated from this corpus are mostly intuitive, then the process may be significant and any results generated on a set of news stories can potentially be of interest and worthy of further exploration by domain experts.

A real-time implementation can be a source of additional insight for economists. For example, being able to generalize a group of companies as a cluster and to find the company that has the highest "centrality" in that cluster may enable institutions to focus on only one company as representative for the entire cluster. Additionally, policy-makers may find it useful to observe trends in supplier relationships that result from advancements in technology. Lawmakers may also want to examine mutual dependence to forecast the potential systematic impact and loss of jobs that one company's collapse may bring. Financial investors and traders may be interested in quantifying the connections between companies because of the potential for news stories from related companies predicting the near-future financial performance of a company of interest.

2 Process overview

Many of the natural language processing problems that compose this project have already been satisfactorily solved. This is a brief description of all the moving parts, in the order they are used in the procedure:

(1) Named entities tagging, and anaphor resolution to substitute pronouns with antecedents: BBN Technologies provides a proprietary annotation for the Penn Treebank corpus of Wall Street Journal texts and some anaphor resolution.

(2) Co-reference resolution (e.g. abbreviations, full legal names) to substitute all aliases for the same entity with a single name: This project includes its own routines to resolve these aliases.

(3) Co-occurrence: in the context of this project, two names co-occur if they appear together in the same sentence. It does not matter whether one entity acts on the other or both entities are agents or patients, because any scenario would suggest that the entities are both related[1] in some way. Normalized Pointwise Mutual Information (Bouma 2009) is used for scoring frequency.

(4) Community detection: create a graph with entities as nodes and co-occurrences as edges (weighted by the NPMI score). Then filter out insignificant co-occurrences. Clusters in this filtered network are detected with the Louvain method (Blondel et al. 2008).

3 Corpus preparation

To fix non well-formed XML in the BBN corpus, I wrote a program that runs the XML parser on a malformed file, automatically corrects the first error reported, and repeats until no errors

[1] As competitors, suppliers, customers, etc.

remain. This was successful because all the errors had similar and unambiguous resolutions.

There are also instances where tokens in BBN's anaphor co-reference file do not match the tokens at the offsets in their annotated corpus. There is no discernable pattern that can be used to correct for these errors, so the co-reference entries whose substrings in the corpus do not exactly match the entity name in the entry are simply ignored.

4 Co-reference resolution methodology

There are two routines performed to determine whether two names are aliases of each other:

(1) Trim both names by removing common suffixes (e.g. "Corp.", "PLC", "S.p.A."), then common prefixes (e.g. "The"), then common punctuation (e.g. commas, hyphens, apostrophes, slashes, and ampersands). If both strings do not match (case-insensitively), then on the longer of the two names (i.e. the "full name"), try deleting 0 tokens from the front and 1 token from the back of its trimmed name, then 1 from the front 0 from the back, and finally 1 from the front and 1 from the back. The two names are aliases if any of these substrings of the trimmed names match.[2]

(2) If none of the substrings matched, the program tries skipping letters within the middle tokens of both the trimmed and untrimmed names. The routine iterates on the shorter of the two names (i.e. the "abbreviation"). It starts at the first character of the abbreviation and the first character of the first token in the full name. On each iteration, the routine has two options: (a) continue to the next character in the current token in the full name, or (b) continue to the first character of the next token in the full name. If the current character in the abbreviation matches neither the character in (a) nor (b), then the routine fails. When the last letter of the abbreviation is matched, the routine succeeds and the two names are deemed aliases.[3]

[2] The number of deletions becomes more aggressive as substrings are matched. E.g. when "Lipper Analytical Services" is matched with "Lipper Analytical", it can then match "Lipper" because it is allowed to delete one more than needed for "Lipper Analytical".

[3] To avoid false positives, routine (2) uses additional heuristics to determine whether options (a) or (b) are safe. It also uses additional options similar to (b) that skip to next uppercase letters rather than next tokens.

4.1 Co-reference resolution evaluation

Generally, this program performs well. Simple deletions such as "& Co." in legal names and acronym recognition do most of the work. Portmanteaus such as *Amex* (for *American Express*) are matched as well. The task may be harder for more unstructured corpuses, but the program's rules work well presumably because journalists are required to follow strict style guidelines.

The biggest improvement may result from limiting which existing names can be compared to an unrecognized name. For simplicity, the program currently compares new names to all previously found names in the corpus. The number of false positives can be decreased by comparing only the existing names in the document (and merging all document-specific names in the end) because it provides the program contextual clues. For example, it is unlikely that *MCI Communications*, a telecomm company, would be mentioned in the same document as *Mitsui, C. Itoh & Co.*[4], a Japanese conglomerate. This would thus eliminate the false positive that the program currently generates, and also decrease run-time substantially.

Exhaustive searches may also improve results. Currently the program stops searching for matches as soon as one is found. Routine (2) frequently is too aggressive, leading to many false positives. If multiple existing names can match to the new name, matches made by routine (2) should be least preferred, followed by matches made by routine (1) with the most deletions, followed by matches made by routine (1) with the fewest deletions. However, such a system increases run-time substantially.

