Tracking information propagation across media sources to analyze news events by modelling implicit content networks

Student: Anirudh Joshi
Supervisor: Professor Richard Sinnott
Research project: 75 Points
MSc. Computer Science: COMP60002

Declaration

I certify that:

- This thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- Where necessary I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department.
- The thesis is 16,450 words in length (excluding text in images, table, bibliographies and appendices).

Acknowledgements

I'd like to thank Prof. Richard Sinnott for giving me the freedom to pursue this research project, my family for supporting me through it, and my friends who helped review this thesis.

Abstract

The Internet has become the premier news source for hundreds of millions of people all around the world, and in turn, the news it reports has become a critical component in important decisions made every single day. As its importance has grown, the Internet has become a target for manipulation by numerous entities, such as corporations, PR agencies, advertising companies, governments and even news organizations themselves. Coupled with the sheer scale the Internet operates on, and the limited time normal people have to process the news that's reported to them, it becomes extremely difficult to see how any piece of content fits into the grander media landscape, where it is sourced from, or how its reporting differs from others.

Current strategies used to analyze the media landscape have numerous limitations. Aggregators simply cluster together documents and provide frequency analytics, but provide no insight into how news sources are interrelated, or how their reporting differs, expecting the user read every document. Search engines operate in a similar fashion, instead clustering documents by a query, rather than related content through time. Link based information tracking systems lack the ability to analyze how content changes over time. Exact or near-exact content systems that track quotes and exact matches fail to capture connections that exist at a paragraph or sentence level, and like the link based approaches, are far too coarse to truly inform users.

To overcome the problems presented, we propose a proof of concept solution which utilizes continuously trained paragraph vectors at the paragraph and sentence levels on news events provided by existing news clustering systems. In doing so we show that we are able to generate comprehensive semantic content indices and graphs that can be used as higher level abstractions to analyze news events. Using these abstractions we are able to interrelate the reporting of news articles across articles, news sources and through time, in addition to aiding differential analysis between articles at a content level.

To validate our proof of concept we use standard semantic measures against reference paragraph vector models. We then illustrate the applicability of our abstractions through a series of demonstration applications that give users insight into the reporting behind the news events they read every day.

Keywords: Information Retrieval, Internet News, Information Diffusion, News Networks, Machine Learning, Neural Networks, Semantic Content

Table of Contents

List of Figures

List of Tables

1. Introduction

The Internet's importance in the global media landscape has become central in recent years. It has become a critical component of political and advertising campaigns, as well as government sponsored propaganda operations. Internet news reporting is often the first and last source that millions of people use to read about an event, and in turn, it shapes the reality of what they believe to be true. Given its mind boggling scale and the limited time most people have to read, it is effectively impossible for news consumers to understand how every piece of content they consume relates to each other without deep in-depth research. The sheer scale and relentless quantity of news makes it difficult for consumers to make sense of how what they read arrives on their screens, whose hands it has passed through, and what differences occur between news sources without essentially reading it all (**Figure 1.1**).

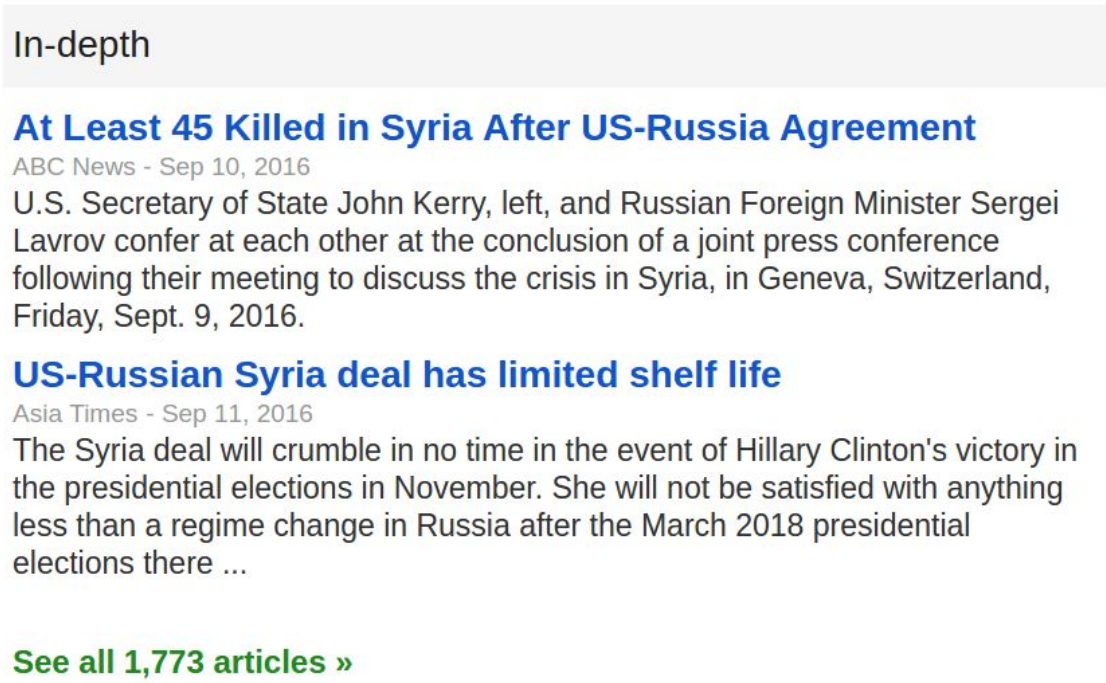

Figure 1.1: People are simply unable to consume all of the information reported on in an event with thousands of articles. This makes it extremely difficult to compare and contrast them, or to even see relationships between news sources [29]. This is a problem our proof of concept aims to address.

Google News, and other aggregation services such as Event Registry, report thousands of news events each day in an attempt to help users deal with the

information deluge [35] [36]. Academic systems have demonstrated methods of tracking information propagation across online media by essentially monitoring conserved content markers on parts of the article such as links, quotes, or relational predicates [1] [25] [16] [23] [9] [24]. Both massive event aggregation systems, and their academic counterparts, only use parts of the content graph to cluster and organize articles together, forcing users to read them all for actual analysis (**Figure 1.2**). This is a gap that our proof of concept system aims to fill.

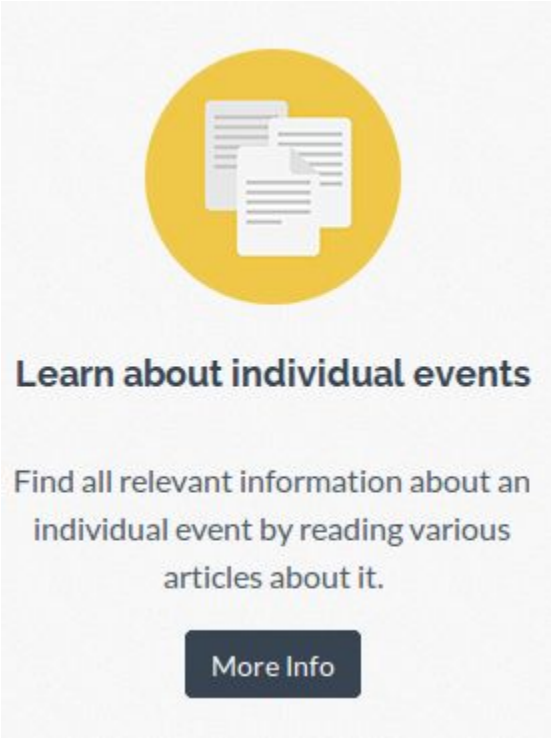

Figure 1.2: Existing state of the art clustering systems force users to read all of the content within an event to extract and compare all the relevant information [36].

Before we explore the motivation behind news event analysis, a few clarifying definitions are in order. The online news media landscape can be broadly split up into mainstream news (e.g. online news articles from professional news sources) and user-generated content (e.g. microblogs/Twitter/social media posts). This research project will be focusing on mainstream news as we find that it is usually the original source for the vast majority of media referenced by others, and often the final source of truth. We define a news event to be any series of news articles that are clustered together due to proximity in document content and time, regardless of the underlying method.

The goal of our proof of concept is to augment the abilities of readers and researchers, not by generating an exhaustive content graph like other approaches, but by facilitating the generation of specific views on the content graph. We achieve

this by using a critical simplifying assumption to cap computational costs, namely that we only process existing news event clusters, rather than operate on a stream of independent articles. We find that it's much better to leverage existing state of the art news clustering systems to increase the reliability of cross article semantic connections than it is to operate on every article in isolation. This caps the complexity of our graphs and boosts the power of semantic models by ensuring that the articles they organize are already found to be highly related, drastically reducing the number of false positives. This project will focus on tracking current events, rather than retrospective tracking, as the vast majority of events are often surfaced and consumed within a day or so of their existence, and as such, this is the time when they will have the most impact (**Figure 1.3**).

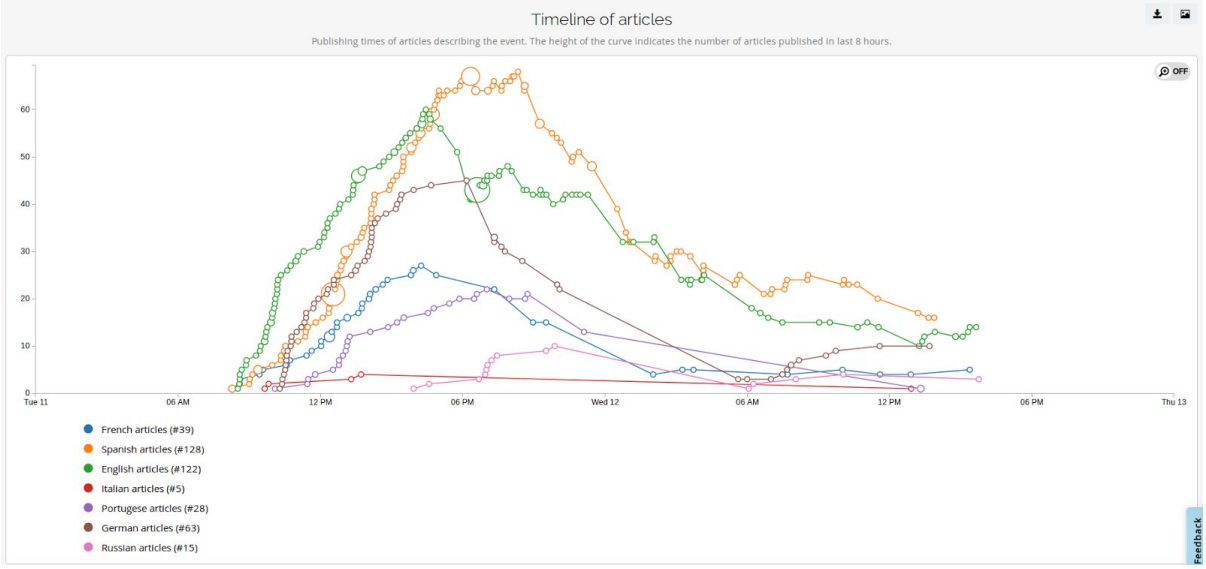

Figure 1.3: We constrain our analysis to the reporting of clusters of news events, which usually only occur over a single day, capping our computational costs. This image is frequency graph of news reports on a WHO recommendation event from Event Registry [30] [36].

Now we will move onto some of the motivating real world examples that illustrate why understanding and analyzing news events is important. Many news sources simply parrot what others have already written, but write it slightly differently. This may give a false sense of consensus. This fact can be most prominently seen when major news sources report inaccurate information when they rely on a single unreliable source for their information on an event. This was most prominently seen in the reporting of stories that originated from *The New York Times* (NYT) about Iraq in the lead-up to the Iraq war [37]. For those reading the news, it may have appeared, at the time, that there was a consensus on the facts on the ground in Iraq regarding their capacity with Weapons of Mass Destruction. Without the tools required to do a comparative source analysis on the reports, it appeared that there

was lots of consensus about the situation in Iraq, when in fact there was none. This occurred because the claims published by a single anonymous source from a single, yet reputable, newspaper (NYT) were seemingly backed up by reporting from thousands of others. However these other reports were in fact merely parroting the same false information. The cost of this misunderstanding in the consensus about Iraq's capabilities during that time lead to the pursuit of a highly expensive war. The issues with false perception do not merely stop with news organizations, with calls recently from world leaders like Angela Merkel warning that "search engines are 'distorting perception'" by controlling what is and is not seen by users through personalization algorithms; operating analogously to what is and is not reported by news organizations [41].

Another example as to why differential analysis of the content graph between news sources can be seen in the works of well known comedic programs such as *The Last Week Tonight* and *The Daily Show*. As perhaps the most pathological consumers of the news, their researchers and writers see news at a level we wish to produce automatically for consumers using our system. By viewing such a vast quantity of news media, they can see the interconnections between news sources automatically; the parroted lines, the duplicated content, and most important of all, with enough consumption, they can do a rapid differential analysis. For example comparing and contrasting how *Fox News* and *MSNBC* report on the same event. By providing users with this global insight through our proof of concept system, we hope to reduce the effect of the filter bubbles that surround us (**Figure 1.4**).

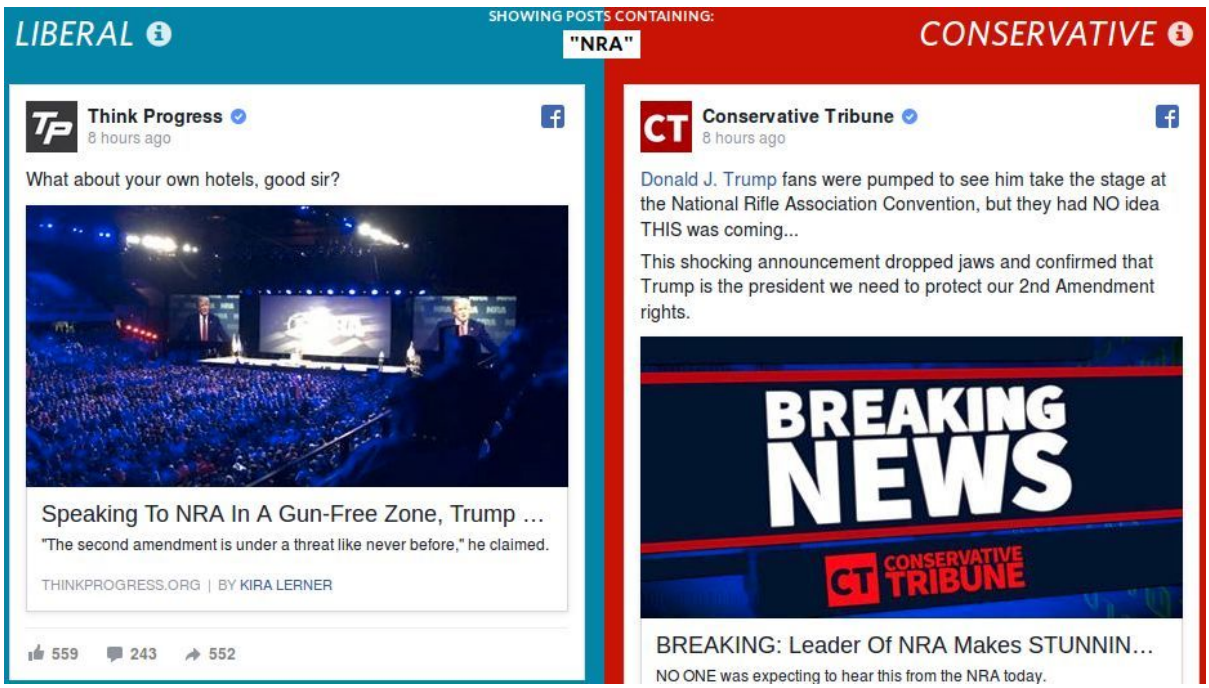

Figure 1.4: A demonstration of “filter” bubbles in a differential Facebook feed application created by the *Wall Street Journal* (WSJ) [38]. It demonstrates how depending on who reports what you read, and how they report it, you may come away with radically different interpretations of the same event.

Our motivation examples extends to active political warfare that states commit against each other through the manipulation of the news. There have been numerous investigations recently of Russia’s utilization of troll armies [39], state sponsored media and third party organizations to meddle with US elections [40]. Utilizing our system, we will be able to observe these types of operations by monitoring news source relationships and performing differential content analysis.

We now move onto some of the different approaches that are currently used to make sense of the news events seen every day. There are many different ways to consume the news, such as checking news aggregators or subscribing to RSS and Twitter feeds. We will focus on some of the largest aggregators, such as Google News and Event Registry, who aggregate millions of articles per day using a variety of algorithms to produce clustered events as you can see in **Figure 1.5**.

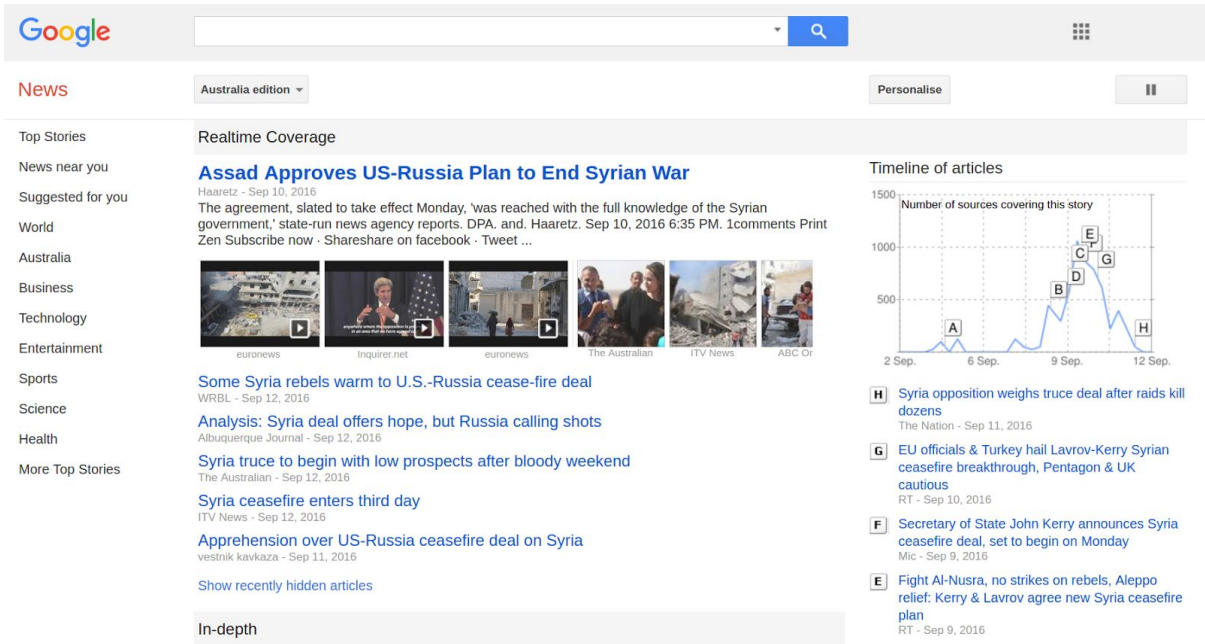

Figure 1.5: Most aggregation systems, like Google News, simply list top articles with frequency counts [35]. However, there can often be 10s to 1000s of articles concerning an event, most with highly duplicated information mixed in with novel insights. Our goal is to be able to tease out the interactions between news sources by utilizing content similarity using paragraph vector models.

The issue with most existing systems is that they stop at document level clustering and frequency analysis, expecting a user to consume and understand how the content within the event cluster relates, how it is sourced, and the order in which it was published. Using aggregators such as these require people to delve deep into the content, consume it all and then make judgments as to the veracity, sourcing, and context of the claims. Aggregators expect users to read and understand everything themselves, merely clustering together documents but giving no sense of how content itself is interrelated. Academic systems too seem to exhibit this problem, often focusing on the complete content graph, at the expense of aiding individual news event analysis (**Figure 1.6**). We will explore such academic tracking systems further in the Literature Review (Section 2).

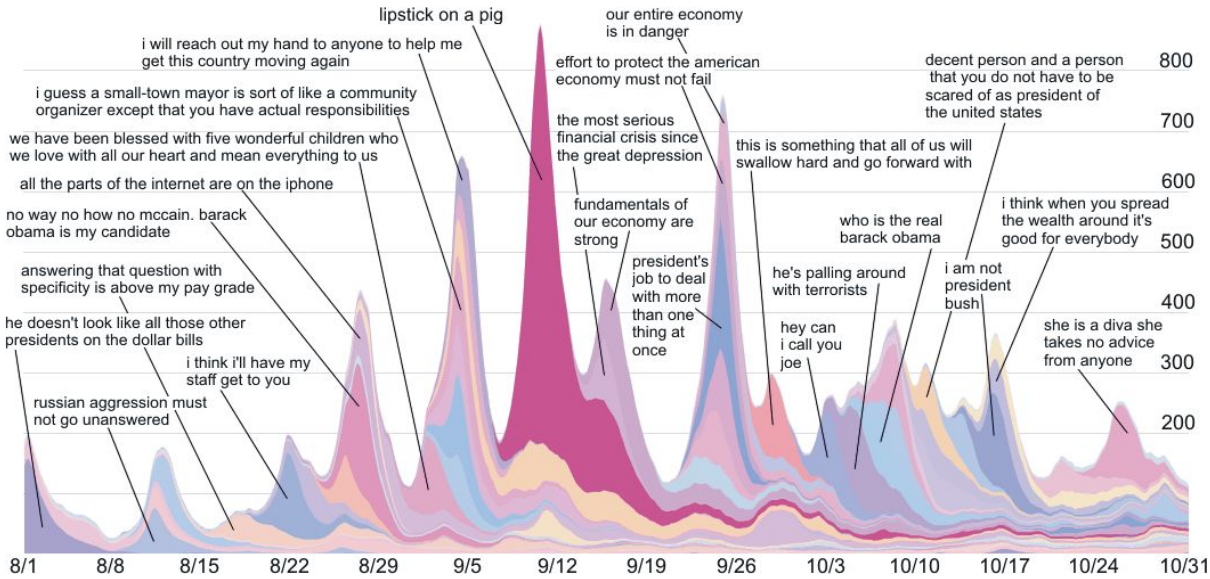

Figure 1.6: Academic systems are often concerned more about aggregate dynamics and frequency counts than they are in aiding news event analysis, a problem we wish to address. This image is from Leskovec et al. (2009) which demonstrates that observing global counts does not aid in actually understanding news events [16].

To address these issues we propose a system built on entirely unsupervised paragraph vector models. We use them to generate vector indices for all the paragraphs and sentences within a news event. We train these models by utilizing just the news data itself. We will then use standard clustering and graphing workflows to generate views on the content graph that will power demonstration applications to aid analysis.

We show that these applications can help users analyze, at a paragraph and sentence level, how information propagates across sources, compare news reporting between sources, and observe news source interrelationships at an aggregate level.

We verify the power of our entirely unsupervised self-training system against a set of reference models on a series of standardized semantic evaluation tasks. With these results in tow, we hope to demonstrate the next logical step in implicit content network modeling to allow users global insight into the news events they read every day.

