Modelling Implicit Content Networks to Track Information Propagation across Media Sources to Analyze News Events

Anirudh Joshi
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
anirudhj@student.unimelb.edu.au

Richard O. Sinnott
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
rsinnott@unimelb.edu.au

***Abstract*—With the rise of the Internet as the premier news source for billions of people around the world, the propagation of news media online now influences many critical decisions made by society every day. Fake news is now a mainstream concern. In the context of news propagation, recent works in media analysis largely focus on extracting clusters, news events, stories or tracking links or conserved sentences at aggregate levels between sources. However, the insight provided by these approaches is limited for analysis and context for end users. To tackle this, we present an approach to model implicit content networks at a semantic level that is inherent within news event clusters as seen by users on a daily basis through the generation of semantic content indexes. The approach is based on an end-to-end unsupervised machine learning system trained on real-life news data that combine together with algorithms to generate useful contextual views of the sources and the inter-relationships of news events. We illustrate how the approach is able to track conserved semantic context through the use of a combination of machine learning techniques, including document vectors, k-nearest neighbors and the use of hierarchical agglomerative clustering. We demonstrate the system by training semantic vector models on realistic real-world data taken from the Signal News dataset. We quantitatively evaluate the performance against existing state of the art systems to demonstrate the end-to-end capability. We then qualitatively demonstrate the usefulness of a news event centered semantic content index graph for end-user applications. This is evaluated with respect to the goal of generating rich contextual interconnections and providing differential background on how news media sources report, parrot and position information on ostensibly identical news events.**

I. Introduction

Internet news has become central to society in recent years, and is now a critical component within global soft power campaigns from elections, to advertising, to government-sponsored propaganda operations. Due to the speed of online news as it currently stands, and the Internet's ascendance as the primary information delivery mechanism worldwide, news sourced from it is often the sole source of truth for billions of people around the world. This shapes the reality of vast swathes of the global population and bends what they believe to be true or false every day.

Due to its enormous scale, and the limited time available for people to interrogate and cross-correlate news stories across sources, it is basically impossible for the average citizen to understand how the news they consume everyday originates, relates between sources (differential reporting) and morphs across time and space without systematic investigative research. To deal with this information deluge, a variety of approaches, both commercial and academic have been put forward. On the commercial side, many stories that people read every day are often sourced from event aggregation systems such as Google News or Event Registry. These systems report and display clustered news events in summarized formats, often delivering thousands of articles per day [1] [2]. On the academic side, systems have been built that demonstrate various methods of tracking information propagation online, mostly through the use of tracking conserved content markers on parts of an article (e.g. links, quotes, relational predicates) [3] [4] [5] [6] [7] [8] [9] [10]. However, for actual analysis of a news event, these systems rely on a user's ability to cross-correlate and contextualize the story themselves, i.e. by physically reading each and every article. This gap is what we attempt to address.

II. Related Work

Most commercial aggregation systems, like Google News, simply list top articles with frequency counts [1]. However, there can often be 10s to 1000s of articles concerning an event, most with highly duplicated information mixed in with novel insights. It is essential to be able to tease out the interactions between news sources and visualize them for differential analysis by end users. The issue with most existing systems, however, 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, including how it is sourced and the order in which it was published. Using such aggregators requires 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

978-1-5386-9156-4/18/$31.00 ©2018 Crown
DOI 10.1109/eScience.2018.00136

everything themselves, merely clustering together documents but giving no sense of how content itself is interrelated [2].

Academic systems also exhibit this problem, often focusing on the complete content graph and high level aggregated views, at the expense of aiding individual news event analysis. Here we focus entirely on content focused modelling approaches, which do not track specific entities through time, or attempt to model the underlying dynamic knowledge graph between entities [11] [12]. There are numerous approaches to analyzing information propagation at a content level within online news events. They can be broadly organized into two overarching methodologies, one being the link diffusion method and the other being the content diffusion method.