Possessives were also problematic because BBN's named entity annotations always included a following "'s" token if such a token exists. In the baseline system, simply discarding apostrophes leads the program to mistakenly classify "Warner-Lambert 's" as an alias of "Warners" because it recognizes "WarnerLambert s" as a portmanteau of "Warners" by routine (2). However, an alternative system that treats the whole token "'s" as a suffix generates false negatives such as where "Lloyd 's" is mistakenly distinguished from "Lloyd 's Bank". In the baseline system, "Lloyd 's Bank" matches "Lloyd 's" after one back token deletion in routine (1). In the alternative system, an attempt to match by routine

[4] Which itself is an error by BBN's name entity parser. *Mitsui* and *C. Itoh & Co.* should be two entities.

(1) fails because “Lloyd 's Bank” requires two deletions to reach “Lloyd”. There are a number of workarounds that fix the alternative system and can be tested in the future.

Also, there are very few times that a front delete, as opposed to a back delete, is needed in routine (1). It may improve accuracy if front deletes are eliminated entirely. One mistake the program made was matching “Time Warner” with “Warner Bros.” by interpreting “Bros.” as a suffix and front deleting “Time” from the full name. Similar problems are exhibited between “Times-Stock Exchange” and “Exchange”, between “Toronto Stock Exchange” and “Stock Exchange”, between “Northeast Utilities” and “Utilities”, and between “News Corp.” and “Detroit News”.

5 Pair analysis methodology

The previous programs produced a list of co-occurring entities in each sentence. This program aggregates the entity pairs found in the sentences. It organizes a list of annotated sentences into a list of entity pairs scored by frequencies.

It is important to first note that for a sentence that has n distinct entities, there are $n * (n-1)/2$ symmetric pairs of entities. This means that a sentence with 4 co-occurring entities will have $\frac{4*3}{2} = 6$ pairwise relationships.

The corpus-wide frequencies[5] of each distinct pairwise relationship and the corpus-wide frequencies of each entity in the pair are counted as $f(A,B)$, $f(A)$, and $f(B)$ respectively. The total number of mentions in the entire corpus, N, is also counted. I then define a probability measure $P(A \cap B) = \frac{f(A,B)}{N}, P(A) = \frac{f(A)}{N}, P(B) = \frac{f(B)}{N}$. The Pointwise Mutual Information (PMI) score is calculated using the formula $\log_2 \frac{P(A \cap B)}{P(A)P(B)}$, and subsequently normalized to the NPMI score by dividing PMI by the factor $-\log_2 P(A \cap B)$, as described by Bouma.

NPMI scores are distributed in the range $[-1.0, 1.0]$, where 1.0 indicates that both entities are always found together, -1.0 indicates both entities are never found in a sentence together, and 0 indicates statistical independence, i.e. $P(A \cap B) = P(A)P(B) \Leftrightarrow P(A \mid B) = P(A)$.[6]

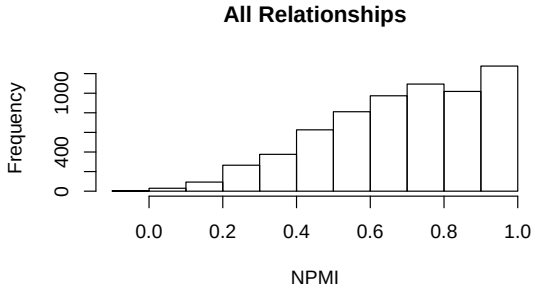
[5] Here, frequency is the number of sentences that an entity or an entity pair is found in in the corpus.

[6] NPMI is positive when $P(A \cap B) > P(A)P(B)$ and negative when $P(A \cap B) < P(A)P(B), 0 \leq P(\cdot) \leq 1$.

5.1 Pair analysis evaluation

To detect irregularities early on, I performed a quick verification of these NPMI scores before preparing the network analysis. One of my immediate observations was that there are only five relationships with negative NPMI.

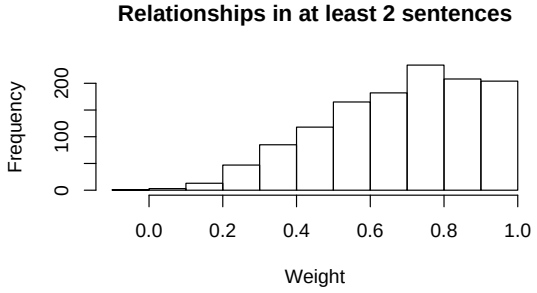
The distribution of NPMI scores is indeed negative skewed towards strong relationships:



Furthermore, pairs of entities are also very rarely found together in multiple sentences:

# Sentences Found In	*# Relationships*
1	5308
2	739
3	227
4	114
5	38
6	36
7	22
8+	84

Perfect relationships are the most common relationship strength, and most relationships occur only once or twice in the corpus. This suggests that most of the perfect relationships are a result of both entities being mentioned only once in the corpus, and in the same sentence. Relationships such as these, though they may be strong in reality, are not very significant from an analytical standpoint. Therefore, only relationships that exist in at least two sentences will be passed to the network analysis.