2. Literature Review

There are numerous approaches to analyzing information propagation in online news events. They can be split into two overarching methodologies, one being the link diffusion method and the other being the content diffusion method. During our review, we found that the vast majority of systems only focus on a single or small aspect of the entire article, namely links or highly conserved content. To the best of our knowledge, we found that there is no significant research into utilising unsupervised systems to rapidly organize, at a paragraph and sentence level, the mass of semantic interrelations found within a news event. These are issues our approach attempts to address. We discuss some of the approaches we discovered during our research and address their contributions and shortcomings below. We also discuss which core ideas we built upon in the process of building our own workflow.

2.1 Link Diffusion

Link diffusion systems use explicit linking between articles to create a model of how information propagates over time. These approaches simply look at each link as a pointer to generate a propagation graph, with historical correlation indicating a direction of flow.

One of the foundational papers on link-based information propagation is that of Adar and Adamic (2005) who analyzed the link structure of the blogosphere to track information epidemics [1]. However, as there are numerous cycles, and no indication of direction when simply detecting a link between sources, they used heuristics to resolve directionality. To address these ambiguities they used an analogy to epidemiological models, which let them generate a directed acyclic graph which could be ordered by time to illustrate the flow of information. The key insight into resolving the ambiguities was noticing that explicit flows in the previous history between two nodes would give us a probabilistic direction. They assumed that information propagation took a viral form, and hence to find the flow path of the virus, one must simply look at the correlations between the infections of patients, or in this case, blogs.

The issue with link based systems is that they can only use links between articles, and not the articles themselves. In the case of long form unstructured content, like those found in news articles which rarely, if ever, link directly to their source, this is a big issue. Hence link-based information propagation systems are severely limited to only a small subsection of the total content graph (links). These issues are tackled by content diffusion tracking systems which we discuss below.

2.2 Content Diffusion

To address the issues that arise from simple link-based systems, the next generation information tracking systems focused on monitoring the propagation of conserved content (repeated strings or quotes) contained within articles. We explore these approaches below.

Yang and Leskovec (2010) utilized a different approach from pure network tracking by doing away with network topology altogether [25]. They focused on conserved content such as short textual phrases (e.g. quotes) or Twitter hashtags, and used their appearance at various nodes over time to build up a global linear influence model for each node. They tracked this by making the assumption that by just knowing when a node mentions a piece of conserved content and the time that it occurred, they would be able to generate how influential it was to any subsequent infection. This allowed them to build up an implicit global influence model based upon how influential a node was at causing the infection of other nodes. They generated this model of influence quickly by using a least squares like solution. However, by doing away with the network topology, it is difficult to make definitive statements about node connections, which the following systems attempt to address.

The MemeTracker system by Leskovec et al. (2009) was an extension of the link based structure to quotes, which can analogously be seen as links which mutate very little from source to source [16]. Thus they can be used as pointers to infer connections between news sources over time. They tracked short textual phrases, called “memes”, that were enclosed in quotes. They tracked how these memes rose and fell over time. They isolated them by generating an acyclic graph of similar quotes (quotes that were substrings of other quotes), whose root node best represented the full content of that chain of quotations (longest quote in chain).

They achieved this by identifying what they call phrase clusters, that is groups of strings that exhibit substantial textual similarity. They built the phrase cluster graph by observing that many mutational variations of phrases involve subsequences, or cuts, such that two phrases can be classified as similar if one is a continuous

subsequence of another. They connected phrases by using the word edit distances between them. If one phrase was a small series of edits away from another (e.g. a 1 word edit away), a connection would be made. By making links from smaller to larger phrases they generated an acyclic graph. They then weighted these connections by the size of the edit distance, decreasing the weight the greater the distance, and increasing it if the longer phrase occurred frequently. As partitioning this graph is a NP-complete problem, they utilized heuristics. To partition the graph they simply deleted low weight connections until the phrase cluster graph fell apart into a series of disjoint mutational quote chains, with shorter quotes feeding into a single long quotation. This was achieved by proceeding up from the longest quotations and assigning shorter text content to the clusters they are most connected to in a greedy fashion. This leads to quotes being assigned to phrase clusters to which they are most strongly connected (**Figure 2.1**). Despite this, the system is extremely computationally expensive. As one can see, when tracking all the content interconnections between news sources at a global level, it rapidly becomes extremely difficult to process. This is why in our system we focus on reducing the scope to analyse active news events, capping our complexity and boosting semantic performance, whilst still producing useful analysis. However we heavily use the idea that similar content should be connected into graphs by content similarity to better understand content flow.

Figure 2.1: Variants of a Sarah Palin quote from the MemeTracker paper by Leskovec et al. (2009) [16]. Shorter phrases are connected to longer phrases to which they are most strongly connected, leading to the falling apart of the graph into mutational quote chains.

The successor to MemeTracker, dubbed NIFTY by Suen et al. (2013), attempts to mitigate the extreme computational costs of tracking all the incrementally changing quotations by utilizing an incremental meme-clustering algorithm [23]. It uses the same underlying insight of connecting each quotation to a longer quotation it is a close subsequence of as part of a large directed acyclic graph, with long strings being the cluster assignments and the longest versions of the quotations. To improve performance they attempt to keep the graph size essentially constant, by freezing old clusters that haven't changed for quite some time, and constructing new ones when they don't fit into the existing clusters. They also find that through the use of aggressive filtering to remove documents that are duplicated, or contain phrases that are infrequent or from non-target languages, they achieve a significant boost in performance. This alludes to a critical aspect of analyzing news events online, as it is essential to reduce and preprocess the masses of exactly duplicated content if useful abstractions are to be extracted quickly and efficiently. We use this insight aggressively within our system to boost performance and extract insight by focusing exclusively on unique, but related content.

To generate the connections between two textual phrases, they trained a decision tree on Levenshtein distance hyperparameters for connections from a manually annotated phrase graph, and used that as a heuristic to generate future connections. Finally to speed up the generation of the graph they utilized Locality Sensitive Hashing to cluster together items that exhibit strong similarity. Through the use of shingling and min-hashing (essentially fast content similarity measures) they quickly reduce the text variants that need to be compared by grouping them. They then assign weights to these connections using both the edit distance and the relative time between the two phrases. A key insight was the utilization of mini-batches and the reuse of existing partitioned phrase graphs. In doing so they are able to add new phrases dynamically and freeze out old clusters, without having to recalculate the entire graph. Thus by only focusing on phrases whose edges are new, and by removing clusters that haven't exhibited significant new connections for a period, they can essentially keep the cost of processing millions of documents to basically constant time.

The goal of both of the above tracking systems was to track how "memes" spread rapidly throughout online media, and in turn, lead to the observations of short-term events. However, these abstractions do not allow differential content analysis. Another significant failing of this system is that, just like the link based systems, it only relies on a portion of the content served by news organizations about an event, namely things enclosed in quotes. Its use was also more trained towards analyzing event bursts, rather than observing propagation between sources or comparative analysis. These are issues we wish to address in our system.

In Colavizza et al. (2015) they tried a different attack to analyse information propagation by focusing on text reuse in early modern newspapers [9]. Their insight was understanding that barring explicit links, the vast majority of early news paper information flows would essentially be entirely conserved content, in the form of copied text reporting facts between papers over time. They analyzed gazettes from the year 1648 using OCR, string kernels and local text alignment to track how information flowed in early Italian newspapers. Through this method they were able to analyze the relationships and differential reporting between news sources about very similar events by observing the propagation of news during an event. This approach builds on ideas previously seen, and we utilize the ideas of tracking whole paragraph and sentence content flow as a core abstraction within our system.

They followed the flow of entire sections of unstructured text by detecting similarities between text pairs. They did this by globally finding similar candidate pairs, then locally aligning them to find text passage overlaps, and then using the series of overlaps to determine news source relationships. The reason for their text alignment approach is simple: OCR of early modern newspapers is very poor, leading to mangled sentences, which require a bioinformatics inspired approach, akin to comparing two pieces of DNA for the same genes. The power of this system is that by utilizing string kernels, all that is required to generate candidate pairs are sequences that contain characters with subsections that contain repeated content. Once two candidate pairs were found globally, a final local alignment was achieved using the Smith-Waterman algorithm to determine string overlaps. We utilized the ideas of paragraph and sentence level unstructured text tracking, as well as the analogies to bioinformatics problems in analyzing genes, and leveraged toolkits in that area to help produce our views on the content graph using paragraph vectors.

Despite the apparent difference between modern day news systems and early modern newspapers, the pattern of newspaper propagation still follows a similar source to source content similarity flow, operating on the order of weeks rather than hours. However, despite the power of such a generalizable approach, it has failings outside of its specific application. This system requires exact or near exact text alignment in content between two articles (e.g. exact copying), and doesn't generalize to content that is semantically similar. Its advantage over other systems is that it can extend to unstructured text level as it involves messy substring overlaps. However, an issue with this approach, which we attempt to address in our system, is that it is unable to determine if two pieces of semantically similar content are connected despite there being little or no overlapping exact text.

A fresh approach to these systems was attempted in the work of Vakulenko et al. (2016) [24]. Their goal was to improve upon the previous information diffusion

models and provide a useful abstract content dissemination graph that could be used by journalists to shed light on how news propagates online. Their approach was to track n-gram like grammatical relations through the news media. Their core insight was to parse news articles and convert them into a bag or relations, rather than conserved content, or sequential n-gram strings. This allowed them to extract out related content with great specificity, by using grammatical parsers that convert a freeform news article into a series of: subject -> predicate (verb) -> object relations.

To achieve this, they parsed each sentence into a grammatical parse tree and then traversed that to generate a series of relations. They extracted these relations by using normalized predicates as markers around which they could generate their subject -> predicate -> object relations. Specifically, they converted the predicates into a series of synsets (sets of synonyms) or lemmas (the canonical form of a verb - e.g. fighting/fought are represented by fight). They then combined these predicates with a series of flags, such as past tense, future tense, negation, and auxiliary verbs. By then connecting them to objects (nouns after the predicate) or subjects (nouns before the predicate), they were able to generate relations that could be found across articles.

This improves on previous systems that require exact, or near exact and sequentially matching content to analyze information propagation over time. By essentially joining these grammatical triplets across news articles they can generate a content dissemination graph. The following image (**Figure 2.2**) demonstrates this over their dataset for the query ‘president barack obama -> state D’. The query can be explained by imagining that you are looking for all the times President Barack Obama and a past tense verb occurred within a sentence (e.g. President Barack Obama said).

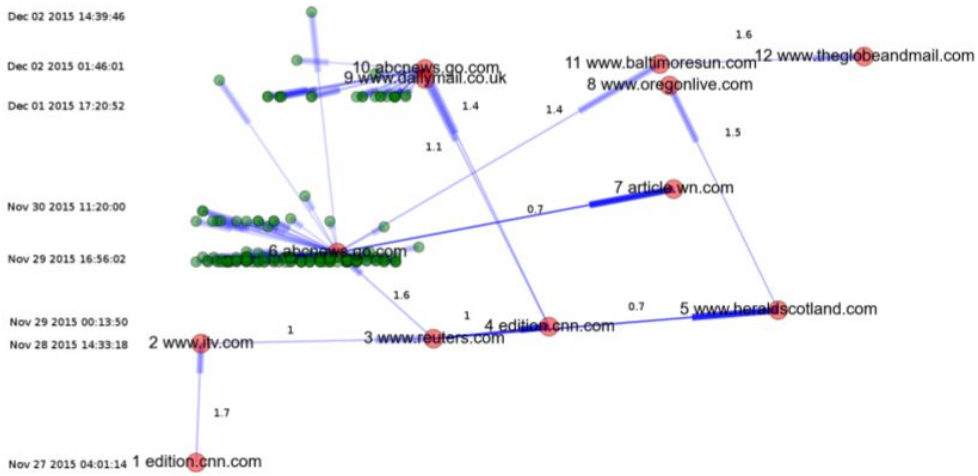

Figure 2.2: Sample information diffusion model for ‘president barack obama state D’ from Vakulenko et al. (2016) [24]. Notice how tracking similar content at an event level can immediately extract out source to source relations by using single source nearest neighbours.

To generate this graph they made a few assumptions. They assumed that all content was related, that each piece was related to its closest neighbor, and that each was sourced from a single source. They generated these neighbor connections by using a Nilsimsa hashing function to connect similar relations.

Despite the power of this system, it has numerous shortcomings that our system aims to address. It requires the creation of specific language grammatical parsers. It requires users to define the query and search through the content space to generate graphs. Due to its simplifying assumption of a bag of relations at a sentence level, it loses out on being able to track more complicated pieces of content. Finally, it is dependent upon synsets in Wordnet. We do utilize their ideas of tracking semantically similar content across sources, however we use paragraph vectors, rather than predicates as our core feature. We also utilize the entire content graph, not simply a bag of predicate relations, and extend our analysis to full sentences and paragraphs. We also use their idea of using a hashing like function with nearest neighbours to generate content dissemination graphs.

In summary, our approach attempts to address many of the concerns previously raised. Our goal is to combine, extend and evolve upon the ideas analyzed within the literature review, and use them to help answer questions that arise from the following Problem Statement.

3. Problem Statement

The amount of information that users must consume to simply keep up with the news has grown exponentially over the last two decades. In a single day, there can be thousands of news events and millions of news articles. Understanding how each article fits into the global context of a news event, and how its reporting relates to or differs from other articles requires hours of careful research and analysis. Existing systems, both academic and commercial, fail to address these issues and do not provide a generalizable content analysis workflow that would allow users global insight into the events they read every day. In the case of commercial systems, they merely cluster and produce frequency counts, expecting users to make sense of the content graph by reading all the articles. In the case of academic systems, they all use simplifying abstractions that require them to only operate on specific subsections of the text such as links, quotes, predicates, or exact text matches. In turn they do not generalize to all paragraphs and sentences at a semantic level.

To overcome these problems, we propose to analyze news articles from existing news events at a semantic level. We use paragraph vector models that can ingest all the sentences and paragraphs within articles to produce semantic vector representations. We then use these semantic representations to discover all the interrelationships and differences that exist between news sources. Our novel insight is combining unsupervised paragraph vector models trained on news article data along with clustering aggregation algorithms to produce novel views on the entire event content graph. We then demonstrate the utility of these views by leveraging existing analysis toolkits to generate useful applications that will allow users to analyze news events at a total content level. As a result, we will allow users to compare and contrast content rapidly, observe how content is reported over time, and observe the relationships between news sources at an aggregate level.

We aim to address the following questions for our research project:

- Can we automatically train paragraph vector models on representative messy real world news articles to show that our system is generalizable and self-training?
- Do the models we train approach state of the art performance against reference models on standard semantic evaluation tasks, especially on in-domain news tasks?
- Can we use trained paragraph vector models to generate paragraph and sentence vector indices which can be used in conjunction with aggregation

functions, such as k-nearest neighbours and hierarchical clustering algorithms, to generate views on the content graph?
- Can we use these views on the content graph to aid media analysis in useful demonstration applications to analyze information propagation and news source relationships?
- Can our system scale up to the volume of news events currently published?
- Is our system cost effective?

In the following section we propose a proof of concept approach for an end-to-end unsupervised paragraph vector model semantic analysis system which can operate by simply ingesting news events. We then analyze how well our approach addresses the questions posed above in Results (Section 5) and Applications (Section 6).

4. Proposed Approach

4.1 Overview

In this section we explore the background of the various abstractions, models, and algorithms we will use in our overall approach, the datasets we trained on, and the evaluation methodology we used to validate our proof of concept.

4.2 Datasets

4.2.1 The Signal Media 1 Million Article Training Dataset

We will be using the Signal 1 Million News Article dataset for model training purposes [10]. We chose this dataset as it was a standardized, reasonably large crawl of news articles that accurately represents the type of data we wish to analyze.

A thorough analysis of the dataset was performed by the original authors Corney et al. (2016), and it's from this analysis that we summarise details of the dataset henceforth [10].

The dataset was created to stimulate research in large scale News Information Retrieval (IR). The dataset is a cross section of news articles from September 2015 which includes a mix of mainstream news sources, as well as blogs, and other publications.

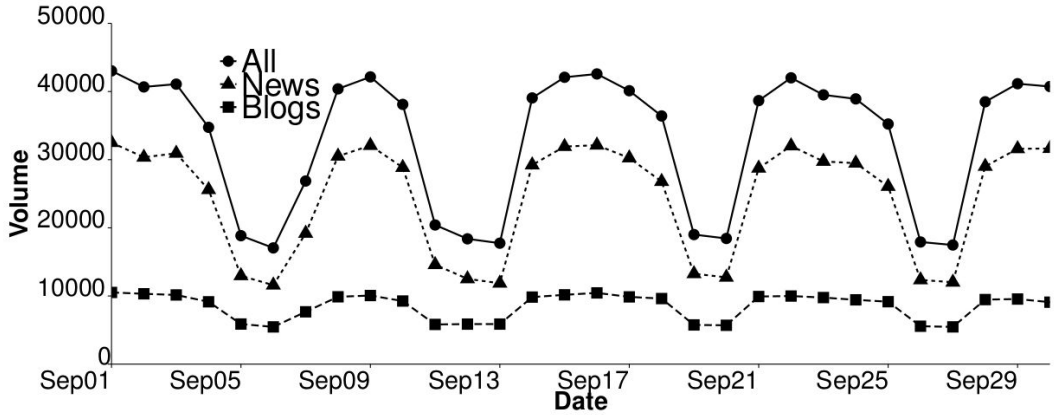

Figure 4.1: A volume timeline from Corney et al. (2016) that demonstrates the volume dynamics of the news cycle [10].

We chose to train on this dataset as it comprises nearly 93,000 real world news sources, and differs from the cleaner single source datasets used to train the reference models we will be comparing against [14]. Thus it gives us an accurate representation of the semantic performance that could be achieved by our proof of concept if it were to be scaled up to a massive stream of real world news data.

As the crawl is from a thousands of real online sources, the dataset includes liberal amounts of duplication, noisy data, incorrect language articles and code snippets that were not correctly parsed out. This gives us an extremely realistic dataset on which we can train our unsupervised models. In essence it will let us observe how robust paragraph vector models are on what is essentially a continuous stream of real world news data.

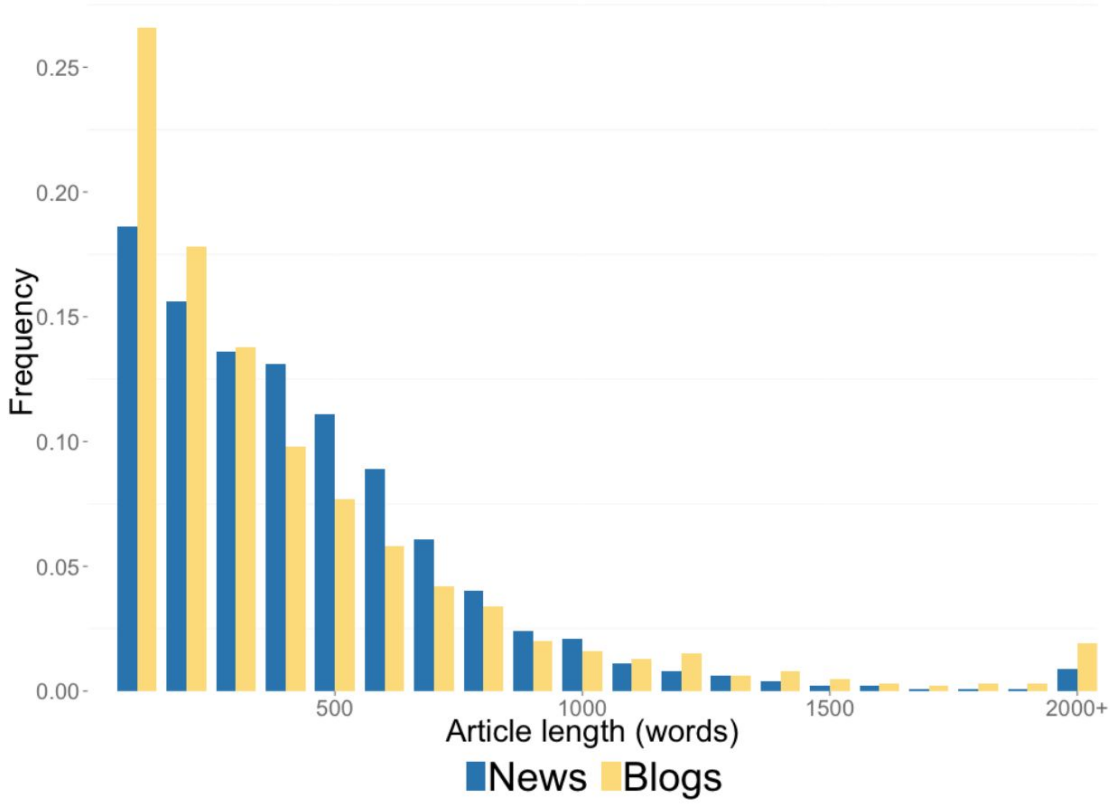

Figure 4.2: A histogram taken from Corney et al. (2016) that illustrates the length and breakdown of article length statistics between News and Blog sources [10].

4.2.2 Analyzed Events

To illustrate the applications of our proof of concept we take events from two news event clustering systems, Google News and Event Registry. These events provide us with a real world demonstration set of news articles that were aggregated into news event clusters for actual users.

By using these real world event clusters, from outside of our training data, we are able to demonstrate the effectiveness of applications developed using our proof of concept system in the general case. Another benefit to using these events is to show how our system can integrate with existing news aggregation workflows, and in turn, demonstrate how we can leverage the real time crawling abilities of existing aggregation systems.

We extract one above the fold event cluster from Google News (that is news articles deemed most important on the main page of the event), which was taken from early September 2016 [29]. This event concerned the ceasefire brokered between the main parties involved in the Syrian Civil War (the Syrian event). We choose this

event because it was a popular world news story and by using the most interesting articles listed above the fold on Google News, it gives us a good representation of the articles that real world users are actually likely to browse.

To extend our system to the long tail of news reporting, we extracted a series of events from Event Registry that allow us to demonstrate how our system is able to scale up from just a few dozen articles to thousands.

The first news event cluster from Event Registry concerned a new recommendation by the World Health Organization (WHO) that urged countries to tax high sugar drinks to alleviate obesity and health issues, taken from early October (the WHO event) [30]. This event contains a couple of hundred articles.

The second news event cluster from Event Registry concerned the release of Amazon's new unlimited music streaming service, taken from mid-October (the Amazon event) [31]. This event contains a couple of hundred articles.

The third news event cluster from Event Registry concerned the performance of U.S. presidential nominee Hillary Clinton at the third U.S. presidential debate against Donald Trump, taken from late October (the Hillary event) [32]. This is our largest event, with a couple of thousand articles.

The reason we choose these 3 specific events from Event Registry is because they represent a good spread in terms of the types of news events we see everyday. The WHO event represents a news cluster that is sourced from a study or report by a well known organization. The Amazon event represents a new corporate press release or product announcement. The Hillary event represents thousands of articles that are not wholly related to well defined single topic, but instead a myriad of article types (e.g. live blogs, opinion pieces, factual reporting etc.) on the same event, showing us how we can perform on noisy data.