Link diffusion systems use explicit links 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 [3]. They used analogies to epidemiological models to generate a directed acyclic graph which could be ordered by time to illustrate the flow of information. 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 major 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.

The next generation of information tracking systems focuses on monitoring the propagation of conserved content (repeated strings or quotes) contained within articles. Yang and Leskovec (2010) utilized a different approach from pure network tracking by avoiding the network topology altogether [4]. 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. The MemeTracker system by Leskovec et al. (2009) offered an extension of the link-based approach to quotes, which can analogously be seen as links which mutate very little from source to source [5], i.e. 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 the chain). They achieved this by identifying what they call phrase clusters, that is groups of strings that exhibit substantial textual similarity. As partitioning this graph is an NP-hard problem, they utilized heuristics. To partition the graph they simply deleted low weight connections until the phrase cluster graph decomposed into a series of disjoint mutational quote chains, with shorter quotes feeding into a single long quotation. However the system was extremely computationally expensive and tracking all content interconnections between news sources at a global level would not be possible.

The successor to MemeTracker, dubbed NIFTY by Suen et al. (2013), attempted to mitigate the extreme computational costs of tracking incrementally changing quotations by utilizing an incremental meme-clustering algorithm [6]. It uses the same underlying insight of connecting each quotation to a longer quotation that is a close subsequence as part of a large directed acyclic graph, with long strings being the cluster assignments and the longest versions of the quotations. To improve the performance they attempt to keep the graph size essentially constant, by freezing old clusters that haven't changed for some time, and constructing new ones when they don't fit into existing clusters. The goal of both of the previous 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 these systems is that, just like the link-based systems, they only rely on a portion of the content served by news organizations about an event, namely things enclosed in quotes. As such, they are more targeted towards analyzing event bursts, rather than observing propagation between sources or comparative analysis.

Colavizza et al. (2015) tried a different attack to analyze information propagation by focusing on text reuse in early modern newspapers [7]. Their insight was understanding that barring explicit links, the vast majority of early newspaper information flows would essentially be primarily conserved content, i.e. directly repeated text reporting facts between papers over time. They analyzed gazettes from the year 1648 using OCR (optical character recognition), string kernels (as similarity measures) and local text alignment (to determine overlaps) 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. 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 the 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 levels as it tolerates noisy substring overlaps.

Another approach to these challenges was attempted in the work of Vakulenko et al. (2016) [8]. 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. Despite the power of this system, it has numerous shortcomings. 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 is unable to track more complicated content relationships. It is also dependent upon synsets in Wordnet [13].

III. Approach and Implementation

As identified, the vast majority of systems only focus on a single or small aspect of news articles, namely links or highly conserved content. Hitherto, there has not been significant research into utilizing unsupervised semantic systems to rapidly organize, at a paragraph and sentence level, the mass of semantic interrelations found within news events, and especially those that focus on mainstream online news. In this work, we define news events to be clusters of articles based on document similarity and time of co-occurrence.

The aim of our system is to fundamentally augment the abilities of researchers and typical news readers, not through the generation of unfocused content graphs like other approaches, or high level summarizations of news events, but through the creation of useful views on the content at a semantic (implicit) level and to deal with any arbitrary news event. To achieve this, we use a simplifying assumption to keep computational costs bounded: namely we focus our analysis on what users would find most useful through the analysis of specific news events, rather than attempting to create a global news index. This makes the generation of the content semantic index both tractable and useful, and greatly reduces the number of false positive connections that would be seen in a global graph.

Rather than reinvent existing clustering systems, we work in the frame of an actual user consuming news from event aggregation systems. Specifically, we leverage such systems to increase the reliability of our semantic relation algorithms, by focusing them within news events thereby reducing false positive connections. This also caps our complexity by breaking the news into discrete chunks that can be processed in constant time, and subsequently boost the power of semantic models by ensuring that cross connections are those that are already found to be highly relevant.

This paper focuses on tracking contemporaneous events, rather than retrospective tracking. This is representative since 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.