With this filter, perfect relationships are much less common. The ones that remain are more significant than the ones that were removed. Co-occurrences found multiple times are more likely to reflect a real-world relationship, while co-occurrences that are found only once in a corpus can exist only by pure chance.

One improvement I would make is to weight the NPMI scores by the number of times both entities appear in the corpus so that these relationships do not have to be deleted entirely.

6 Network analysis methodology

This program extensively uses algorithm implementations from the Gephi Toolkit, an application programming interface that exposes the features available in the graphical user interface of the graph visualization software, Gephi.

The cluster analysis is not very sensitive to NPMI scores, so retaining weak edges leads to poor quality results. For this reason, the following types of relationships were first filtered out:

(1) Relationships found in only one sentence. The reasoning is explained in the last section.

(2) Relationships with an NPMI score of less than 0.5. Organizations such as research groups, ratings agencies, and stock exchanges are indiscriminately connected to all kinds of other organizations, so relationships with them are meaningless. An NPMI score cut-off is effective because such co-occurrences do not tend to repeat.

(3) Connected components[7] of size 2. Such components consist of only two entities and one distinct relationship. These components clutter the graph while providing little insight.

After the weak edges are deleted, any vertex that is isolated[8] as a result of losing all its incident edges is deleted from the graph as well.

Clusters are detected with the Louvain method for modularity maximization. The Louvain method, although non-deterministic, empirically converges in linearithmic time with respect to the number of nodes. Its output has been described as being as high quality as other algorithms.

Next, the nodes are laid out to distinguish the clusters in the visualizations. The first algorithm used is ForceAtlas2 (Jacomy et al. 2014). ForceAtlas2 attracts nodes that are mutually connected while repelling nodes that are not. Because this has the effect of repelling nodes away from the center until nodes are fully drawn into their clusters, a second algorithm, known as Fruchterman-Reingold (Fruchterman et al. 1991), is used after ForceAtlas2 terminates to attract all nodes to the center of the graph. Used in tandem, the generated graphic will not be so sparse that magnification is needed to see individual nodes, yet clusters will still be readily identifiable.

6.1 Network analysis evaluation

It appears as though the clustering algorithm is hitting what is known as a resolution limit (Fortunato et al. 2007, Lambiotte et al. 2009, Good et al. 2010). For example: because *The New York Times Co.* is a cut-point that bridges *McCall's* with *Washington Post Co.*, the *NYT* connects newspaper companies to a cluster of women's magazines that also includes *Michaels Stores Inc*. From looking at the graph, a human would likely say that *McCall's* "should be" in a cluster distinct from that of the newspapers because the *NYT* is a cut-point. However, because these two clusters are small relative to the size of the graph, the modularity maximization algorithm merges them. As a result of this problem with resolving small clusters in large connected components, it appears as though clusters that are coextensive with a small component in the graph may be the strongest indicators of mutual relationships.

7 Conclusion

Some clusters in the graph exhibit relatively high connectivity while others have a readily identifiable central node. The graph certainly exhibits an interesting structure that is worthy of further exploration. Because the Penn Treebank corpus consists of Wall Street Journal texts from the late 1980s, many of these entities have already experienced events that may stem from relationships with other entities. This enables one to test the existence of any forecasting power of this graph.

An interesting extension would be to implement semantic role labeling so that the exact relationship between entities in a sentence can be understood. For example, understanding the role that one entity has on the other in a sentence can enable one to classify whether the two companies are competitors, suppliers, or other.

TO BE CONTINUED

Reference

Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. 2010. The performance of

[7] In the context of graph theory, a connected component is a subgraph in which all nodes can be reached from all other nodes through some series of edges.

[8] I.e. the node's degree is 0. This happens when all relationships to other entities are weak.

modularity maximization in practical contexts. In *Phys. Rev. E* 81, 046106.

Gerlof Bouma. 2009. Normalized (Pointwise) Mutual Information in Collocation Extraction. In *Proc. GSCL Conf. 2009*, pages 31–40.

Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, Mathieu Bastian. 2014. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. In *PloS one*, 9(6), e98679.

Om P. Damani. 2013. Improving Pointwise Mutual Information (PMI) by Incorporating Significant Co-occurrence. In *CoRR*, abs/1307.0596.

Renaud Lambiotte, Jean-Charles Delvenne, and Mauricio Barahona. 2009. Laplacian Dynamics and Multiscale Modular Structure in Networks. In *arXiv*:0812.1770.

Santo Fortunato and Marc Barthélemy. 2006. Resolution limit in community detection. In *Proc. Natl. Acad. Sci. USA* 104, 36.

Thomas M. J. Fruchterman, Edward M. Reingold. 1991. Graph Drawing by Force-Directed Placement. In *Software: Practice & Experience*, 21 (11)

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. In *J. Stat. Mech*. P10008.