4.3 Data Analysis Stack

We utilized the Anaconda scientific platform with both Python 2.7 (training/generating vectors) and 3.5 (parsing/manipulating articles) versions [33]. In addition we also used the sci-kit learn library for the standard implementation of clustering and distance algorithms, NetworkX for the generation of content networks, and gensim for the training and generation of paragraph vectors [20] [12] [21]. Stanford CoreNLP is also used in the preprocessing of the article dataset into tokenized vectors [27]. Furthermore, we generated graph visualization using the Cytoscape graphing library [22].

4.4 Semantic Models

4.4.1 Word Vectors

For this proof of concept we used paragraph (document) vectors generated by doc2vec from the gensim python library. However before we introduce the relevant concepts and detail for paragraph vectors, it is necessary to understand, at a high level, how word vectors (generated by word2vec, also from gensim) are generated.

The core concept for both approaches is that of a shallow neural network from Mikolov et al. (2013) [17]. The goal of the word vector model is to generate high dimensional vectors, often on the order of 100 to 300 dimensions, which essentially encode the semantic position of a word. These vectors are a distributed representation of a word, and are effectively the result of smearing the word's semantic meaning across all of the numeric values [18]. The goal of the shallow neural network at the heart of the model is to force vectors that represent similar words (which start out as random weights) to be located near similar locations in a higher dimensional vector space [18]. It achieves this by relying on the Distributional hypothesis which argues that given similar words are used in similar contexts, we can simply push the vectors of words that share similar contexts together over time to encode their semantic meaning [17].

The model is trained through millions of samplings of some document corpus at a context level. The samplings, or context windows, are rolling windows over the document corpus taking for example a series of 15 words. Once a sample is taken, one of the words (the target word) is held out, and the remaining words are input as 1-hot vectors into a neural network's input. The goal of the neural network is to guess the target word that was kept out based solely upon the context words that surround it. By using the target word's 1-hot vector as the value to check against, the neural network is able to use the loss on a target word for use with stochastic gradient descent and backpropagation to learn the correct weightings for words given their contexts over time [17]. The embedded weights generated by the neural network are then used as vectors that represent the semantic meaning of the word.

The power of this model is the fact that it is totally unsupervised. It is a model that is able to generate a compressed semantic representation of a word corpus by simply ingesting thousands of contexts from real world documents.

In term of specific implementation Mikolov et al. (2013) discovered that the skip-gram model with negative sampling in general outperforms the continuous bag of words

(CBOW) in both speed and semantic performance, and thus we will focus on skip-gram with negative sampling here [18].

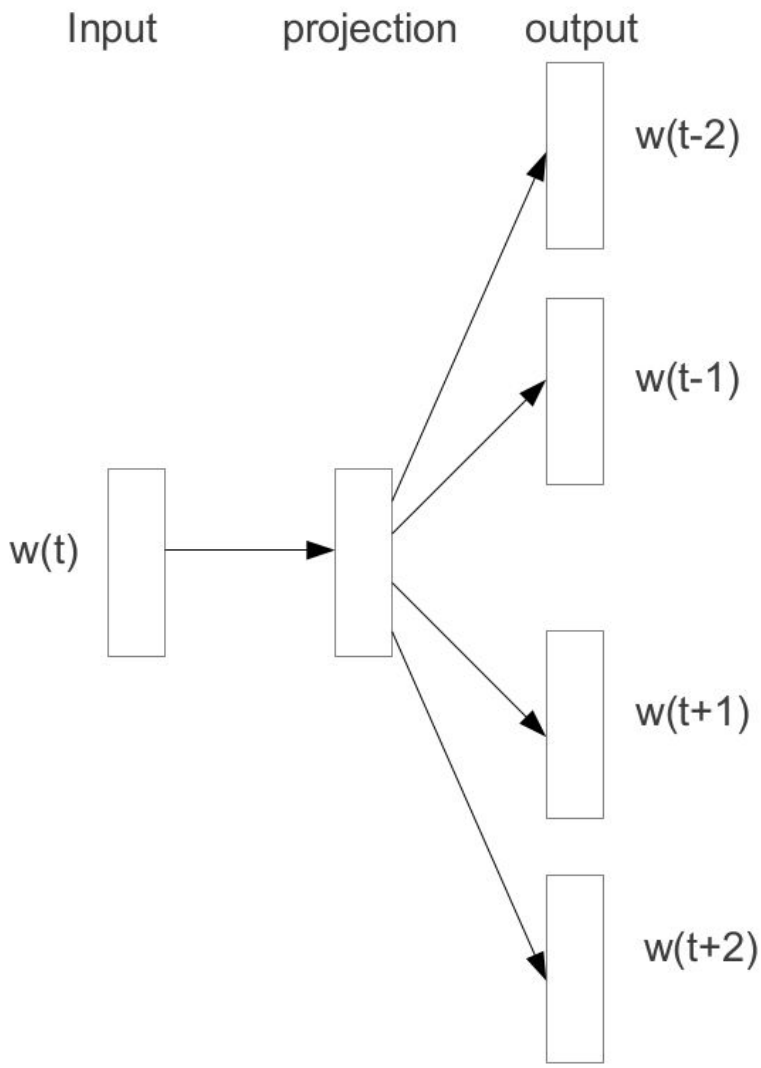

Figure 4.3: A demonstration of the skip-gram with negative sampling system used to rapidly generate high quality word vectors from Mikolov et al. (2013) [18].

In the skip-gram model we attempt to predict context words given a target word [18]. This increases the amount of data by essentially training on pairs of target and context words thousands of times. Hence the term skip-gram, as our context windows skip our target word. To boost performance we combine it with negative sampling [18].

Negative sampling essentially attempts to improve the embeddings by generating negative samples to train our neural network against [18]. The negative samples are words taken from a random sample of the corpus and thus are not likely to be

predictive of the context. We then optimize the loss function to ensure the neural network is able to push the weights of the words around such that it is able to discriminate between negative samples (random words) and our target word for predicting the output context words.

In the course of generating the semantic vector representations of these words, other surprising relationships are also encoded, such as those between capital cities and countries [18]. This demonstrates the power of word vector models.

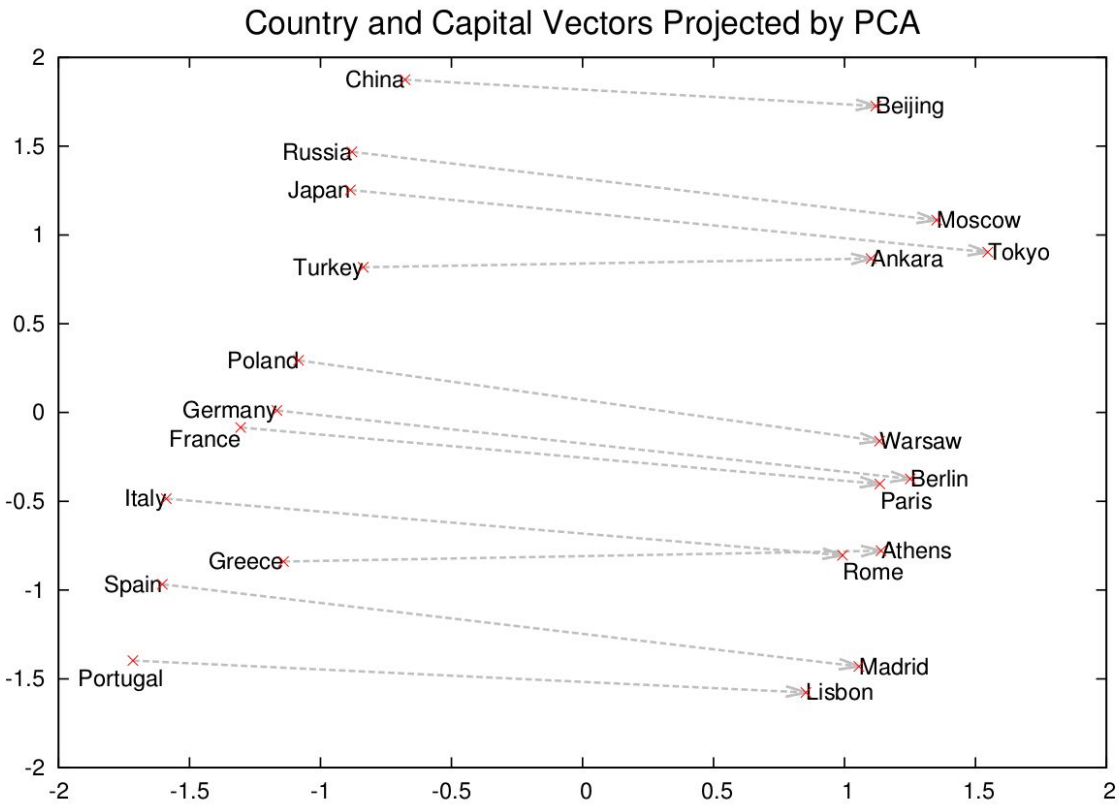

Figure 4.4: A demonstration of the higher dimensional relationships that are persistent and encoded in the differences between related words from Mikolov et al. (2013) [18].

4.4.2 Paragraph Vectors

Paragraph vectors are a natural extension of word vectors to a sentence, paragraph or document level. The goal of paragraph vectors is similar to that of word vectors, namely generating string vector embeddings that can be used to semantically compare various texts. The core theory is extended by Le and Mikolov (2014) from word vectors to paragraph vectors [15].

Once again 100-300 dimensional vectors are learned for each sentence or paragraph, with paragraphs or sentences that are semantically similar having their vector embeddings pushed together in a similar manner as to that of word vectors. The method of optimization is the same as for the previous models, stochastic gradient descent on the predictive output with back propagation into the weighted embeddings for input and output [15].

Once again there are two methods of training, one known as distributed memory (DM) and the other known as distributed bag of words (DBOW). DBOW is analogous to the skip-gram model with negative sampling, and in practice it's found to be the higher performance, and more robust method, and hence we will use it for this thesis [14].

The way DBOW operates is that a target paragraph is assigned an id, and converted into a vector, as are the words within the paragraph. The paragraph vector is then used to predict the word vectors, just like the skip-gram with negative sampling word vector model. Therefore the target word vector, becomes a target paragraph vector, whose values are used to predict a random sub-sample of the words within a context window of the paragraph [15].

The name distributed bag of words comes from the fact that the paragraph follows the distributional hypothesis at a bag of words level, with the aim being that paragraphs that predict similar random bag of words (or contexts) will be semantically close [15].

For both word vectors and paragraph vectors we utilize off the shelf solutions in the popular topic modelling library gensim.

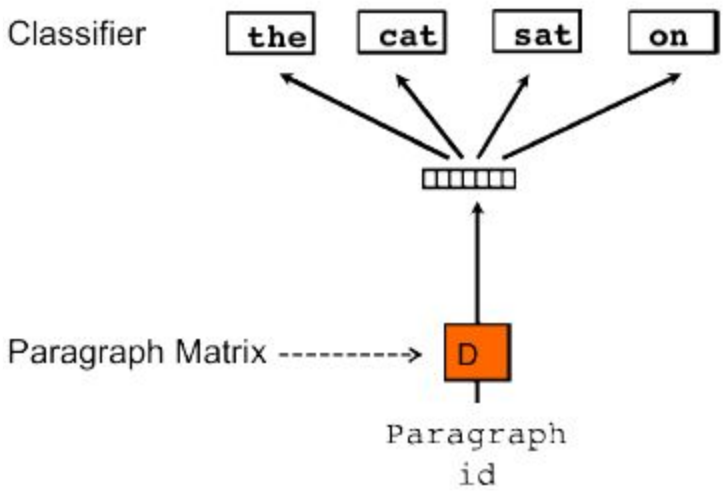

Figure 4.5: An example of the training process for a paragraph vector model using DBOW from Le and Mikolov (2014) [15].

4.4.3 Reference Models

We utilize two state of the art reference models with differing hyperparameters from Lau and Baldwin (2016) trained on two single source datasets (Wikipedia and AP News datasets) [14]. We use them as benchmarks for comparison against our trained models on standard semantic evaluation tasks. These reference models will be referred to as ap_dbow, and enwiki_dbow, as they are in the original paper.

We then use their best practice hyperparameters to train paragraph vectors on messy multi-source real world online news article data (Signal 1M News Article Dataset). These models will be referred to as ap, intersect_ap, wiki, and intersect_wiki. The relevant hyperparameters used to train our models can be found in the Appendix in Section 11.1, with both ap/intersect_ap representing the ap_dbow hyperparameters, and wiki/intsersect_wiki representing the enwiki_dbow hyperparameters.

The difference between our general ap/wiki hyperparameter models and the intersect_ap, intersect_wiki variants is that during the vocabulary generation phase of the paragraph vector model training, we intersect the word embeddings found in the larger ap_dbow, and enwiki_dbow models into ours. By comparing the general models with the intersection models, we will be able to see if utilizing pre-trained word vectors increases performance, as was argued in the original paper.

By comparing our models trained on what is essentially a stream of real world scraped news articles (Signal 1M dataset) against the two reference models we can observe the best combination of hyperparameters for a continuous unsupervised training system. In doing so, we can demonstrate how by only ingesting news articles with the best hyperparameters, we should be able to scale up our proof of concept system to continuously train solely on real world news data streams.

We demonstrate this by training on the Signal 1M Article Dataset, which despite being smaller than those used by the reference models, is still highly representative of online news data streams. Our goal is to show that even with a limited amount of messy real world multi-source news data that contains significant amounts of poorly formatted, noisy and duplicated content, we are still able to reach near state of the art performance with our paragraph vector models.

4.4.4 Training Datasets Comparison

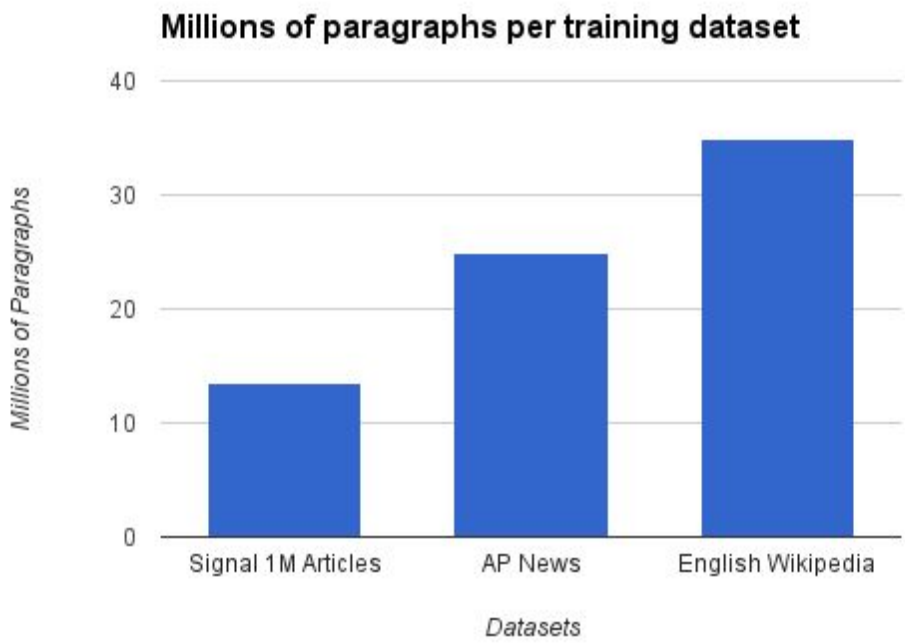

Figure 4.6: A comparison in the number of millions of paragraphs found in the respective training datasets for our models, ap_dbow, and enwiki_dbow.

As can be seen in **Figure 4.6**, there is a significant difference in training dataset size between our models and the reference models. Our goal, however, is not to show that training on real world messy news article data will perform better than the reference models, but to show that they can perform close to them on tasks involving

the semantic discrimination of in-domain sentences. By training multiple hyperparameter models, we should be able to optimize which set of hyperparameters will perform best on a scaled up systems. This would in turn prove that our system would be able to continuously train in an online fashion when scaled up, by only ingesting news articles and events generated by high-velocity external systems like Google News or Event Registry, with no other requirements.

4.4.5 Model Evaluation

For model evaluation we use the English Semantic Textual Similarity (STS) tasks from the *SEM and SemEval for the years 2012-2016 [4] [3] [2] [6] [5].

STS allows one to test the performance of semantic models by applying them to a variety of different language domains where they are made to estimate the similarity of a pair of sentences. The ratings fall on a scale of 0 to 5, with 0 meaning the two sentences are not remotely semantically similar, and 5 meaning they are perfectly semantically similar. These calculated ratings are then compared to human ratings to demonstrate their power.

To evaluate the model performance against human ratings we calculate the standard Pearson’s r correlation using the STS toolkit. The closer this correlation is to 1, the better the model is said to perform. Comparing different models is as simple as observing which models exhibit the higher Pearson's r.

We also qualitatively evaluate the graphs and clusters generated in the applications stage, as the purpose of that aspect of the thesis is to generate useful abstractions or views upon the content graph. We rely on the fact that the STS results will give an indication as to a paragraph vector model's power in general. Using this as a base, we can build upon the paragraph vector models as an abstraction that will allow us to organize sentences and paragraphs. In turn, this allows us to display their interrelationships so that rapid analysis of a news event at a content level can occur.

4.5 System Algorithms

4.5.1 Cosine Similarity

For this proof of concept, we heavily utilize the standard measure of similarity between vectors known as cosine similarity [13]. The method calculates the amount of similarity between two vectors by observing that the cosine of the angle between two vectors gives a dimensionless measure of their similarity. This similarity is

calculated through a combination of a dot product between two vectors and then a normalization by their multiplied magnitudes.

Mathematically the cosine similarity between two vectors is defined as follows: Given two vectors A and B, where θ is the angle between them, we can define the cosine similarity between them with the following equation:

$$sim_{cosine}(A,B) = cos(\theta) = \frac{A \cdot B}{||A||||B||} \quad (1)$$

By the nature of the cosine function, the values are within the range of [-1, 1]. The greater the value, the more similar two paragraph vectors are, and by extension the more semantically similar the two strings they were calculated from are.

4.5.2 k-Nearest Neighbours

We used k-Nearest Neighbours (k-nns) along with cosine similarity throughout the system for the generation of the exploration, content, and source propagation graphs.

In essence, k-nns simply involve the calculation of all pairwise distances between all pairs within a list of vectors using some distance metric [8], which in our case is cosine similarity, and the vectors are our sentences and paragraphs.

We then simply order the distances of a particular vector's neighbors by similarity, and extract those closest to it by some thresholding value, whether that is a similarity threshold or simply the number of neighbors we wish to obtain.

We utilized a single nearest-neighbour model for the creation of propagation and source analysis graphs. We used a multi-neighbour model for the generation of the exploration graph. We threshold the neighbors we retrieve by ~10-20 which we found worked well in practice, or by a similarity threshold of 0.6 for the generation of the exploration graph data structure.

4.5.3 Hierarchical Agglomerative Clustering

4.5.3.1 Algorithm

We utilized hierarchical agglomerative clustering (HAC) with cosine distances to generate semantically similar content clusters [7]. We used these clusters as seeds for the generation of a content propagation graph in conjunction with k-nns.

The way HAC works is that it takes a list of vectors and assigns each to a singleton cluster. It then iteratively works its way up from each singleton cluster by linking together any two clusters that exhibit the greatest similarity, until we end up at a single global cluster. This process can be seen in the dendrogram of paragraph vectors in **Figure 5.12**.

The reason for choosing HAC over other methods was due to the fact that it accurately represents how the underlying content is structured (see **Figure 5.12**). At a high level, due to the massive amount of similarity in sentences between news reports, we are able to rapidly cluster them together in a bottom-up approach. This lets us generate a content dendrogram that accurately represents how the content is interrelated at a local and global level.

Another advantage of this system, as compared to something such as k-means clustering is that it does not require one to define the number of clusters within an index, allowing us to scale to any number of sentences, paragraphs or articles by using a linkage metric with an inconsistency criterion.

4.5.3.2 Similarity Metric

The similarity metric used to join clusters is called the linkage metric, and for our purposes, we found that the complete linkage method produces the best qualitative results in our applications [7].

Using the complete linkage method we calculate the maximum pairwise distances between clusters. We do this by finding the two most dissimilar items between all cluster pairs. We then join the two clusters whose maximum pairwise distance for the two most dissimilar items is the global minimum across all cluster pairs at each iteration.

To calculate the complete linkage using a similarity metric formally, we find the minimum similarity between any two vectors in clusters A and B by using cosine similarity. We then join the two clusters that exhibit the maximum complete link similarity.

$$sim_{complete}(A,B) = \min_{x \in A,\, y \in B} sim_{cosine}(V_x, V_y) \tag{2}$$

As our goal is to generate well defined and well separated content clusters, we found that the complete linkage metric produced the best results over other methods due to

the fact that it preferentially generates tight small clusters which accurately represented the underlying sentences and paragraphs [7]. This is as opposed to single link clustering which involved chaining sentences that were not well separated.

4.5.3.3 Cutting

To generate our clusters we have to cut the dendrogram using some adaptive metric. We utilize flat clustering on our linkage structure with the default inconsistency method using sklearn's fcluster.

The inconsistency method operates by cutting clusters based upon the dynamics of the linkage structure [26]. In essence it observes that if a join link generated during HAC occurs with a much greater distance than the average of the joins at its level, then it likely defines a cluster boundary. We found that this method provided distinct and tight clusters on top of which we were able to generate useful content propagation graphs.

4.5.4 News Article Preprocessing

In the process of our analysis, we found that for both Google News and Event Registry anywhere from 50%-90% of articles published within the long tail of news sources were near exact duplications of existing content from either press agencies or other news sources.

As our goal is to demonstrate differential analysis of sources from differing viewpoints on similar data, as well as to illustrate hidden semantic connections between sources, we filtered out these duplicates. This is because, with duplicates intact, they overwhelm the graphs we generate and the similar content we find. This makes them work effectively like noise, providing little insight, as they simply cover up differential reporting and hidden correlations.

To filter out these duplicates we used a combination of exact headline matching as a heuristic for similar documents, as well as a simple top level sentence filter that collates and removes articles that exhibit a significant number of exact sentence matches. When combined these heuristics drastically reduced the number of duplicates.

4.6 Summary

In this section we provide a step by step summary of our entire workflow.

4.6.1 Model Training

1. Acquire a batch of representative news articles, in this case those from the Signal 1M Article Dataset.
2. Convert them into a series of paragraphs.
3. Tokenize and lower case the paragraphs with Stanford CoreNLP.
4. Build a vocabulary of the words within the paragraph, intersecting word vectors if required.
5. Train a paragraph vector model with best practice hyperparameters using gensim.
6. Shuffle paragraphs randomly on every iteration to improve performance.
7. Measure performance on standard semantic evaluation tasks.

4.6.2 Content Index Generation

1. Take a batch articles that comprise a news event from an existing clustering system (Google News, Event Registry) to leverage their algorithms, crawlers and to demonstrate integration with existing consumption workflows.
2. Take each article in the event and split it up into a list of paragraphs or sentences, depending on which type of content you wish to analyze.
3. Take a trained paragraph vector model and convert each paragraph or sentence into a paragraph vector.
4. Use these vectors as essentially a paragraph or sentence index on top of which we can apply clustering and graphs to generate a view on the content graph.

4.6.3 Generating Views on the Content Graph

By integrating all the content in an event (e.g. sentences/paragraphs) with a combination of paragraph vector models, k-nns, and HAC, we can generate useful graph data structures.