To address these issues we propose a system built on unsupervised paragraph vector models. These models are used to generate vector indexes for all the paragraphs and sentences within a news event. We train these models by utilizing just the news data itself. We then use standard clustering and graphing workflows to generate views on the content graph for demonstration applications that 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 inter-relationships at an aggregate level. We also verify the power of our entirely unsupervised self-training system against a set of reference models on a series of standardized semantic evaluation tasks. Ultimately the goal is to provide users with global insight into the news events they read every day.

A. Data Analysis Stack

We utilized the Anaconda scientific platform with both Python 2.7 (training/generating vectors) and 3.5 (parsing/manipulating articles) versions [14]. In addition, we also used the sci-kit learn library for the standard implementation of clustering and distance algorithms, NetworkX was used for the generation of content networks, and gensim for the training and generation of paragraph vectors [15] [16] [17]. Stanford CoreNLP was also used in the preprocessing of the article dataset into tokens [18]. The generated graph visualizations use the Cytoscape graphing library [19].

B. Datasets

1) The Signal Media 1-Million Article Training Dataset: The system uses the Signal 1-Million News Article dataset for model training purposes [20]. We chose this dataset as it was a standardized, reasonably large crawl of news articles that accurately represented the type of data we wish to analyze. 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 [20]. It comprises nearly 93,000 real-world news sources and differs from the cleaner single source datasets used to train reference models [21]. Thus it gives us an accurate representation of the semantic performance that could be achieved by a system and especially those that are to handle massive streams of real-world news data.

As the crawl is from thousands of real online sources, the dataset includes liberal amounts of duplication, noisy data, incorrect language articles and code snippets that cannot be easily parsed. As such, this provides an extremely realistic dataset on which to train unsupervised models. In essence, it allows one to observe how robust paragraph vector models are on what is essentially a continuous stream of real-world news data.

2) Analyzed Events: To illustrate the applications of our approach we take events from two news event clustering systems: Google News and Event Registry [1] [2]. These sources provide us with a real-world set of news articles that have been aggregated into news event clusters for actual users. 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) [22]. This event concerned the ceasefire brokered between the main parties involved in the Syrian Civil War (the Syrian event). We chose 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 provides 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 concerns a recommendation by the World Health Organization (WHO) that urged countries to tax high sugar drinks to alleviate obesity and health issues (the WHO event) [23]. This event contains a couple of hundred articles. The second news event cluster from Event Registry concerned the release of Amazon's unlimited music streaming service (the Amazon event) [24]. This event contains a couple of hundred articles. The third news event cluster from Event Registry concerns the performance of U.S. presidential nominee Hillary Clinton at the third U.S. presidential debate against Donald Trump (the Hillary event) [25]. This event comprised a couple of thousand articles.

C. Semantic Models

1) Word Vectors: For the implementation, we used paragraph (document) vectors generated by doc2vec from the gensim python library [17]. However, before introducing the relevant concepts and details of 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 based on a shallow neural network from Mikolov et al. (2013) [26]. The goal of the word vector model is to generate high dimensional vectors, often on the order of 100-300 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 [27]. 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 [27]. 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 [26]. The power of this approach is the fact that it is totally unsupervised.

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 [28].

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 to that of word vectors. The method of optimization is the same as previous models, e.g. stochastic gradient descent on the predictive output with back propagation for weighted embeddings of the input and output [28].

There are two main methods of training: distributed memory (DM) and distributed bag of words (DBOW). DBOW is analogous to the skip-gram model with negative sampling. In practice DBOW is found to exhibit higher performance and be a more robust method, hence it is adopted here [21].

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) [21]. These are used as benchmarks for comparison against our trained models on standard semantic evaluation tasks. These reference models are referred to as ap_dbow, and enwiki_dbow, as they are in the original paper. We then use these best practice hyperparameters to train paragraph vectors on noisy multi-source real-world online news article data (the Signal 1M News Article Dataset). These models will be referred to as ap, intersect_ap, wiki, and intersect_wiki.