We use these graphs to create demonstration applications that allow us to complete differential analysis, watch how a piece of content propagates over time, and observe the interrelationships between news sources.

These graphs are expanded upon in the Applications section.

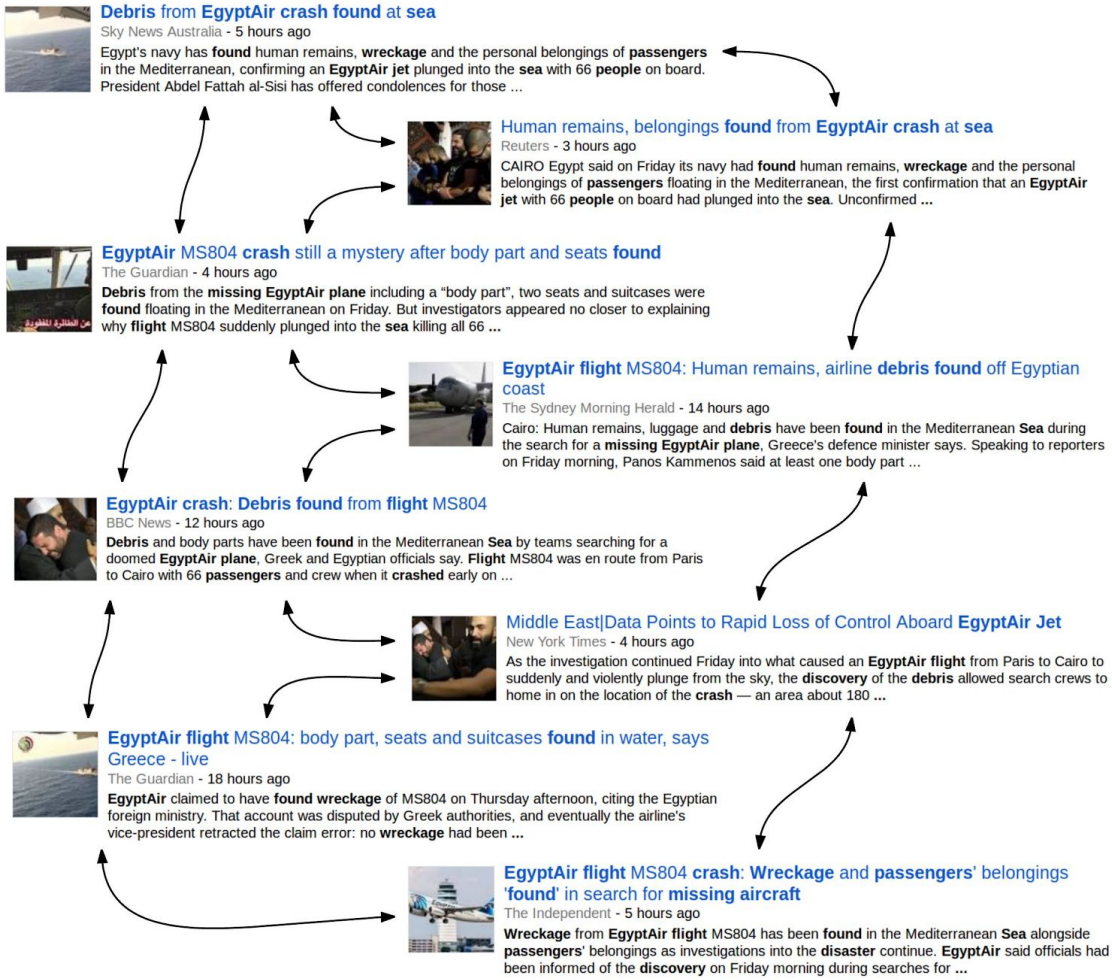

Figure 4.7: An example of news article interrelationships. Identical content is highlighted in bold from Google News [28].

4.6.4 Complete Proposed Approach Validation

1. Train semantic models on real world online news data (e.g. the Signal 1M Article Dataset) and evaluate their effectiveness on standardized Semantic Evaluation (SemEval STS) tests, as well as for speed and cost, to demonstrate how they would perform when scaled up.
2. Use the trained models to generate a vector index of paragraphs and sentences for various news events.
3. Generate views on the content graph by applying high similarity threshold clustering aggregations (hierarchical agglomerative clustering and k-nns)
4. Use these views on the content graph to develop demonstration applications to aid media analysis.

5. Results

5.1 Training

5.1.1 Cloud Platform Comparison

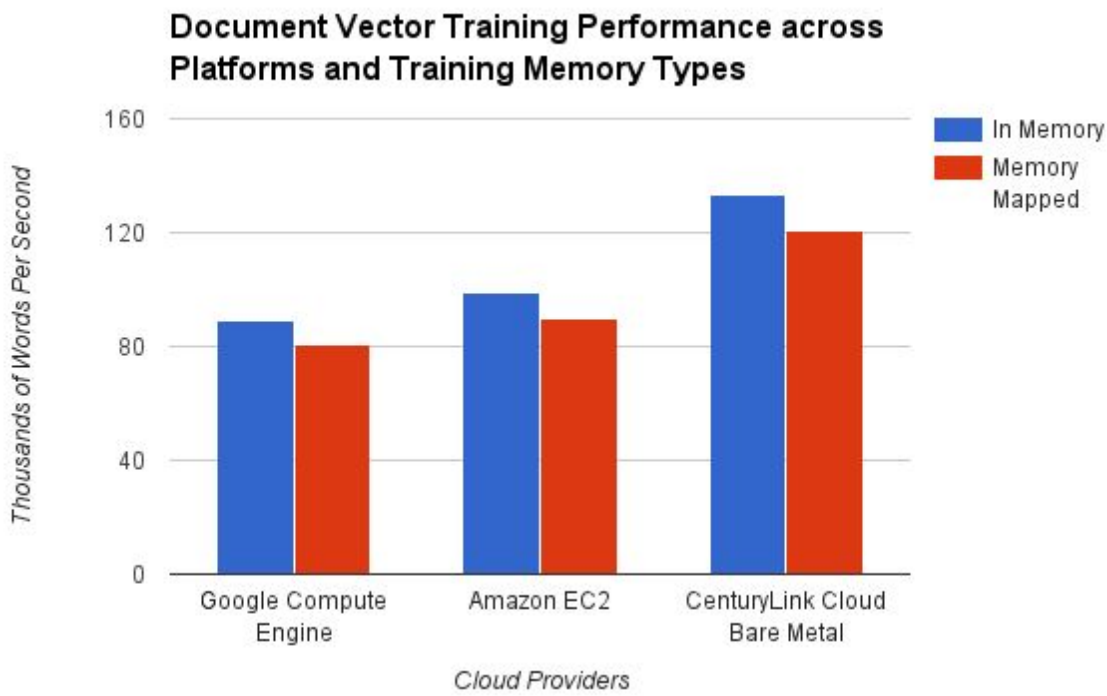

Figure 5.1: A graph comparing the speed of paragraph vector training on different cloud platforms.

Before we began training our large vector models, we wished to determine which cloud provider would give the best cost to performance advantage. We took 3 equivalent server options from mainstream cloud providers, Google Compute Engine, Amazon EC2 and CenturyLink Cloud Bare Metal (bare metal) and compared their performance using gensim's thousands of words per second metric [42] [43] [44]. The metric essentially measures how many words from the context windows the model can process every second.

Before embarking on this analysis we determined the optimal hyperparameter settings for gensim training performance and found that, in practice, 6 workers on 8 core machines provided the best performance. We believe this occurs because gensim processes its document as a queue of mini-batches. By keeping our workers

(aka processes) at an even number that is slightly less than the number of cores, we are able to better saturate the machine. Increasing the number of cores past 8 harmed performance, as did choosing a different number of workers, and we believe that this is due to memory locality issues.

From the graph in **Figure 5.1** we can see that bare metal servers perform the best, with Google Compute Cloud servers being the slowest, and Amazon EC2 in between. In terms of cost, we found Google Compute Cloud to be the cheapest, but the slowest, bare metal being the second cheapest and the fastest, and Amazon EC2 being the least performant per dollar. However as prices and hardware change continuously this may not be true in the future

We hypothesize that the reason bare metal servers perform better is because they have full control over both their CPU and their memory, leading to fewer interrupts and memory churn, giving us higher utilization. We believe that the bare metal systems outperformed the virtualized instances due to issues with memory cache locality. By avoiding the “jigger” that occurs with overutilized virtual systems, we believe bare metal platforms, and by extension, dedicated hardware, provide the best case performance per dollar for the computational needs required to train large paragraph vector models.

We also tested the performance penalty of memory mapping for training the models. We found that there was an approximate 9-11% decrease in performance using memory mapping. Memory mapping the training models drastically reduces memory consumption from 16-17 GB (wiki/ap hyperparameters respectively), down to just a couple of GB. This allowed us to use much cheaper instances, and thus we we found the trade off acceptable. We choose CenturyLink Cloud Bare Metal servers due to both their speed and cost performance.

The reason for this analysis was, as the process of training paragraph models is crucial to a scaled up system, understanding the performance dynamics of the underlying computation providers is critical.

5.1.2 Training Cluster Specifications

We found that the best combination of price to performance was bare metal hardware, and hence we utilized CenturyLink Cloud’s bare metal instances [44].

We used their 4 core instance that costs $0.56 per/hr. We trained on 4 machines, each learning one of our paragraph vector models, with each machine having the follow specifications:

- Intel Xeon E3-1271 v3 @ 3.60 GHz
- 4 cores (1x4) (8 virtual)
- 16 GB RAM
- 2x 1TB HDD Raw @ 7200 rpm
- 0.91 TB RAID 1 usable

SERVERS	CPU	MEMORY (GB)	STORAGE (TB)
4	16	64	7.81

Figure 5.2: A dashboard view from CenturyLink Cloud illustrating the resources required to train our 4 hyperparameter models on the Signal 1M Article Dataset.

5.1.3 Model Training Time

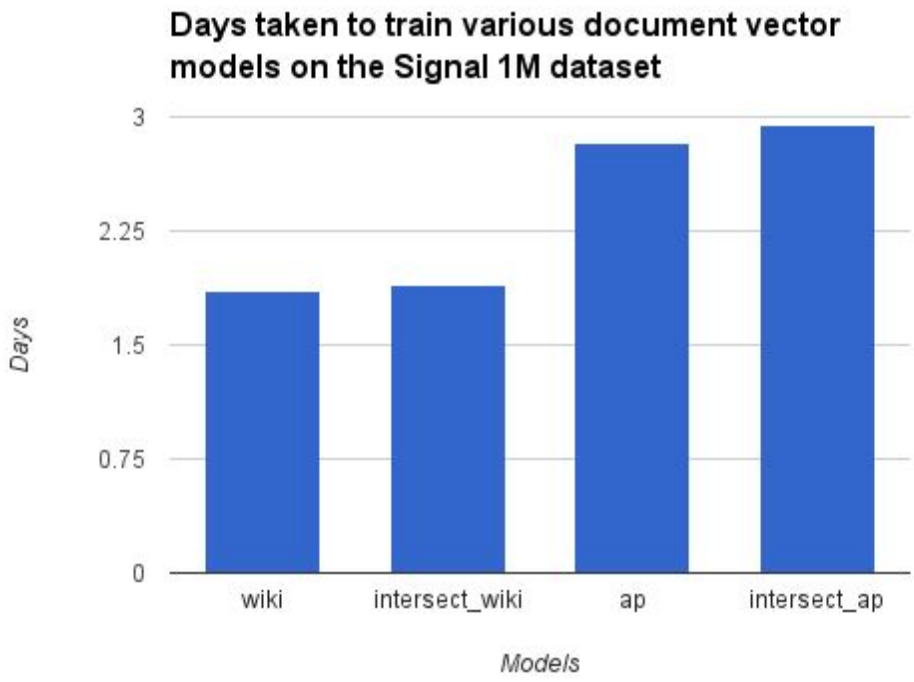

Figure 5.3: A graph demonstrating the numbers of days to train the models with different hyperparameters.

A critical component to training large machine learning models, and optimizing their hyperparameters, is knowing how long it takes for different hyperparameter combinations to produce good performance. On our training cluster we found that

the wiki hyperparameter servers took significantly less time than our ap hyperparameter models (**Figure 5.3**).

We believe that this is mostly due to the increase in the number of training iterations (20 vs. 30 for wiki vs. ap), with another contributing factor being the difference in minimal thresholding for terms, leading the ap models to process more terms (200,338 vs. 317,238 words for wiki vs. ap).

We notice that intersecting the word vectors of our reference models into ours has almost no time penalty.

5.1.4 Model Cost

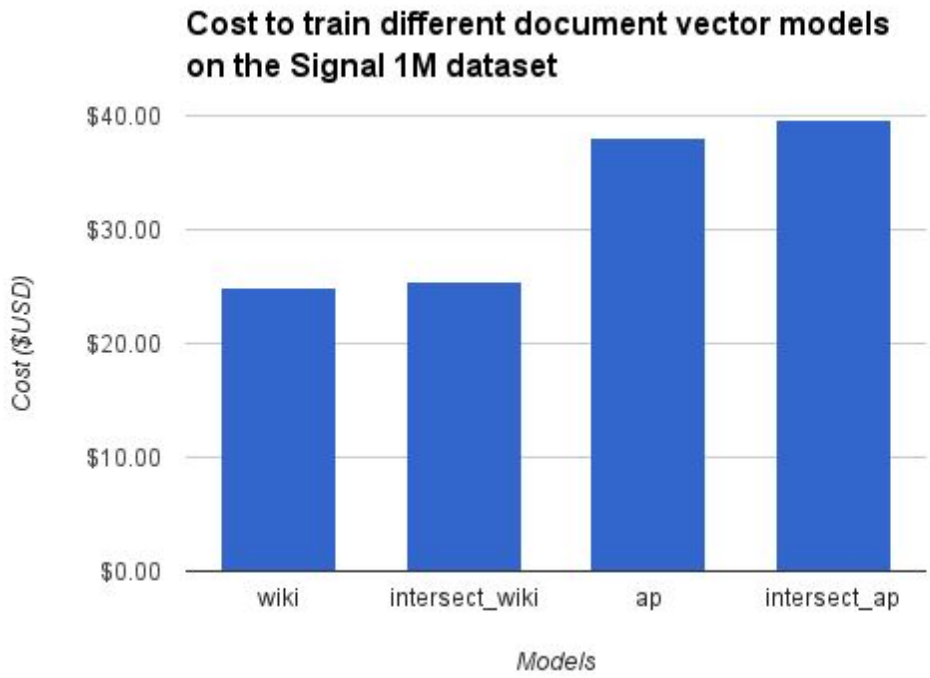

Figure 5.4: A graph demonstrating the cost to train the models with different hyperparameters.

Another component to an operational system is how expensive the differing hyperparameters are to train for the same semantic performance. As servers in cloud instances are priced by the hour, we found that length of time was directly correlated to cost. Thus our wiki models cost much less to train than our ap models.

5.2 Semantic Evaluation

5.2.1 Semantic Results

The following graphs compare the Pearson's r correlation results across all models for multiple years of the SemEval STS evaluation task, the higher the better.

In turn the graphs compare how our multi-source, noisy, and relatively smaller real world news article models perform against the much larger and cleaner single source reference models across multiple semantic domains. This in turn demonstrates how our training system would likely perform when scaled up to continuously ingest messy news article data.

The first four bars in each of the following graphs are our trained models, with the final two being the reference models.

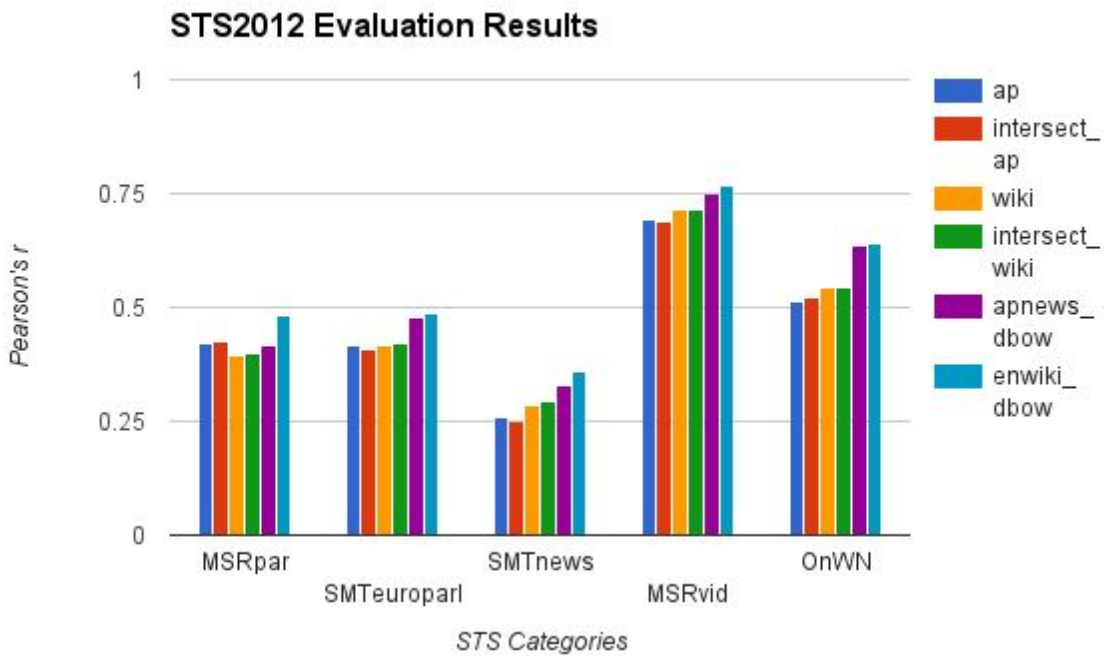

Figure 5.5: A graph that represents the STS2012 evaluation task comparing the Pearson's r correlation across multiple domains and models.

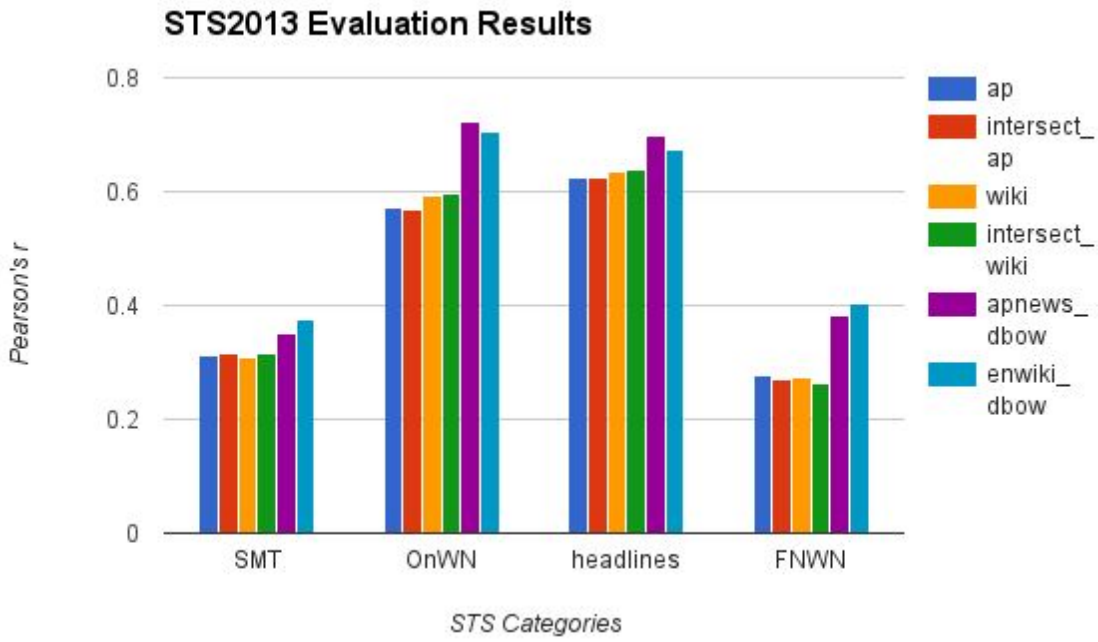

Figure 5.6: A graph that represents the STS2013 evaluation task comparing the Pearson's r correlation across multiple domains and models.

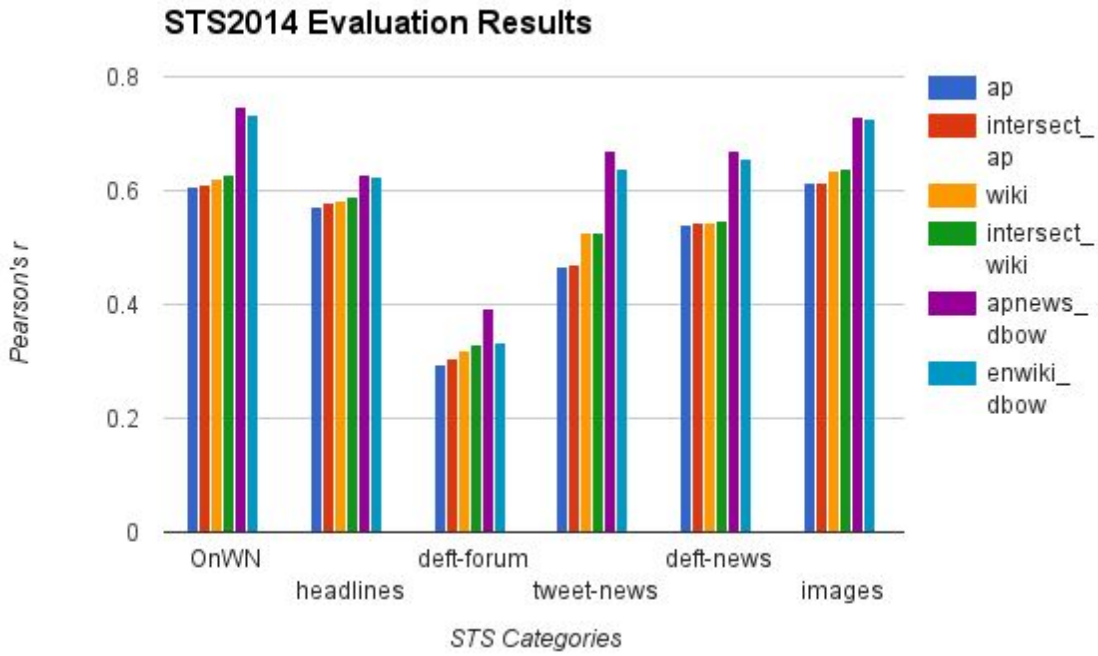

Figure 5.7: A graph that represents the STS2014 evaluation task comparing the Pearson's r correlation across multiple domains and models.

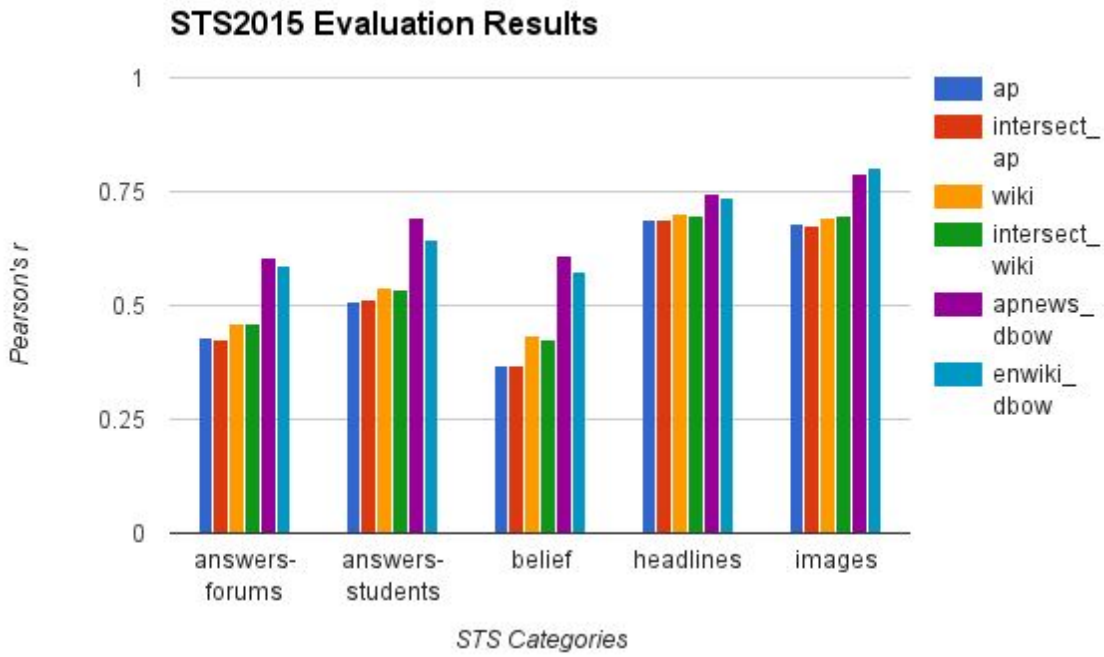

Figure 5.8: A graph that represents the STS2015 evaluation task comparing the Pearson's r correlation across multiple domains and models.

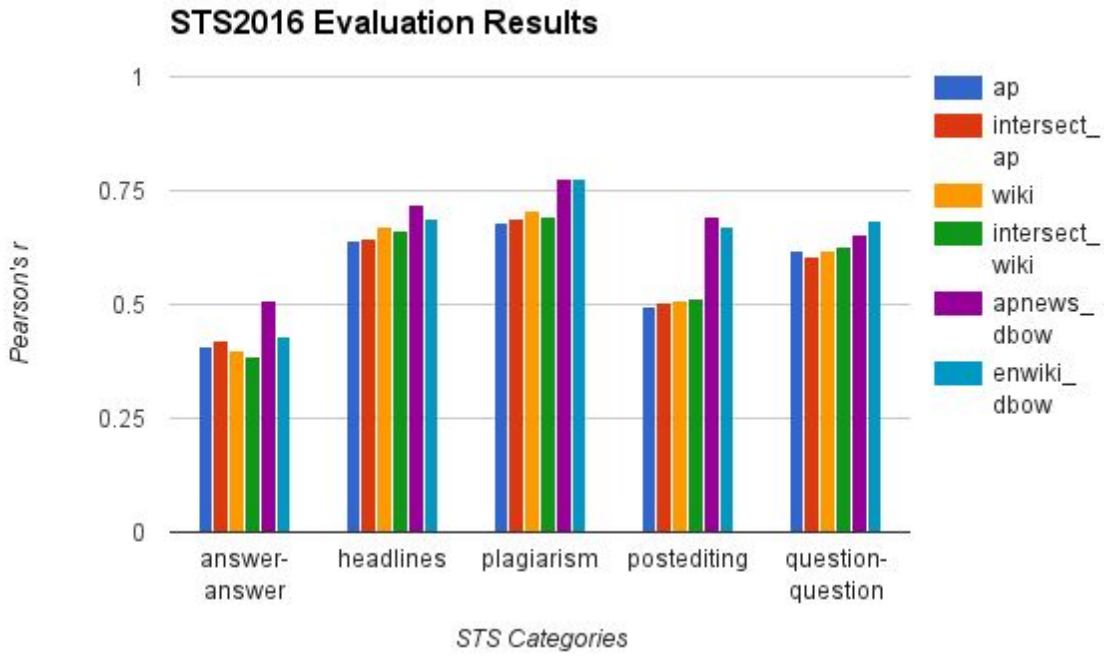

Figure 5.9: A graph that represents the STS2016 evaluation task comparing the Pearson's r correlation across multiple domains and models.

5.2.3 Semantic Results Discussion

In general the results of the STS evaluations are clear, messy real world news data models, especially if the dataset is smaller than the reference models, will underperform much larger single source datasets across all domains. Specifically enwiki_dbow was trained on the entirety of English Wikipedia, and ap_dbow was trained on nearly 5 years of articles from a single news agency [14]. This is in comparison to our Signal 1M article dataset which comprises only a single month of messy real world online news article data.

However our goal was not to prove that our workflow would operate better than models trained on cleaner single source datasets, but to show that, despite how badly formatted or noisy our input stream of news articles is, our paragraph vector models will still rapidly approach state of the art performance, even on limited data, and especially in in-domain areas. This lets us to demonstrate that we should be able to merely scale up our system on a constant stream of news events to both train our system as well as to generate useful applications completely unsupervised.

Across the evaluation years above, we notice that in general our models perform surprisingly well across many domains, and specifically they work well in the domains of news headline, image caption and plagiarism comparisons (STS2016, STS2015, STS2014), except for STS2012. Thus we should be able to scale up both the training and graph generation of our systems by simply continuously ingesting news events, giving us close to state of the art performance across hyperparameter models for our in-domain tasks. We compare the relative performance between our trained models in the next section.

5.2.4 Best Trained Model

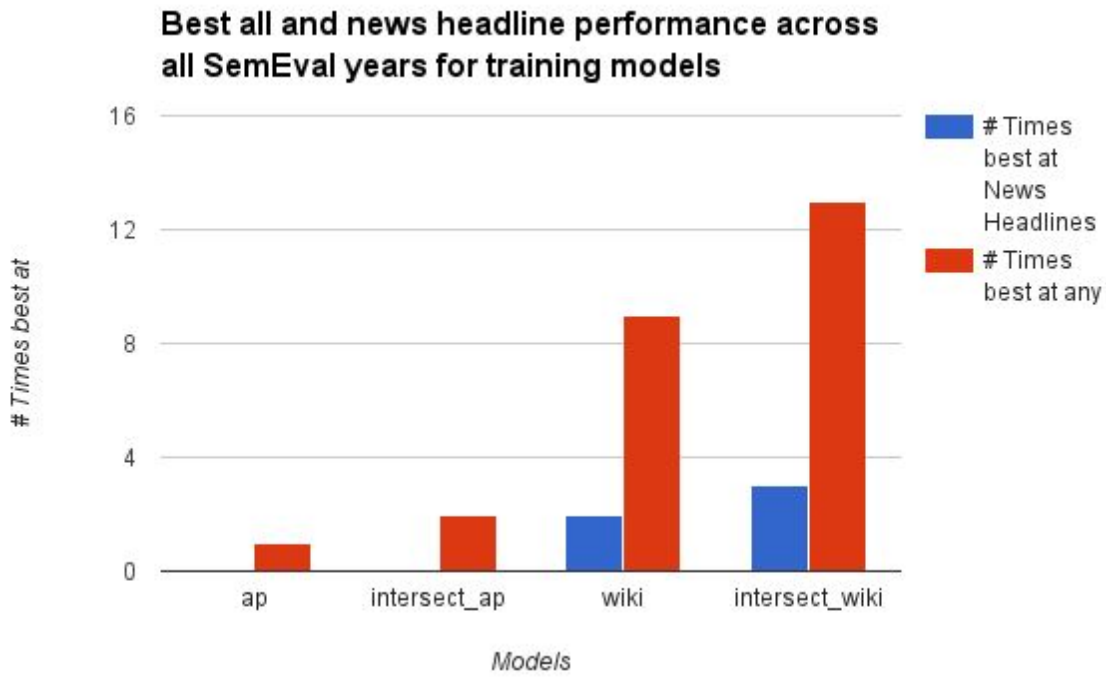

Figure 5.10: By counting which model performs best in each domain across years, we are able to show which hyperparameters seem to perform the best on limited noisy datasets.

We calculated the best overall performing trained model by simply counting which trained models performed best across all domains (e.g. max score), and a separate class focusing on the in-domain dataset of headlines. As can be seen in **Figure 5.10** the wiki hyperparameters appear to give better overall max results, both across domains and specifically on our in-domain headline section. Thus if we were to scale up our system, we should utilize the wiki hyper parameters, with an incremental performance boost by intersecting the enwiki_dbow word vectors (intersect_wiki).

We believe that the wiki hyperparameters outperform the ap hyperparameters because they essentially filter out a significant amount of the noise within the Signal 1M News Article dataset by using threshold parameters that act as a high pass filter. Specifically, the min_count hyperparameter which only allows words that appear a certain number of times to be included in the vocabulary is greater for wiki (20 occurrences) vs. ap (10 occurrences). This leads us to hypothesize that the ap hyperparameter models learn lower quality vector representations due to the fact

that the signal within the dataset is washed out by our neural network learning noise. In the next section we observe which of the reference models is the best overall, and in turn which best represents an example of a model that would simulate a scaled up system.

5.2.5 Best Overall Model

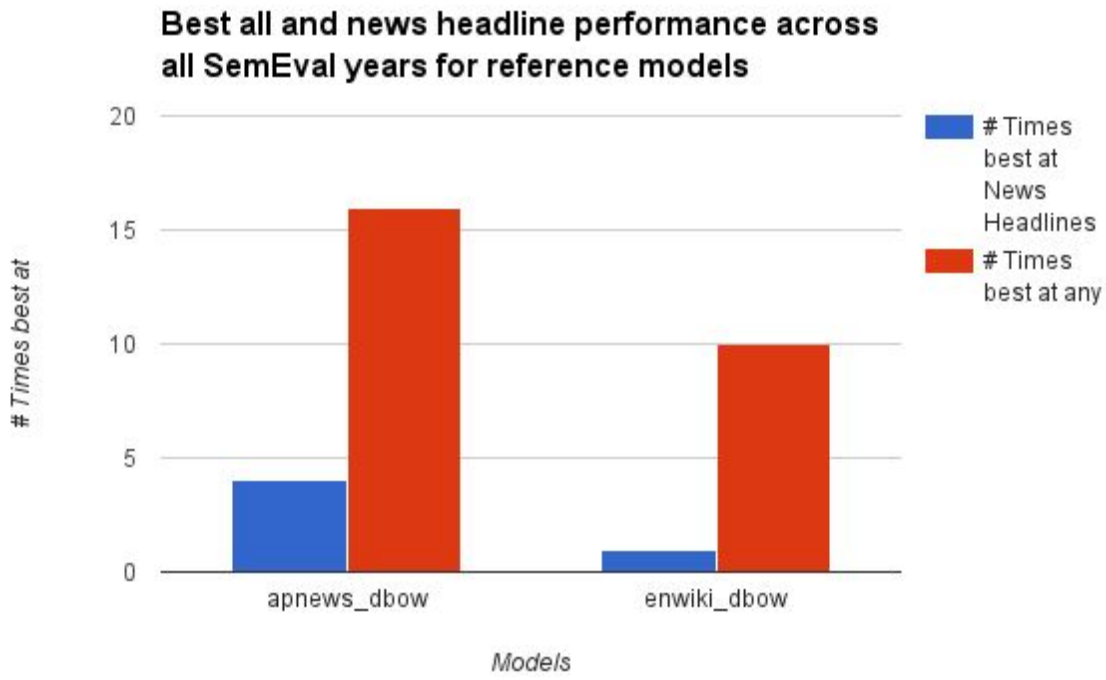

Figure 5.11: By counting which model performs best in each domain across years, we are able to show which hyperparameters seem to perform the best on limited noisy datasets.

In the case of reference models, it appears the benefits and disadvantages of the hyperparameters are reversed. With a large, clean single source dataset, it seems that the ap hyperparameters are more likely to learn better representations of the underlying semantics. Another contributing factor may be that as Wikipedia is a community generated dataset, it approximates content more like that found in our Signal 1M Article dataset, reducing performance.

Thus to train on real world noisy data it appears that a high thresholding hyperparameter setting improves performance by filtering out the noise, but on a high quality professionally created document dataset, like the AP, learning at a lower frequency level aids performance. However as apnews_dbow best represents a scaled up version of our underlying dataset, as we are processing news articles,

albeit with far more noise, we will use that model as the semantic index generator for our demonstration applications, as it represents the kind of performance expected in a scaled up system.

5.3 Clustering Qualitative Evaluation

5.3.1 k-Nearest Neighbours

We found that in practice k-nns were an extremely fast and efficient way of generating lists of neighbors. k-nns allow us to create undirected content exploration graphs by joining semantically similar content. We found that k-nns were far better for content exploration tasks for the simple fact that they are far more serendipitous. This is because their goal isn't to find isolated content clusters but merely to connect related content. We found that when combined with HAC clusters as connection seeds, we were able to select sections of the content graph for examination, and by ordering these graphs over time, we were able to generate approximate content propagation graphs.

5.3.2 Hierarchical Agglomerative Clustering

HAC acts complementary to k-nns and allow us to select out tight, well defined, and disjoint content clusters, rather than a continuous range of overlapping content. We found using HAC was critical for graph analysis applications, specifically for selecting out content clusters for investigation, and for removing a significant number of connections which at a higher level of k-nns make graphs difficult to read.

These tight clusters can be seen in the dendrogram in **Figure 5.12**, with **Figure 5.13** showing how, at a sentence level, HAC organizes content.

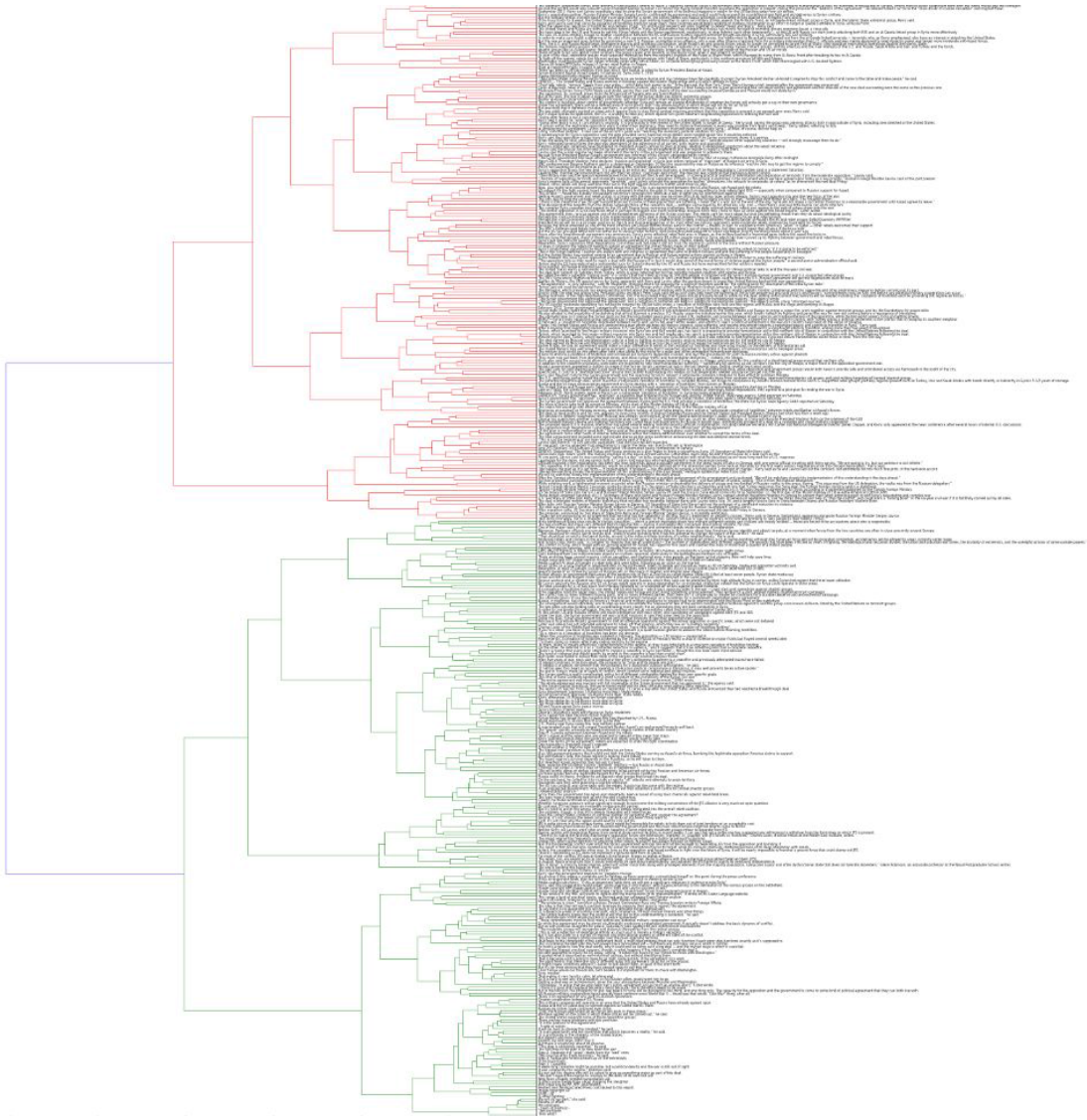

Figure 5.12: An example of a dendrogram produced with HAC clustering on the Syrian event at the sentence level [29].

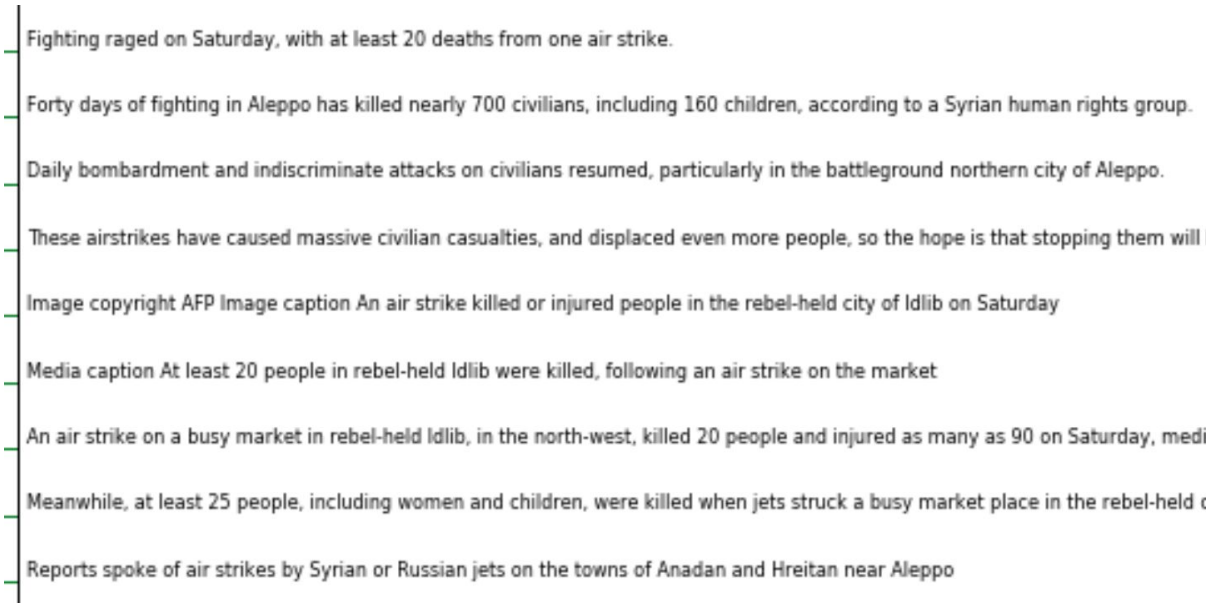

Figure 5.13: A zoomed in portion of the dendrogram found in **Figure 5.12**. Note how semantically similar content, often with no words or even ordering in common, are accurately and reliably organized together using agglomerative clustering with paragraph vectors. A similar pattern appears with k-nns graph linking (**Figure 5.17** and **Figure 5.18**).

5.4 Classification Computational Performance

5.4.1 Index Generation

With any system that wishes to address the enormous amount of news that consumers see every day, being able to process the content we see rapidly is of paramount importance. Hence we perform a speed/computational performance evaluation on the paragraph vector models to see how many paragraphs or sentences our training system can consume, and in turn, how quickly a single computer can generate content indexes.

We find that, in general, we are able to scale up linearly with the size of the dataset for the generation of paragraph vectors at both a sentence and paragraph level, with paragraphs taking less time (see **Figure 5.14** and **Figure 5.15**). We find that there are occasions where there are pathological inputs, such as strings from search farms involving thousands of tags. These can be trivially filtered out, but are a factor when you process a large number of noisy news articles. In general we find the performance to be consistent at the recommended inference hyper parameters (start_alpha = 0.01 and infer_epochs=1000) [14].

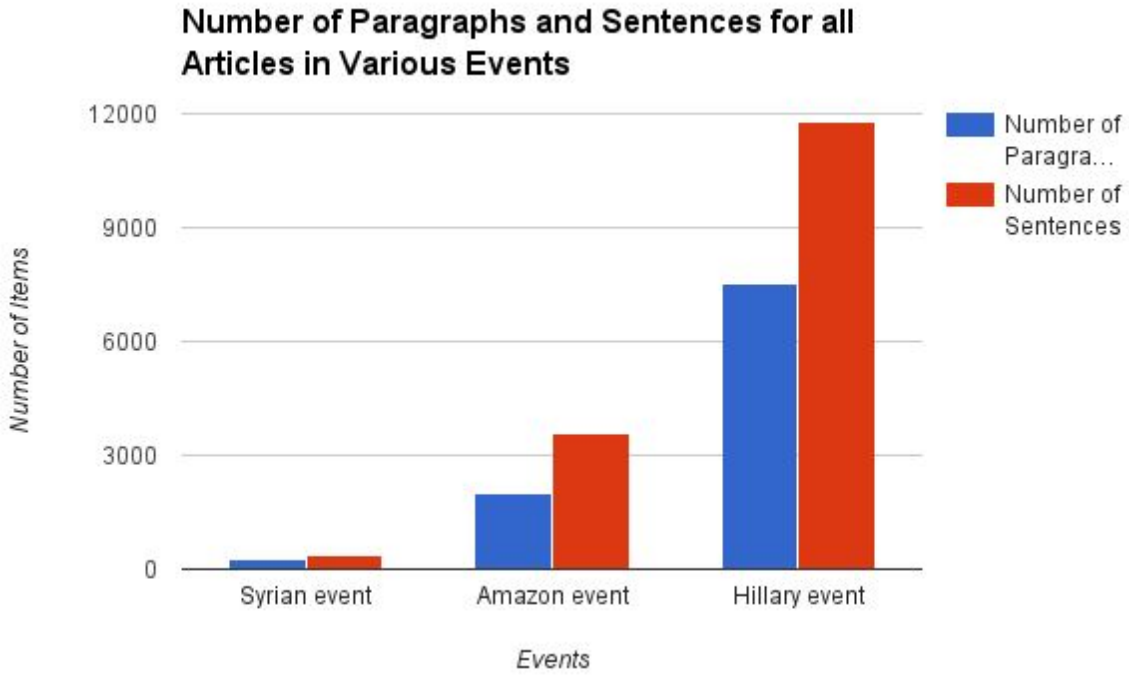

Figure 5.14: A graph comparing the number of sentences and paragraphs found in various events.