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 are able to see if utilizing pre-trained word vectors increase 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 solution to 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 noisy 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) Model Evaluation: For model evaluation we use the English Semantic Textual Similarity (STS) tasks from the *SEM and SemEval for the years 2012-2016 [29] [30] [31] [32] [33].

Concretely these tasks are all simply sets of sentence pairs, across multiple domains, that are manually annotated by semantic similarity with a score of 5 being that the two sentences are perfectly semantically similar, e.g. "Many statements made online are false", and "There are often statements made online that are false", with scores falling away to 0 as the sentences differ.

STS allows us 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 generated ratings from 0 to 5, with 0 meaning the two sentences are not remotely semantically similar, and

5 meaning they are perfectly semantically similar, are then compared to human ratings using a correlation measure.

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 qualitatively evaluate the graphs and clusters generated in the applications stage to generate useful abstractions or views of 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 supports the organization of sentences and paragraphs. In turn, this allows one to display their interrelationships so that rapid analysis of a news event at a content level can occur.

D. System Algorithms

1) Cosine Similarity: For the system implementation, we utilize the standard measure of similarity between vectors known as cosine similarity [34]. 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 based on their multiplied magnitudes.

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 involve the calculation of all pairwise distances between all pairs within a list of vectors using some distance metric [35], which in our case is cosine similarity, and where the vectors are our sentences and paragraphs. We then order the distances of a particular vector's neighbors by their similarity and extract those closest to it using some thresholding value, whether that is a similarity threshold or simply the number of neighbors required.

We utilized a single nearest-neighbor model for the creation of propagation and source analysis graphs. We used a multi-neighbor 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.

3) Hierarchical Agglomerative Clustering: We utilized hierarchical agglomerative clustering (HAC) with cosine distances to generate semantically similar content clusters [36]. We used these clusters as seeds for the generation of a content propagation graph in conjunction with k-NNs. HAC 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 a single global cluster exists.

The reason for choosing HAC over other methods was due to the fact that it accurately represents how the underlying content is structured. 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 is that it does not require the definition of the number of clusters within an index. This allows it to scale to any number of sentences, paragraphs or articles by using a linkage metric with an inconsistency criterion.

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 [36]. 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 linkage similarity.

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 represents the underlying sentences and paragraphs [36]. This is as opposed to single link clustering which involves chaining sentences that were not well separated.

To generate clusters it is necessary to cut the dendrogram using some adaptive metric. We utilize flat clustering on our linkage structure with the default inconsistency method using sci-kit learn's fcluster [15]. The inconsistency method operates by cutting clusters based upon the dynamics of the linkage structure [37]. 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. This method provides distinct and tight clusters on top of which it is possible to generate useful content propagation graphs.

4) News Article Pre-Processing: 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 reduce the number of duplicates. The articles were then tokenized by Stanford CoreNLP [18] at a sentence and paragraph level.

E. Summary

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

1) Model Training:

- Acquire a batch of representative news articles, in this case from the Signal 1M Article Dataset;
- convert them into a series of paragraphs;
- tokenize and lower case the paragraphs with Stanford CoreNLP;
- build a vocabulary of words within the paragraphs, intersecting word vectors if required;
- train a paragraph vector model with best practice hyper-parameters using gensim;
- shuffle paragraphs randomly on every iteration to improve performance, and
- measure the performance using standard semantic evaluation tasks.

2) Content Index Generation:

- 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 workflows;
- take each article in the event and split it up into a list of paragraphs or sentences, depending on the type of content to be analyzed;
- take a trained paragraph vector model and convert each paragraph or sentence into a paragraph vector, and finally
- use these vectors as paragraph or sentence indexes on top of which we can apply clustering and graphs to generate a view of the content graph.

3) Generating Views on the Content Graph: By integrating all of 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 explore differential analysis, e.g. to observe how a piece of content propagates over time, and observe the interrelationships between different news sources.

IV. Results

A. Semantic Results

1) Semantic Results Discussion: In general, the results of the STS evaluations are clear, noisy real-world news data models, especially if the dataset is smaller than the reference models, will under-perform on much larger single source datasets across all domains (see Figure 1 for a representative year, we omit the others for brevity). 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 [21]. This is in comparison to our Signal 1M article dataset which comprises only a single month of noisy 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

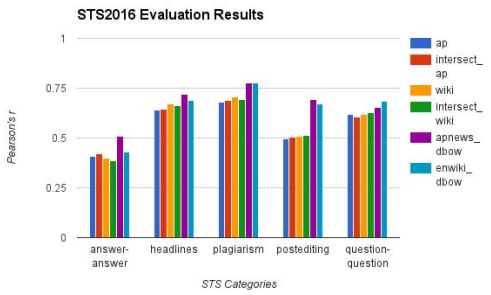

Fig. 1. A graph representing the STS2016 evaluation task comparing the Pearson's R-correlation across multiple domains and models.

TABLE I
Performance across all SemEval years for training models

Model	ap	intersect_ap	wiki	intersect_wiki
No. Times best at News Headlines	0	0	2	3
No. Times best at Any	1	2	9	13

datasets, but to show that, despite how badly formatted or noisy the 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 demonstrates that we should be able to scale up our system on a constant stream of news events to both train our system as well as to generate useful applications that are completely unsupervised.

Across the evaluation, we noticed that in general, our models perform surprisingly well across many domains, and specifically, they work well in domains of particular relevance to news, e.g. headlines, image caption and plagiarism comparisons (STS2016, STS2015, STS2014). 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 hyper-parameter models for our in-domain tasks. We compare the relative performance between our trained models in the next section.

2) Best Trained Model: We calculated the best overall performing trained model by simply counting which trained models performed best across all domains (e.g. max score) with a separate class focusing on the in-domain dataset of news headlines only. As can be seen in Table I the wiki hyper-parameters appear to give better overall results, both across domains and specifically on the 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 hyper-parameters outperform the ap hyper-parameters 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 hyper-parameter which only allows words that appear a certain number of

TABLE II
PERFORMANCE ACROSS ALL SEMEVAL YEARS FOR REFERENCE MODELS

Model	apnews_dbow	enwiki_dbow
No. Times best at News Headlines	4	1
No. Times best at Any	16	10

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 hyper-parameter models learn lower quality vector representations due to the fact that the signal within the dataset is washed out by the neural network learning noise.

3) Best Overall Model: In the case of reference models, it appears that the benefits and disadvantages of the hyper-parameters are reversed (see Table II). With a large, clean single source dataset, it seems that the ap hyper-parameters 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 the system on real-world noisy data it appears that a high thresholding hyper-parameter 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 for processing news articles, albeit with far more noise, we 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.

B. Clustering Qualitative Evaluation

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 creating 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 cluster 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.

2) Hierarchical Agglomerative Clustering: HAC acts complementary to k-NNs and allows one to select 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 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.

C. Classification Computational Performance

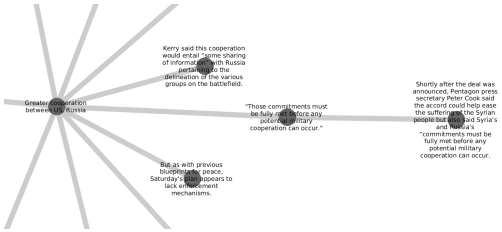

Fig. 2. An example of the content graph generated by connecting nearest neighbours from the Syrian event [22].

1) Index Generation: With any system used to address the enormous amount of news that consumers see every day, being able to process the content 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 the training system can consume, and in turn, how quickly a single computer can generate content indexes.

We find that 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 Table III). 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 processing 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) [21].