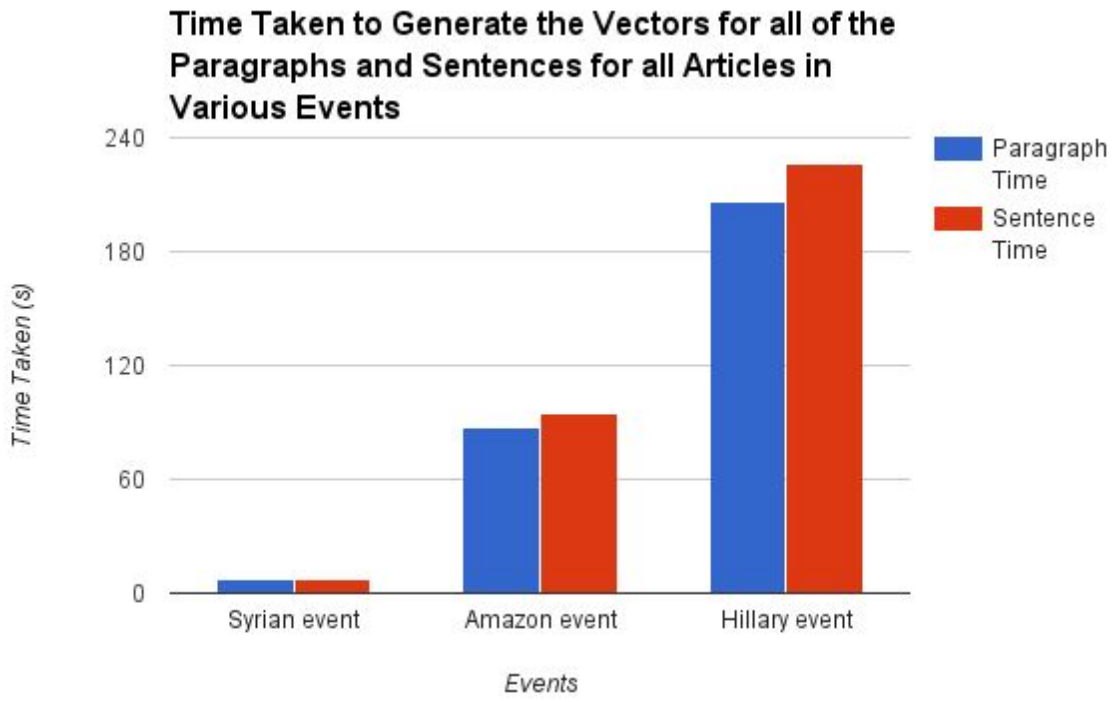

Figure 5.15: A graph comparing the times taken to classify all the sentences and paragraphs for various event clusters. We find that in general it is linear to the input.

5.4.2 Clustering

We found that in practice the computational cost of HAC overwhelms the cost of k-nns, Therefore we focused on the costs of HAC as it dominates clustering costs.

The reason why can easily be seen in the progression of time taken to process the vector index seen in **Figure 5.16**.

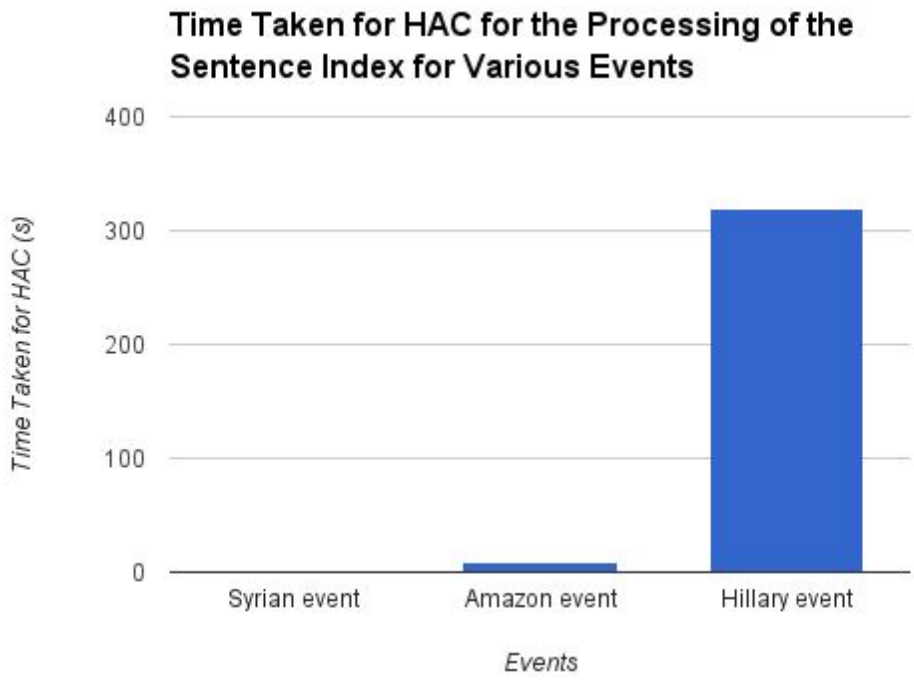

Figure 5.16: The time taken to process the sentence vector index for various sized events as seen in **Figure 5.14**. Note the Syrian event takes less than 1 second.

As HAC with complete linkage clustering is $O(n^3)$ in complexity, it rapidly increases in processing time as it scales up from news events that contain dozens of articles to those that are comprised of hundreds to thousands [7]. However, thanks to the fact that we process events as singular event clusters, with the largest events being capped in practice to a few thousand articles, coupled with a reasonable news article deduplication scheme, we can essentially keep the time cost per event to no more than a few minutes.

In the future we would look into capping the HAC cost using preprocessing steps such as those found in the NIFTY system to break up the vector space, but for our proof of concept, the performance of HAC is adequate [23].

5.5 Content Graph Visualization

In this section we visualize, at a high level, a view of the content graph by connecting the nearest neighbor sentences for the Syrian event [29]. This effectively gives us a high-level view that demonstrates the interrelationships we will exploit within our applications to help users rapidly analyze and observe events at both a local and global level.

As we see in **Figure 5.17**, the content, which in this case are sentences represented by semantic vectors generated by the ap_dbow model, naturally fall apart into semantically similar clusters which can be explored and analyzed. If we observe the insert to the previous figure in **Figure 5.18**, it demonstrates at a local level what each one of those connections and clusters represents. Each edge is a weighted similarity cluster joining two nearest neighbors using k-nns.

This is merely one example of a view on the content vectors at a sentence level. By utilizing different combinations of neighbors chosen, or indeed by using HAC to select out particular disjoint clusters to seed our analysis, we can rapidly develop and display various views that aid the local and global analysis of a news event. We will see more of these graphs in the applications section. We generate exploration graphs comprising solely of k-nns, content propagation graphs that utilize HAC disjoint clusters as seeds for their generation, and news source analysis graphs that aggregate content connections at a source level to demonstrate implicit content networks.

Figure 5.17: An example of the content graph generated by connecting nearest neighbours from the Syrian event [29].

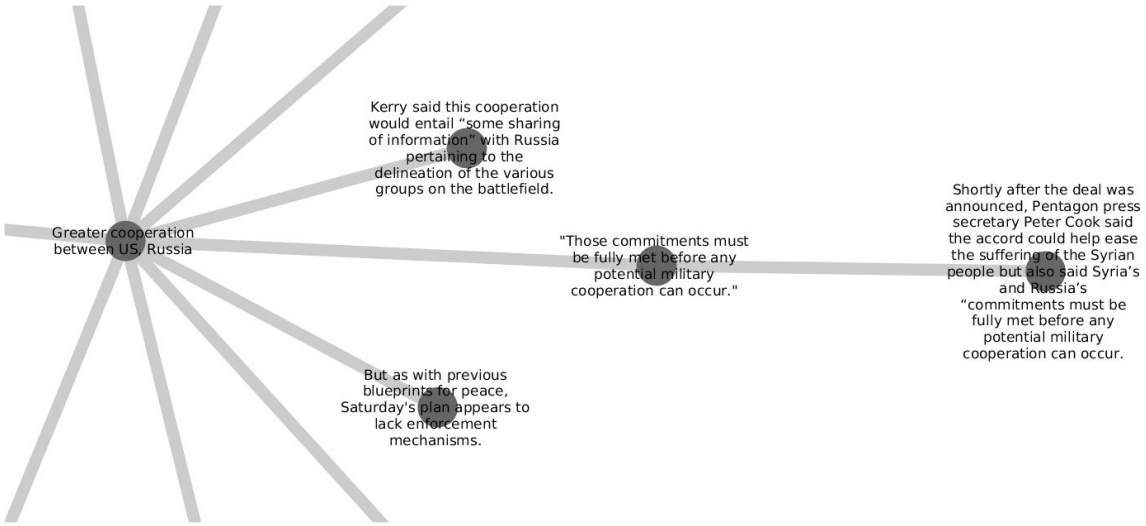

Figure 5.18: An insert of **Figure 5.17** example of the content graph generated by connecting nearest neighbours [29].

6. Applications

Numerous applications can be developed by generating views on the content graph. One can differentially analyze how news is reported on an article by article basis, or scale up to a global view to observe how sources are interrelated. This allows users to see the global context behind any piece of content, and observe the differences and similarities in reporting between sources. We break up the applications into two main types, exploratory analysis and graph analysis. We will focus on the sentence index for the following demonstrations, with vectors generated by ap_dbow, however as our paragraph vectors are generalizable, all applications hold the same for the paragraph index.

6.1 Exploratory Analysis

Exploratory analysis is an extension of the browsing and exploration habits of news consumers to the content level. We demonstrate applications of views on the content graph by exploring applications in related content, contextual comparison and analyzing differential reporting between articles.

6.1.1 Related Content

Related content consumption extends the related article or video paradigms used in online media systems to the sentence and paragraph level. By displaying a group of

the nearest neighbors of a sentence dynamically we can show differing reporting and viewpoints on the same semantic content.

The application seen in **Figure 6.1** and **Figure 6.2** lists the k-nns neighbors for any sentence in the Syrian and WHO events respectively, illustrating how different news sources report on the same event, which aspects they emphasize, and how they position the story. This allows users to click and list the related content for any sentence, or paragraph, and just like related content in other contexts, navigate to similar articles, and highlight the relevant sentences.

The sentence backgrounds have their sentence transparency set by how related they are in terms of cosine similarity. The exploration graph is generated by thresholding the k-nns by a value that works well in practice, which we find to be ~10-20, or with a cosine similarity threshold (~0.6-0.7). The related neighbors are ordered by their similarity, with the most similar at the top, and the least similar at the bottom. HAC is too stringent for this application because the clusters it produces are disjoint, leaving little room for serendipity in the exploration of the content graph.

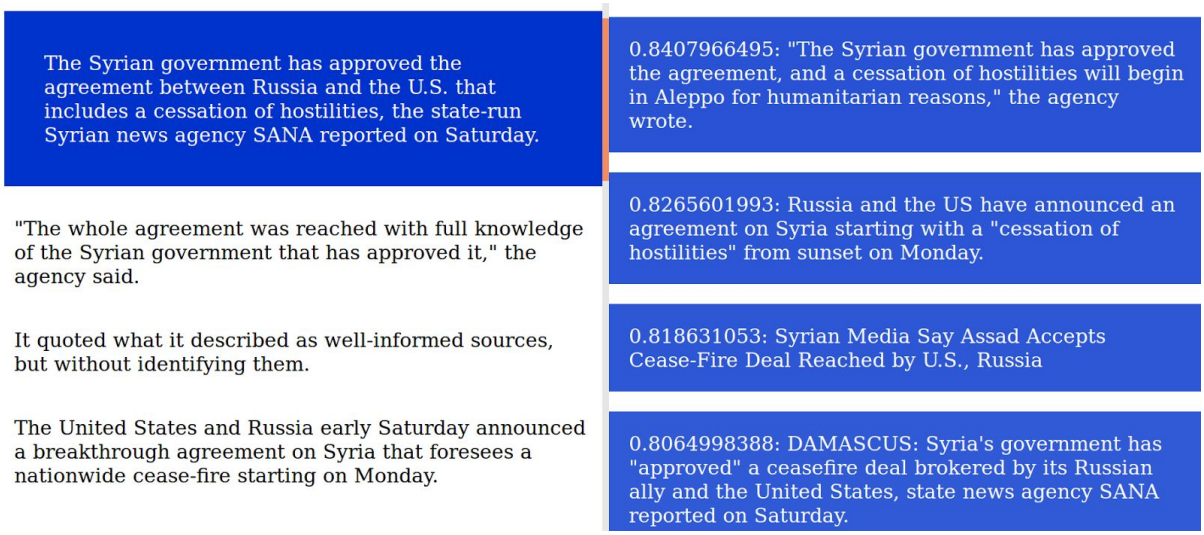

Figure 6.1: An example of a related content application developed using the content graph and k-nns. It allows users to click and navigate related content and their articles by sentence for rapid consumption or to see alternative viewpoints. In this example we use the Syrian event [29].

Figure 6.2: Another example of the application seen in **Figure 6.1** this time with the WHO event [30].

6.1.2 In Context Comparison

A logical extension to related content comparison and exploration for differential reporting is comparing and contrasting semantically similar content between articles within their respective contexts.

The application in **Figure 6.3** operates similarly to the above, but rather than navigating to the article that contains the sentence and merely highlighting it with its neighbours, it instead converts the related content pane into another article, with the clicked content highlighted. This lets one compare and contrast the reporting of a single fact within the context of two separate articles.

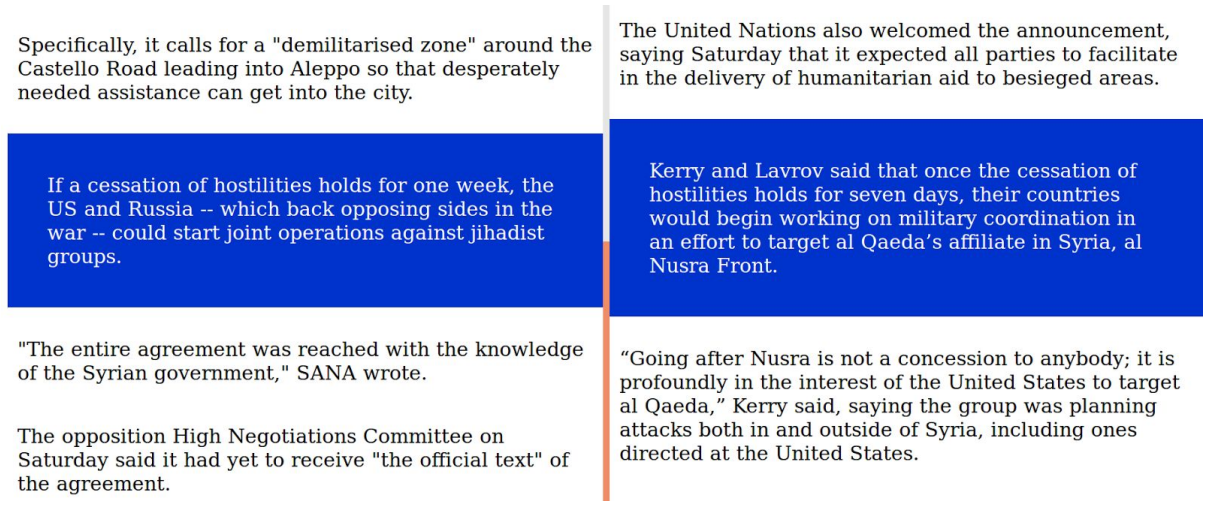

Figure 6.3: An example of an in context comparison application developed using the content graph allowing users to click and read related sections and jump to them in context for rapid consumption and comparison of alternative viewpoints. This example uses the Syrian event [29].

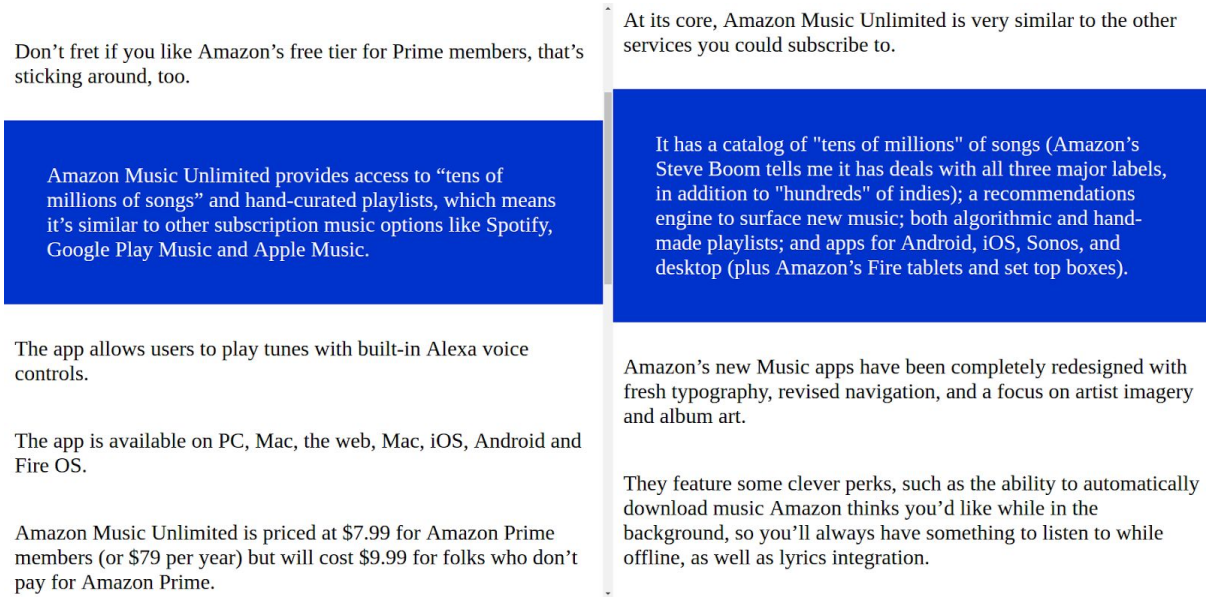

Figure 6.4: Another example of the application seen in **Figure 6.3** this time with the Amazon event [31].

6.1.3 Comparative Analysis

A logical extension to in context navigation is comparative analysis. We essentially extend the in context view to every related piece of content between two articles and then highlight each pair with the transparency being set by how related the two pieces of content are.

This allows one to rapidly visualize how similar two pieces of reporting are and how they are structured in a rapid visual manner. As can be seen in **Figure 6.5** and **Figure 6.6**, two articles can be rapidly compared utilizing unique colors and transparencies for similar content, with voids in the articles indicating differences such as additional opinions, quotes, or news content. These voids can also then be rapidly isolated and compared.

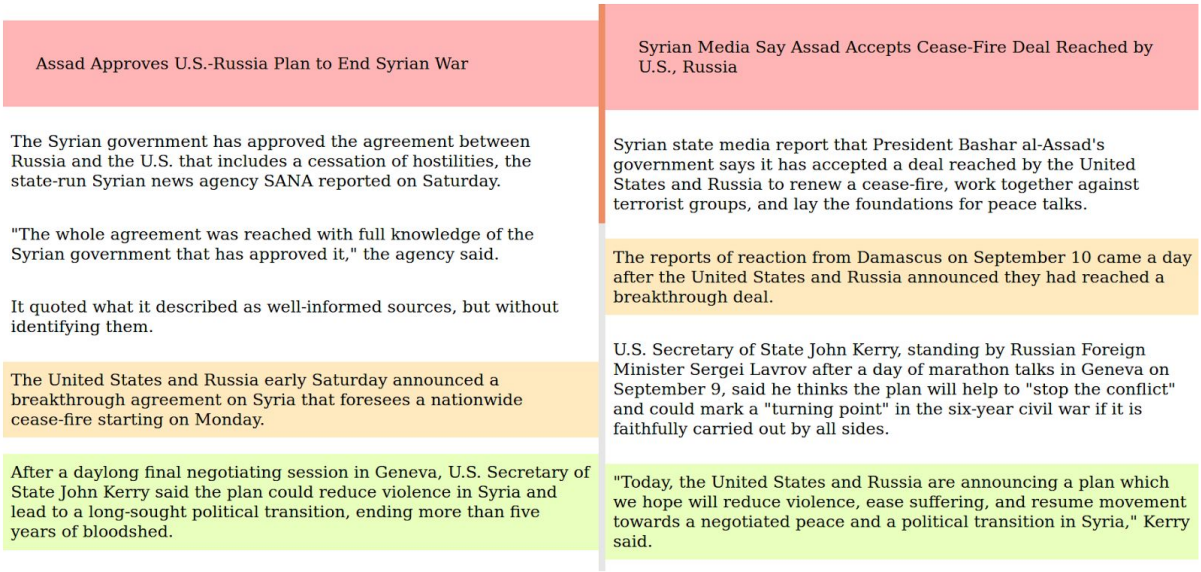

Figure 6.5: An example of a in context comparison application developed using the content graph allowing users to click and read related sections and jump to them in context for rapid consumption and comparison of alternative viewpoints using shading. This example uses the Syrian event [29].

Figure 6.6: Another example of the application seen in **Figure 6.5** this time with the Amazon event [31].

6.2 Graph Analysis

Graph analysis is a key aspect of analyzing news media. From observing how content propagates, to observing how news sources interrelate, it is important to see

at a high level how news media interacts. We achieve this by leveraging toolkits developed by the biomedical community, namely Cytoscape, an open source graphing library to observe how content propagates and interrelates between sources [22]. We also use NetworkX to generate the graphs that are used by Cytoscape [12].

Our first application is observing how a single cluster of related content, such as a series of similar sentences or paragraphs, changes in reporting between sources, and our second application involves the aggregation of content connections at a source level to observe interrelationships.

6.2.1 Content Propagation Graphs

A content propagation graph utilizes the clusters from the HAC system as seeds for the generation of a content graph with the nearest k-nns. Once applied we are essentially able to apply a “mask” to the content graph and extract a well-connected cluster of related content for analysis. By utilizing times taken from the event source, we can simply organize the content by time, allowing us to generate a graph that shows content propagation through time.

This can be visualized conceptually in **Figure 6.7**. The arrows connect together what you can visually see to be similar content. By utilizing the semantic vectors generated by our paragraph vector models, we are able to connect together these relations using a combination of HAC and k-nns.

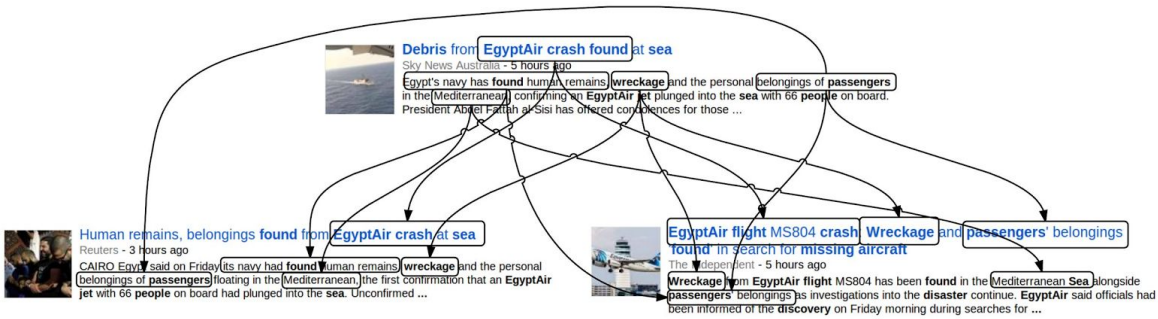

Figure 6.7: An idealized example of all the interrelationships extracted by hand from an event about the EgyptAir crash reported by Google News [28].

We applied this workflow to the Syrian event and selected out one cluster from the content propagation graph [29]. The strength and size of the interrelations indicate the level of similarity between content, and each connection indicates a nearest

neighbour relation. We can extend these connections in numerous ways, however for this application we simply utilized a single neighbour to illustrate strong connections, and left to right time ordering to indicate flow.

This allows us to visualize how content changes or is reported over time, and is useful for investigating specific sentences and paragraphs at a lower level. We extend this idea in the next section by aggregating these similarities at a source level to aid in news source analysis.

Figure 6.8: An example of a propagation graph generated by our system in Cytoscape which is in essence a selection of one of the clusters from the content graph using HAC, with k-nns weighted connections.

6.2.2 News Source Analysis

Another core application for analyzing news media is understanding the interrelationships between sources. In practice, we see large amounts of duplication between sources which makes it difficult to make sense of who sources from whom, and if they do, by how much. The problem is made even more complicated by the fact that news sources, unlike social media, or even other normal websites, will often refer to each other only with text (no or few direct links).

To solve this problem our system takes the idea of implicit content interrelationships and then aggregates them at a source level. We demonstrate an application of this by generating a source graph which involves the aggregation of weighted k-nn connections between sources. We then display the results by leveraging toolkits developed for an analogous task in biomedical science, specifically that of looking for interrelationships between genes in DNA. We once again use the Cytoscape graphing library and NetworkX to produce our graphs [22] [12]. By leveraging

software designed to solve an analogous problem, and combining it with ideas observed in the literature review, we are able to generate a powerful visualisation of how news sources are related to one another.

The result of this abstraction is akin to a “river” map of the media landscape, where the strength and direction of the connections between sources illustrate how they are interconnected. By ordering them over time, we can see how news “flows” from one source to another, in turn allowing users to make judgments as to how the news they read appears on their screens.

However, before we get to the results, let us try to visualize how a news story propagates in abstract, and in turn idealize what a source analysis graph should look like. **Figure 6.9** gives an idealized indication of what we have found through experience to be the generic “flow” for news media. We observe that news often originates from a single source, or a few original sources, which are often hard to find. The content is then filtered through top-level media, before being fanned out to social media.

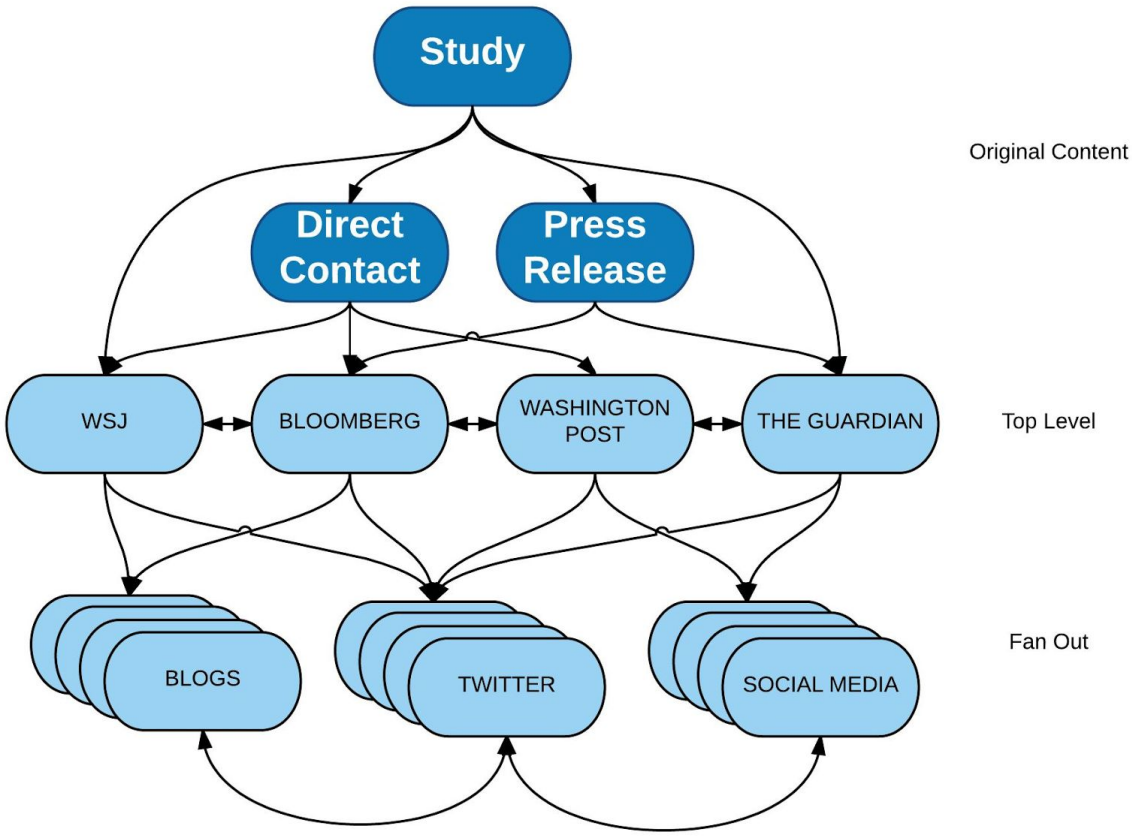

Figure 6.9: A flow diagram indicating how news flows from origin to your screen.

Our system can essentially resolve the interconnections that occur at the top level. This lets users understand which news sources are similar or different to each other, and with which sources they are most likely “aligned” with. As a result, we expect our aggregated content connections to produce something like that seen in **Figure 6.10**.

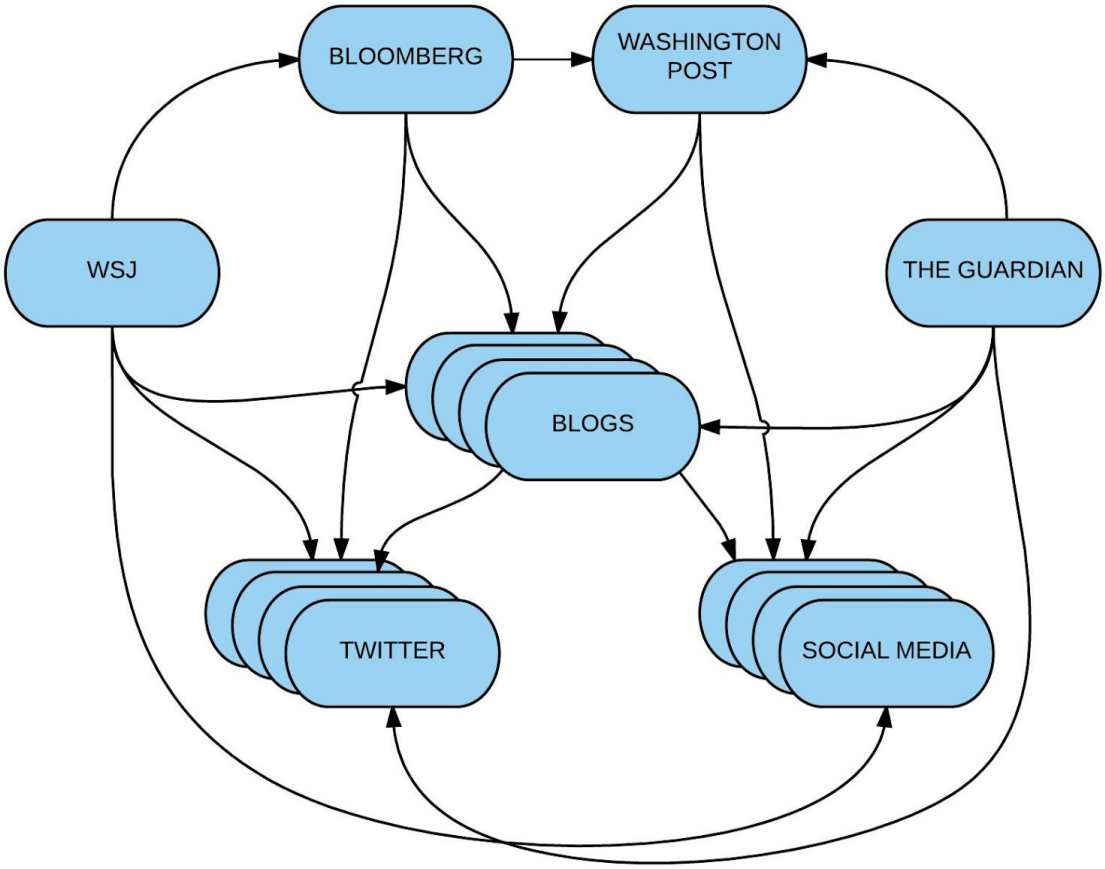

Figure 6.10: An idealized example of what a news source graph would look like.

Indeed when we do aggregate the k-nn single nearest neighbors at a source level, we approach something like that, as seen in **Figure 6.11** and **Figure 6.13**. The weight of connections can be seen in the transparency and width of the connections between sources, letting users understand how different sources interact with each other during a news event. What follows are further examples of these graphs, with captions explaining what they represent.

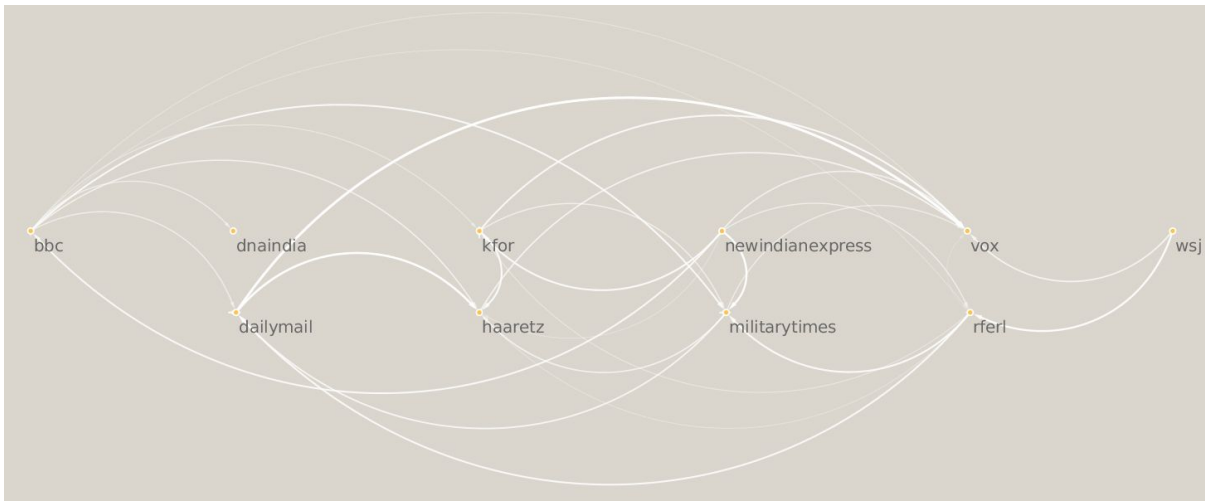

Figure 6.11: We can observe many strong connections to content from the BBC by many other sources, which might indicate a top level sourcing interaction, during the Syrian event [29].

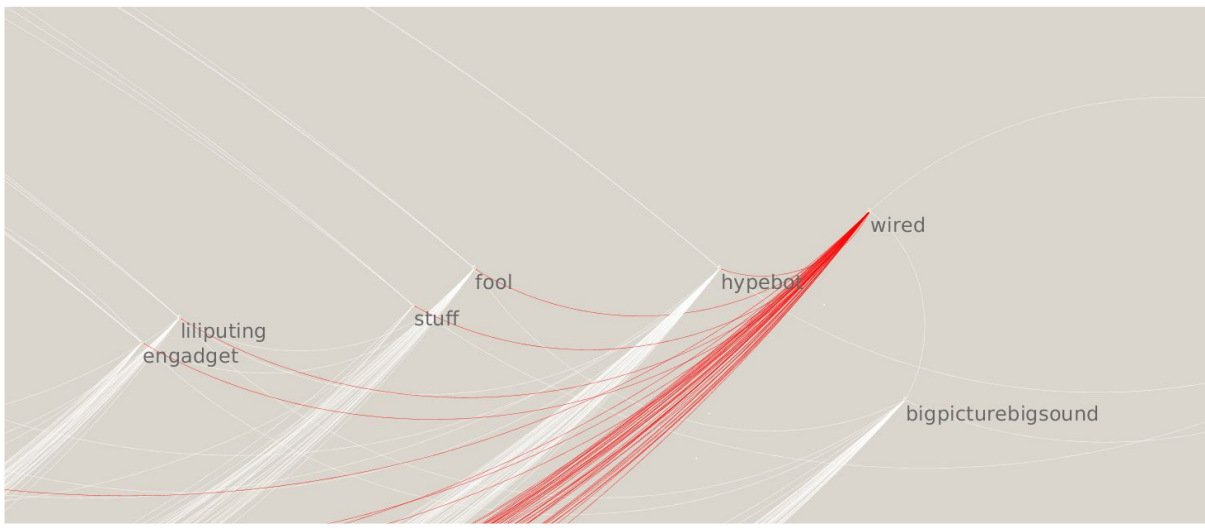

Figure 6.12: A zoomed in example on a larger event. This shows the interactions that Wired had on the Amazon unlimited news reporting story [31].

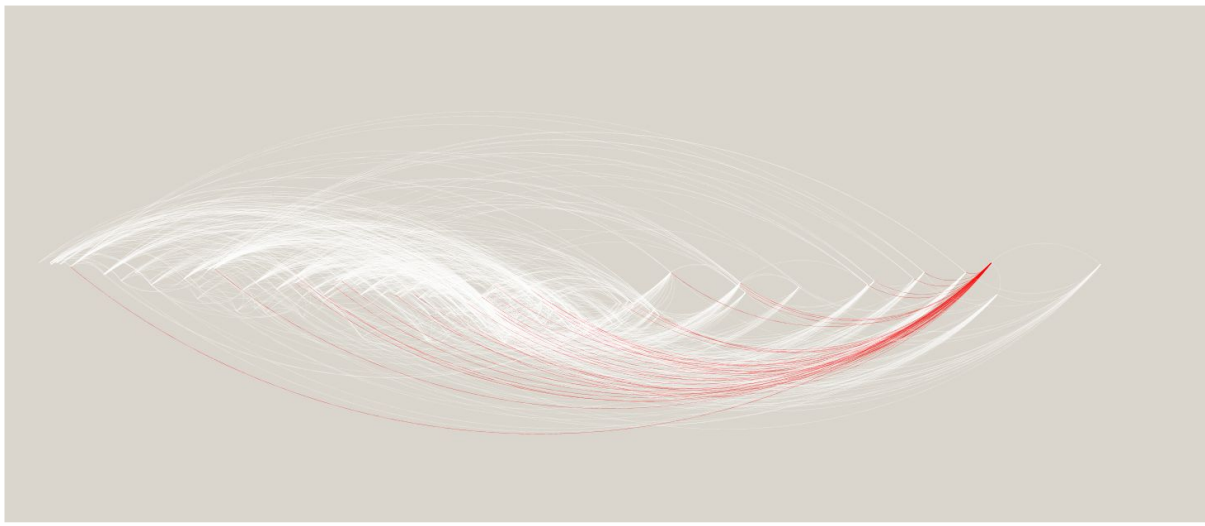

Figure 6.13: A zoomed out example of the event in **Figure 6.12**. This visualizes the content interactions that Wired had with other news sources on the Amazon unlimited news reporting story [31].

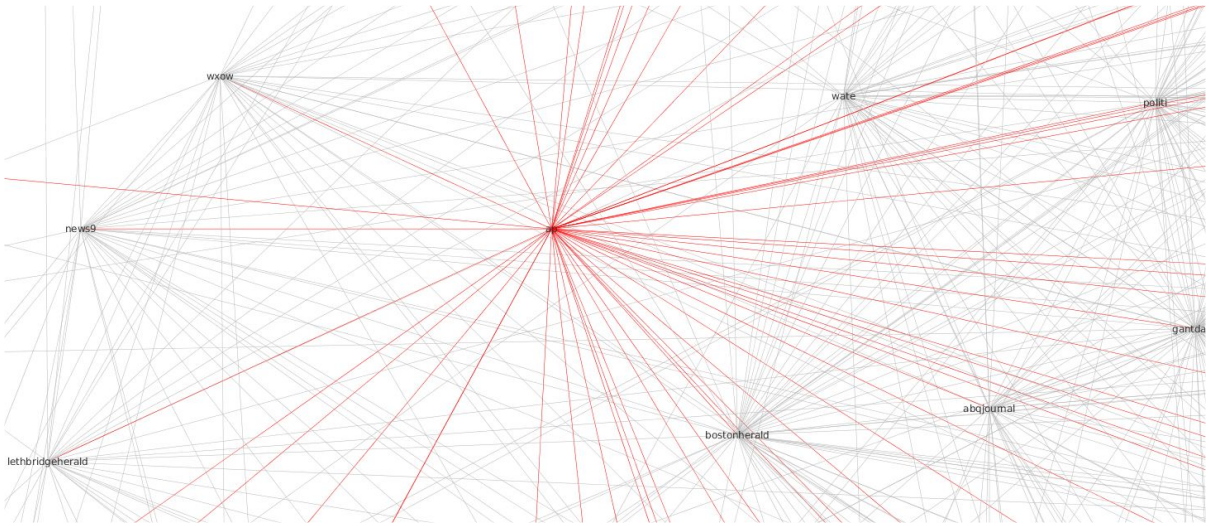

Figure 6.14: This is the selection of the AP news source, zoomed in, for the massive Hillary event [32]. We utilized a different graphing template due to the enormous number of connections.

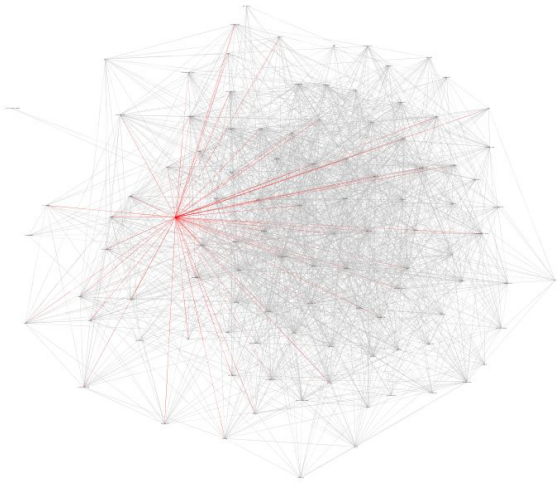

Figure 6.15: This is the selection of the AP news source for the massive Hillary event zoomed out [32]. As you can see it has interrelations across numerous sources. Indeed the tightly linked center indicates the mass of near-duplicate or duplicate sources from press agencies, with the periphery in general being more analytic reporting like the WSJ or Wired.

6.2.3 Generative Art

A perhaps novel, if not directly applicable, result of producing very large graphs is often they have a certain beauty to them. Although not a news analytic application per se, the following figures do give on an appreciation for the enormous complexity of the news media we consume every day.

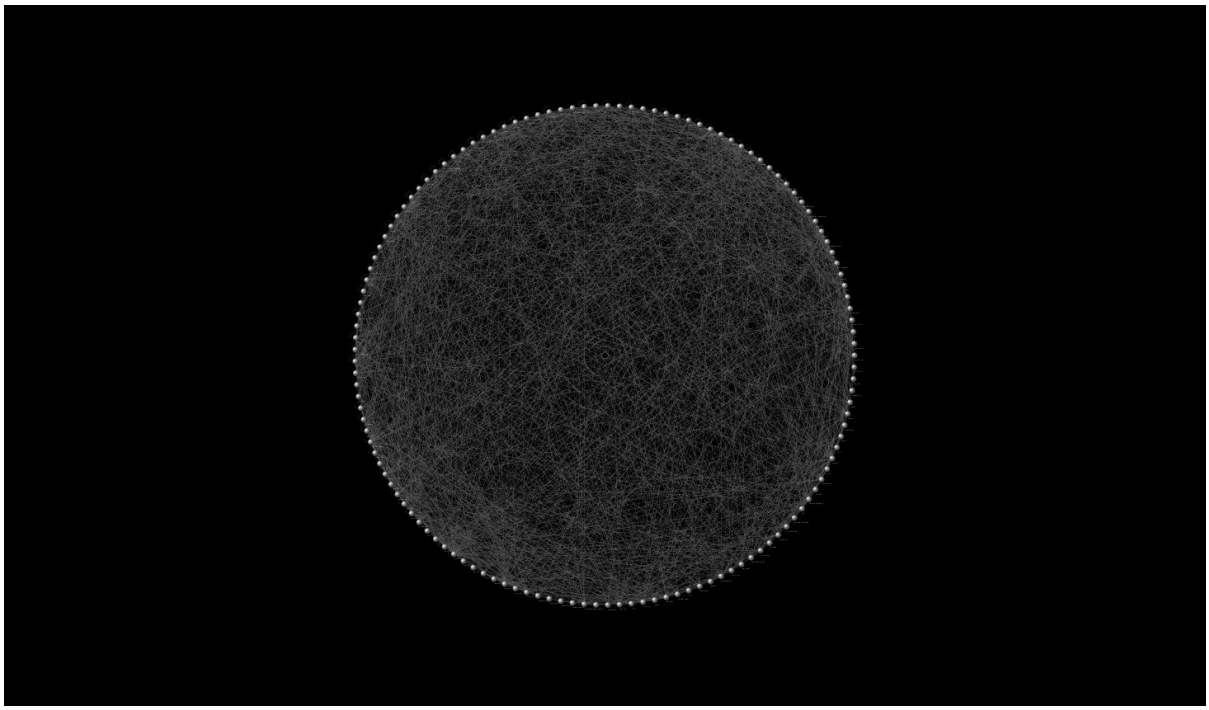

Figure 6.16: By organizing the interrelationships in a circular pattern between the news sources for the Amazon event, we approximate something that looks like the Sun [31].