2) Clustering: We observed that in practice the computational cost of HAC overwhelms the cost of k-NNs. As such, 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 (see Table IV). 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 of articles [36]. However, due to the fact that we process events as singular news event clusters, with the largest events being capped in practice to a few thousand articles, coupled with a reasonable news article de-duplication 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 pre-processing steps such as those found in the NIFTY system to break up the vector space [6].

3) Content Graph Visualization: In this section, we visualize a select view of the content graph by connecting the nearest neighbor sentences for the Syrian event [22] with the best performing model. This effectively gives us a high-level view that demonstrates the inter-relationships we exploit within our applications to help users rapidly analyze and observe events at both a local and global level.

As we see in Figure 2, the content, which in this case are sentences represented by semantic vectors generated by the ap_dbow model, naturally fall into semantically similar clus-

TABLE III
TIME TAKEN FOR SEMANTIC CONTENT INDEX GENERATION FOR THE PROCESSING OF SENTENCE/PARAGRAPH INDEXES FOR VARIOUS EVENTS COMPARED TO THE DOCUMENT SIZE

Events	Number of Paragraphs	Number of Sentences	Paragraph Time (s)	Sentence Time (s)
Syrian event	271	380	7.37	7.58
Amazon event	2,008	3,578	87.12	95.10
Hillary event	7,544	11,812	206.14	226.71

TABLE IV
TIME TAKEN FOR HAC FOR PROCESSING OF THE SENTENCE INDEX FOR VARIOUS EVENTS

Events	Time Taken for HAC (s)
Syrian event	0.03
Amazon event	8.78
Hillary event	320.01

ters which can be explored and analyzed. This demonstrates at a local level what each of the connections and clusters represents. Each edge is a weighted similarity connection 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, or indeed by using HAC to select out particular disjoint clusters to seed an analysis, we can rapidly develop and display various views that aid the local and global analysis of a news event. To support this, we generate exploration graphs comprised 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.

V. APPLICATIONS AND CASE STUDIES

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 fractal 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 focus on the sentence index for the following case studies, with vectors generated by ap_dbow, however as our paragraph vectors are generalizable, all applications can use the paragraph index.

A. Exploratory Analysis

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

1) Related Content: Related content consumption extends related article or video paradigms used in online media systems to leverage sentence and paragraph levels. By displaying a group of nearest neighbors of a sentence dynamically we can

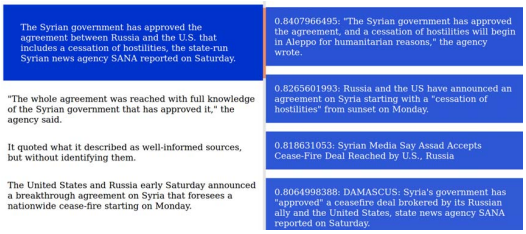

Fig. 3. An example of a related content application developed using the content graph and k-NNs allowing users to click and navigate related content and articles by sentence for rapid consumption or to see alternative viewpoints. In this example we use the Syrian event [22].

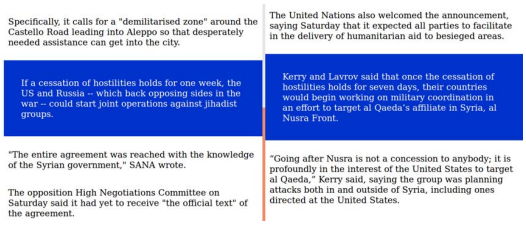

Fig. 4. An example of an in-context comparison application developed using the content graph allowing users to click and read related sections and jump to them in context for rapid consumption and comparison of alternative viewpoints. This example uses the Syrian event [22].

show different reporting and viewpoints on the same semantic content. The application seen in Figure 3 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.

2) In-Context Comparison: A logical extension to related content comparison and exploration of differential reporting is comparing and contrasting semantically similar content be-

Fig. 5. 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 using shading. This example uses the Syrian event [22].

tween articles within their respective contexts. The application in Figure 4 operates similarly to the above, but rather than navigating to the article that contains the sentence and merely highlighting it with its neighbors, 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.