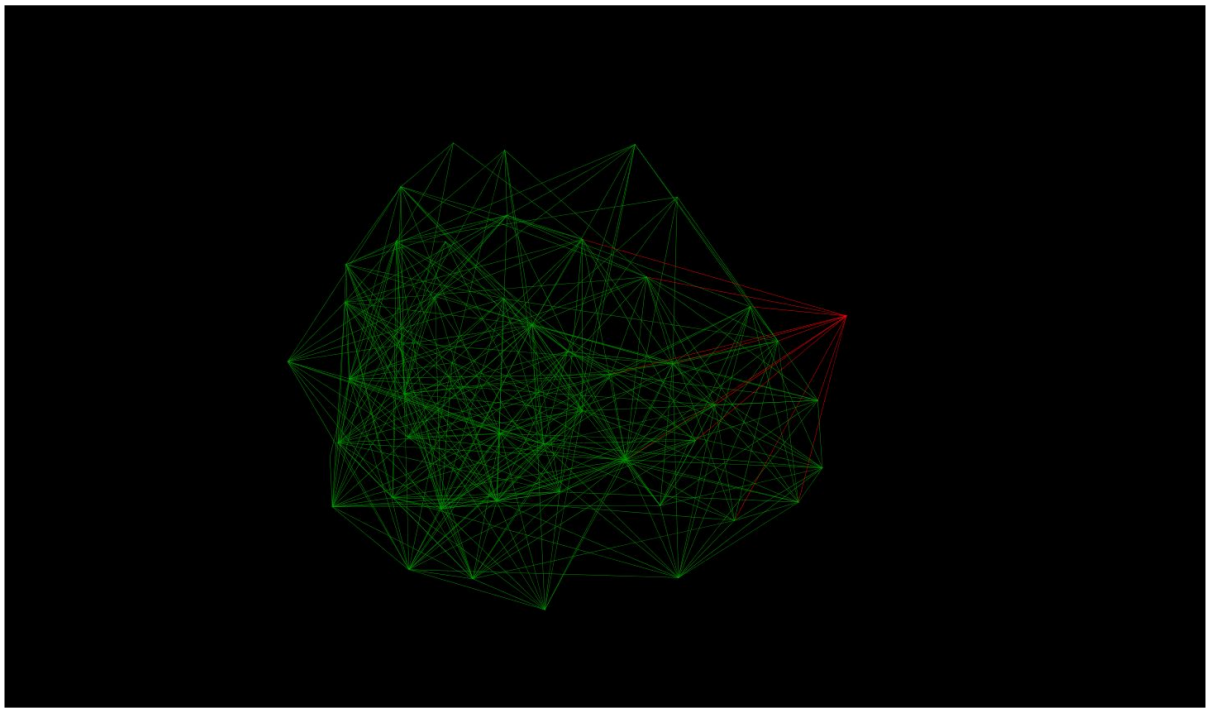

Figure 6.17: The interaction of edge analytical sources with the core reporting cluster from the WHO tax event [30].

7. Significance and Implications

The significance and implications of this proof of concept are numerous. First and foremost, the method of content tracking utilized throughout this system is highly generalizable and provides total content coverage. It can be taken at a fractal level, from sentences, to paragraphs, to documents, and is able to essentially encompass every aspect of the content graph for any clustering of documents.

This power comes from the core understanding that to be able to analyze how content flows, we require a generalizable machine learning model (paragraph vectors) that is able to semantically organize content regardless of what the content is. Indeed, our model is not just generalizable to any arbitrary content clusters, it is essentially language independent, and can be trained and used on different languages with minimal oversight or tuning.

We also show that for our proof of concept to operate, it requires nothing more to train than the clusters of documents it is meant to organize. We demonstrate this by testing the paragraph vector models on messy real world news data (the Signal 1M Article Dataset), showing that even with a limited set of noisy data, we can rapidly approach state of the art performance, especially on in-domain categories. We proved this by comparing the results of our training models on standard SemEval STS tasks against a pair of reference models which were trained on much larger and cleaner datasets. We showed that even with a small amount of content, essentially a month of training, was more than enough to produce powerful results.

We also explored the more practical concerns of such a system and demonstrated that not only was the system cost efficient, but that by forcing our system to operate on news articles within event clusters, it is also eminently scalable. All that is required is for one to replicate the proof of concept to multiple machines and then feed event clusters to each machine. This is due to the fact that event clusters, even for the English language, rarely exceeds a few thousand deduplicated articles. Thus we cap the complexity such that a single powerful machine can process every event in a reasonable amount of time. This makes the system trivially parallelizable. Hence by exploring the computational costs of both training and classification, and by limiting the size of our graphs to operate within news event clusters, we have shown that not only can our system scale but that it is also cost efficient.

By leveraging existing consumption patterns, namely browser based exploration, and utilizing workflows inspired by the biomedical community we give consumers, and investigators, the abstractions that will allow them to make up their own minds about

how news flows and is reported on every day. By making it a goal to integrate into existing state of the art eventing systems, we have shown that we can integrate with existing workflows. In turn, we provide a demonstration of a workflow that will allow the development of global insight into the events people consume every day.

Our proof of concept is the next logical step in modeling implicit content networks for news event analysis. From the early systems that looked at links, to subsequent systems that looked at parts of the content graph, we have taken the next step in producing a system that can track arbitrary content, in arbitrary networks, totally unsupervised.

8. Future Directions

Over the course of this thesis we have demonstrated a proof of concept system that aims to generate high-level views of the content graph produced by news events using paragraph vectors and clustering algorithms. We demonstrated and validated the complete workflow, from model training, to model evaluation, to index generation, to content clustering, and finally to views on the content graph with applications. Following these steps there are a series of clear future directions, and these are explored below.

For this proof of concept system our workflow involved the training of a single paragraph vector model on on a single powerful computer. While adequate for a proof of concept, if we were to scale up our system to train on the global media landscape, we would naturally have to scale out our training off one machine. A future direction would be training our paragraph vector models using a mapreduce like environment such as Apache Spark and a learning framework such as DeepDist [34]. An addendum to this would be to utilize the same mapreduce framework to trivially scale out the processing of thousands of news event clusters.

In terms of paragraph vector generation, a logical extension to our single model system would be the use of ensemble models, and determining how helpful using either multiple paragraph vector models, or indeed other specialized semantic models, would be in helping to analyze news events.

In terms of graph application extensions, a future direction would be to aggregate our source graphs over multiple events. In this thesis we have explored the complicated content graphs generated by single news events. However, we have not explored aggregations of these graphs over multiple events.

An application extension would be the generation of composite articles. Through our demonstration applications we've shown the mass of duplication and interrelations between news sources. A natural extension would be to generate a composite article, ordered by a best fit set of content, with each paragraph or sentence being connected to its near neighbours. This would allow one to rapidly consume differing viewpoints as part of a single article, without having to read them all.

Another future direction would be to evaluate how well our system can generalize to multiple languages. As our proof of concept system can be trained simply on the documents that it ingests, this should, in theory, allow one to generalize to any language with paragraphs, sentences, and words. This would prove that our system could scale up, virtually unsupervised, to the full language spectrum of online news.

A performance extension to our system would be to make the clustering steps entirely incremental to the ingestion of paragraph vectors. As our hierarchical clustering section is the slowest part of our system, and as it requires all the paragraph vectors at the start of processing, a fully incremental addition would be a powerful multiplier to the speed with which content is classified. Akin to the how the NIFTY system improved on MemeTracker by essentially making the cost of generating new graphs constant [23], an incremental paragraph vector system would rapidly reduce costs and speed up performance.

A network extension to the system would be to combine multiple event aggregations with our semantic correlations to make direct predictions about sourcing. Currently, our system acts as an aid to observe the mass of complexity that is involved in the reporting of a news event. In the future, we could use the aggregations we produce to make active predictions as to which news source sourced from which news source rather than merely showing correlations and expecting users to make the connections.

Finally, a more practically minded direction would be to transfer the leveraged workflow we have today, utilizing a myriad of tools and utilities, into a scaled up system that was integrated with existing eventing systems. This would allow millions of real users insight into the way information propagates online at a news event level, and allow them to better perform differential analysis and observe how information propagates online. Complementary to this application is the observation that news event clusters, and search queries, often return similar looking document clusters. Thus by integrating our system into ad hoc search queries, we may be able to find applications to aid research.

9. Conclusion

In this research project our goal was to evaluate if it would be possible to design a totally unsupervised proof of concept system that would reveal the total content structure of news events for the purposes of differential and network analysis. We investigated other works in the field, and noted how each aimed to achieve this by limiting themselves to only parts of the content structure (e.g. links, quotes and predicates). Our goal was therefore to improve on these systems by creating one that was both generalizable and useful at a paragraph and sentence level. We believed that by combining paragraph vectors with clustering algorithms on every paragraph and sentence in every article within a news event, we would be able to generate views on the content graph that would enable numerous useful applications for end users.

To address issues with scalability, difficulty of training, and generalizability, we decided to use completely unsupervised systems in the form of paragraph vector models. To prove their robustness, and applicability to the task, we trained and evaluated a series of models with differing best practice hyperparameters using a standard existing online news article dataset (Signal 1M Articles). We showed that these models were able to rapidly approach state of the art performance as compared to models trained on larger, cleaner datasets using multiple years of SemEval STS English tasks. We observed that despite the fact that the Signal 1M News Dataset had numerous issues involving duplications, mixed in code and poor formatting, in addition to the fact that it was smaller than the datasets used to train the reference models, we were still able to show near state of the art performance for in-domain datasets.

We then demonstrated the effectiveness of paragraph models at producing paragraph and sentence vector indices from clusters of articles found in real world news events. This illustrated how we can leverage existing news event clustering systems that users already use to develop novel applications. We built views on the content graph by clustering at both the paragraph and sentence levels using combinations of HAC and k-nns. We found that in practice k-nns were far better for exploration applications, and that HAC was better for extracting out specific content for analysis.

We used these content graph views to develop useful applications to help users make sense of how news sources and content interrelate. In the process of development we leveraged numerous existing analysis libraries and mirrored existing news consumption workflows to rapidly demonstrate how our system could

be used. We created two classes of applications, one for exploration analysis, and the other for graph analysis. We demonstrated differential analysis through exploration with k-nns in multiple forms. We then showed how by aggregating k-nns and HAC we were able to develop novel visualisations into the interactions between news sources reporting on the same event.

We found that the proof of concept that we created was both scalable and effective at generating views on the content graph of news events. We demonstrated the cost and performance trade offs of different platform providers for the purposes of training our core paragraph vector models. We found that bare metal hardware (CenturyLink Cloud) out performed both Google Compute Engine and Amazon EC2. We hypothesize that this is due to the fact that bare metal servers retain full control over their resources, and hence avoid CPU and memory context switching which cause performance to degrade.

Our work has generated a variety of significant implications. The proof of concept we developed is the next logical step in modelling implicit content networks. It is able to self-train simply on the news events it consumes, and is able to analyze, at a fractal level, from sentence, to paragraph, to document, how content flows across news sources. The system is also trivially parallelizable by utilizing the news event as a core abstraction, in addition to being cost efficient. In the course of our work we have taken the next step in news event analysis by demonstrating a system that can train, classify and produce novel applications on the full content graph of news events in a totally unsupervised fashion.

In regards to future directions, there are many clear paths for exploration. We can scale up our training from the standard Signal 1M News Article Dataset to global media streams by utilizing deep learning cluster platforms, such as DeepDist, and observe how they perform. We can scale out our event classification by utilizing mapreduce abstractions such as Apache Spark, and use the thousands of graphs it produces to observe mult-event relationships. We can evaluate how ensemble models can aid the performance of semantic content organization. We can move our system from using a global HAC model that dominates the computation of processing an event, to an entirely incremental system that keeps computational costs constant. We can extend our model to multiple languages, and observe how they perform. We can observe if multi-event aggregation graphs will allow us to make predictive statements about content sourcing. Finally, we can integrate our proof of concept with an existing news event clustering system to give non-technical users the tools they need for differential and graph analysis of news events.

10. References

10.1 Academic

[1]

Adar, E. and Adamic, L.A. 2005. Tracking information epidemics in blogspace. *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, 2005. Proceedings* (Compiegne, France, Sep. 2005), 207–214.

[2]

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G. and Wiebe, J. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (Dublin, Ireland, 2014), 81–91.

[3]

Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A. and Guo, W. 2013. *SEM 2013 shared task: Semantic Textual Similarity. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task* (Atlanta, Georgia, 2013), 32–43.

[4]

Agirre, E., Diab, M., Cer, D. and Gonzalez-Agirre, A. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)* (Montreal, Canada, 2012), 385–393.

[5]

Agirre, E., Gonzalez-Agirre, A., Lopez-Gazpio, I., Maritxalar, M., Rigau, G. and Uria, L. 2016. SemEval-2016 Task 2: Interpretable Semantic Textual Similarity. *Proceedings of SemEval-2016* (San Diego, California, 2016), 512–524.

[6]

Agirrea, E., Baneab, C., Cardiec, C., Cerd, D., Diabe, M., Gonzalez-Agirrea, A., Guof, W., Lopez-Gazpioa, I., Maritxalara, M., Mihalceab, R., Rigau, G., Uria, L. and Wiebe, J. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. *Proceedings of the 9th*

International Workshop on Semantic Evaluation (SemEval 2015) (Denver, Colorado, 2015), 252–263.

[7]

Berkhin, P. 2006. A survey of clustering data mining techniques. *Grouping multidimensional data*. Springer. 25–71.

[8]

Bhatia, N. 2010. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security, Vol. 8, No. 2* (2010).

[9]

Colavizza, G., Infelise, M. and Kaplan, F. 2015. Mapping the Early Modern News Flow: An Enquiry by Robust Text Reuse Detection. *Social Informatics*. L.M. Aiello and D. McFarland, eds. Springer International Publishing. 244–253.

[10]

Corney, D., Albakour, D., Martinez-Alvarez, M. and Moussa, S. 2016. What do a Million News Articles Look like? *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016)* (Padua, Italy, Mar. 2016), 42–47.

[11]

Gomez Rodriguez, M., Leskovec, J. and Krause, A. 2010. Inferring Networks of Diffusion and Influence. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington, DC, USA, 2010), 1019–1028.

[12]

Hagberg, A.A., Schult, D.A. and Swart, P.J. 2008. Exploring network structure, dynamics, and function using NetworkX. *Proceedings of the 7th Python in Science Conference (SciPy2008)* (Pasadena, CA USA, Aug. 2008), 11–15.

[13]

Huang, A. 2008. Similarity measures for text document clustering. *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)* (Christchurch, New Zealand, 2008), 49–56.

[14]

Lau, J.H. and Baldwin, T. 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *Proceedings of the*

1st Workshop on Representation Learning for NLP (Berlin, Germany, Aug. 2016), 78–86.

[15]

Le, Q. and Mikolov, T. 2014. Distributed Representations of Sentences and Documents. *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)* (Beijing, China, 2014), 1188–1196.

[16]

Leskovec, J., Backstrom, L. and Kleinberg, J. 2009. Meme-tracking and the Dynamics of the News Cycle. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France, 2009), 497–506.

[17]

Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at the International Conference on Learning Representations, 2013* (Scottsdale, USA, 2013).

[18]

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* (2013), 3111–3119.

[19]

Myers, S.A., Zhu, C. and Leskovec, J. 2012. Information diffusion and external influence in networks. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (Beijing, China, 2012), 33.

[20]

Pedregosa, F. et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 12, (2011), 2825–2830.

[21]

Řehůřek, R. and Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (Valletta, Malta, May 2010), 45–50.

[22]

Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B. and Ideker, T. 2003. Cytoscape: a software

environment for integrated models of biomolecular interaction networks. *Genome research*. 13, 11 (2003), 2498–2504.

[23]

Suen, C., Huang, S., Eksombatchai, C., Sosic, R. and Leskovec, J. 2013. NIFTY: a system for large scale information flow tracking and clustering. *Proceedings of the 22nd international conference on World Wide Web* (Rio de Janeiro, Brazil, 2013), 1237–1248.

[24]

Vakulenko, S., Göbel, M., Scharl, A. and Nixon, L. 2016. Visualising the Propagation of News on the Web. *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016)* (Padua, Italy, Mar. 2016), 60–62.

[25]

Yang, J. and Leskovec, J. 2010. Modeling Information Diffusion in Implicit Networks. *2010 IEEE 10th International Conference on Data Mining (ICDM)* (Washington, DC, USA, Dec. 2010), 599–608.

[26]

Zahn, C.T. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on computers*. 100, 1 (1971), 68–86.

[27]

Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J. and McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. *Association for Computational Linguistics (ACL) System Demonstrations* (2014), 55–60.

10.2 News Events

[28]

Google News Coverage of the “EgyptAir Crash” story, *Google News*, May 2016. Retrieved October 30, 2016, from Google: <https://www.google.com.au/search?q=Debris+from+EgyptAir+crash+found&hl=en&authuser=0&source=lnt&tbs=cdr%3A1%2Ccd_min%3A21%2F05%2F2016%2Ccd_max%3A21%2F05%2F2016&tbm=nws>

[29]

Google News Coverage of the “Syrian Ceasefire” story, *Google News*, September 2016. Retrieved October 30, 2016, from Google: <https://www.google.com.au/search?q=syrian+ceasefire&hl=en&authuser=0&biw=1855&bih=953&source=lnt&tbs=cdr%3A1%2Ccd_min%3A8%2F09%2F2016%2Ccd_max%3A12%2F09%2F2016&tbm=nws>

[30]

Event Registry Coverage of the “Tax Soda To Fight Obesity, WHO Urges Nations Around The Globe” story, *Event Registry*, Geneva, Switzerland, Tuesday, 11 October 2016. Retrieved October 30, 2016, from Event Registry: <http://eventregistry.org/event/5151232?lang=&tab=articles>

[31]

Event Registry Coverage of the “Amazon launches voice-driven on-demand music service” story. *Event Registry*, Wednesday, 12 Oct 2016. Retrieved October 30, 2016, from Event Registry: <http://eventregistry.org/event/5156472?lang=&tab=articles>

[32]

Event Registry Coverage of the “Hillary Clinton, Mocking and Taunting, Turns the Tormentor” story. *Event Registry*, Las Vegas, United States, Wednesday, 19 Oct 2016. Retrieved October 30, 2016, from Event Registry: <http://eventregistry.org/event/5178877?lang=&tab=articles>

10 3 Software

[33]

Anaconda Software Distribution. *Continuum Analytics*, 2016, Version: 2-2.4.0. Retrieved October 30, 2016, from Continuum Analytics: <https://continuum.io>

[34]

Neumann D., DeepDist, *DeepDist*, 2016: Retrieved October 30, 2016, from DeepDist: <http://deepdist.com>

10.4 News Event Clustering Platforms

[35]

Google News, *Google*, 2016. Retrieved October 30, 2016, from Google News: <http://news.google.com>

[36]

Event Registry, *Event Registry*, 2016. Retrieved October 30, 2016, from Event Registry: <http://eventregistry.org>

10.5 Articles

[37]

Cozens, C., New York Times: we were wrong on Iraq, *The Guardian*, May 26, 2004. Retrieved October 30, 2016, from The Guardian: <https://www.theguardian.com/media/2004/may/26/pressandpublishing.usnews>

[38]

Keegan, J., Blue Feed, Red Feed: 2016, *The Wall Street Journal*, 18 May 2016. Retrieved October 30, 2016, from The Wall Street Journal: <http://graphics.wsj.com/blue-feed-red-feed>

[39]

Chen, A., The Agency, *The New York TImes*, June 2, 2015. Retrieved October 30, 2016, from The New York Times: <http://www.nytimes.com/2015/06/07/magazine/the-agency.html>

[40]

Nakashima, E., U.S. government officially accuses Russia of hacking campaign to interfere with elections, *The Washington Post*, October 7, 2016. Retrieved October 30, 2016, from The Washington Post: <https://www.washingtonpost.com/world/national-security/us-government-officially-accuses-russia-of-hacking-campaign-to-influence-elections/2016/10/07/4e0b9654-8cbf-11e6-875e-2c1bfe943b66_story.html>

[41]

Connolly, K., Angela Merkel: internet search engines are 'distorting perception', *The Guardian*, October 28, 2016. Retrieved October 30, 2016, from The Guardian: <https://www.theguardian.com/world/2016/oct/27/angela-merkel-internet-search-engines-are-distorting-our-perception>

10.6 Platforms

[42]

Google Compute Engine, *Google Cloud Platform*, 2016. Retrieved October 30, 2016, from Google Cloud Platform: <https://cloud.google.com/compute>

[43]

Amazon EC2, *Amazon AWS*, 2016. Retrieved October 30, 2016, from Amazon AWS: <https://aws.amazon.com/ec2>

[44]

CenturyLink Cloud Bare Metal, *CenturyLink Cloud*, 2016. Retrieved October 30, 2016, from CenturyLink Cloud: <https://www.ctl.io/bare-metal>

11. Appendix

11.1 Training Model Hyper Parameters

Parameters	ap_dbow	enwiki_dbow
sampling_threshold	1.00E-05	1.00E-05
hs	0	0
dm	0	0
negative_size	5	5
dbow_words	1	1
vector_size	300	300
window_size	15	15
min_count	10	20
train_epoch	30	20
alph	0.025	0.025
min_alpha	0.0001	0.0001

11.2 STS2012 Results

	ap	intersect_ap	wiki	intersect_wiki	apnews_dbow	enwiki_dbow
MSRpar	0.42	0.42	0.40	0.40	0.42	0.48
SMTeuroparl	0.42	0.41	0.42	0.42	0.48	0.49
SMTnews	0.26	0.25	0.29	0.29	0.33	0.36
MSRvid	0.69	0.69	0.72	0.71	0.75	0.77
OnWN	0.51	0.52	0.54	0.54	0.64	0.64

11.3 STS2013 Results

	ap	intersect_ap	wiki	intersect_wiki	apnews_dbow	enwiki_dbow
SMT	0.31	0.31	0.31	0.32	0.35	0.37
OnWN	0.57	0.57	0.59	0.60	0.72	0.71
headlines	0.63	0.63	0.63	0.64	0.70	0.67
FNWN	0.28	0.27	0.28	0.26	0.38	0.40

11.4 STS2014 Results

	ap	intersect_ap	wiki	intersect_wiki	apnews_dbow	enwiki_dbow
OnWN	0.61	0.61	0.62	0.63	0.75	0.73
headlines	0.57	0.58	0.58	0.59	0.63	0.62
deft-forum	0.30	0.31	0.32	0.33	0.39	0.33
tweet-news	0.47	0.47	0.53	0.52	0.67	0.64
deft-news	0.54	0.54	0.54	0.55	0.67	0.66
images	0.61	0.61	0.64	0.64	0.73	0.73

11.5 STS2015 Results

	ap	intersect_ap	wiki	intersect_wiki	apnews_dbow	enwiki_dbow
answers-forums	0.43	0.43	0.46	0.46	0.61	0.59
answers-students	0.51	0.51	0.54	0.54	0.69	0.65
belief	0.37	0.37	0.43	0.42	0.61	0.58
headlines	0.69	0.69	0.70	0.70	0.75	0.74
images	0.68	0.67	0.69	0.70	0.79	0.80

11.6 STS2016 Results

	ap	intersect_ap	wiki	intersect_wiki	apnews_dbow	enwiki_dbow
answer-answer	0.41	0.42	0.40	0.39	0.51	0.43
headlines	0.64	0.64	0.67	0.66	0.72	0.69
plagiarism	0.68	0.69	0.70	0.69	0.78	0.78
postediting	0.50	0.50	0.51	0.51	0.69	0.67
question-question	0.62	0.60	0.62	0.63	0.65	0.69