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 set by how related the two pieces of content are. This allows a user to rapidly visualize how similar two pieces of reporting are and how they are structured in a rapid and visual manner. As can be seen in Figure 5, two articles can be immediately 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 be immediately isolated and compared.

B. Graph Analysis

Graph analysis is a key capability for analyzing news media. From observing how content propagates to observing how news sources inter-relate, 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 [19]. We also use NetworkX to generate graphs that are used by Cytoscape [16]. 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 inter-relationships.

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. This shows how the system can be used 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 to generate a graph that shows content propagation

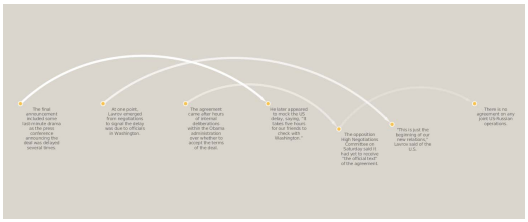

Fig. 6. An example of a propagation graph showing clusters from the content graph using HAC with k-NNs weighted connections.

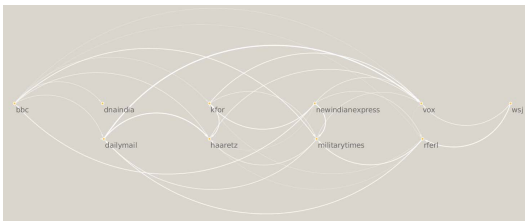

Fig. 7. Strong connections from content from the BBC with other sources indicating a top level sourcing interaction unfolding during the Syrian event [22].

through time. By utilizing the semantic vectors generated by the paragraph vector models, we are able to connect together these relations using a combination of HAC and k-NNs.

We applied this workflow to the Syrian event and selected one cluster from the content propagation graph in Figure 6 [22]. The strength and size of the inter-relations indicate the level of similarity between content, and each connection indicates a nearest neighbor relation. We can extend these connections in numerous ways, however, for this case study we simply utilized a single neighbor to illustrate strong connections and left to right time ordering to indicate the temporal 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.

2) News Source Analysis: Another core application for analyzing news media is understanding the inter-relationships 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, often refer to each other only with text with no or few direct links.

To solve this problem our system takes the idea of implicit content inter-relationships 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

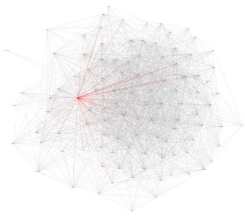

Fig. 8. Selection of the AP news source for the Hillary event zoomed out [25]. As seen, it has interrelations across numerous sources. Indeed the tightly linked center indicates the mass of near-duplicate or duplicate sources from press agencies, with the periphery in general being more analytic reporting like the Wall Street Journal (WSJ) or Wired.

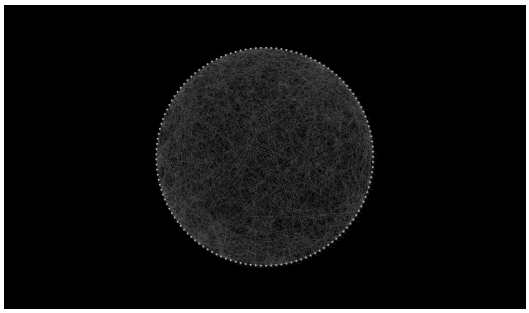

Fig. 9. By organizing the interrelationships in a circular pattern between the news sources for the Amazon event, we produce images demonstrating the complexity of online news media [24].

again use the Cytoscape graphing library and NetworkX to produce the resultant graphs [19] [16]. By leveraging software designed to solve an analogous problem, and combining it with ideas observed in the literature review, we generate a powerful visualization 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. 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, by making connections between Rodriguez et al. (2010) as well as Myers et al. (2012) [9] [10]. 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.

Indeed when we aggregate the k-NN single nearest neighbors at a source level, we approach something as shown in Figure 7 with the core-periphery information structure as found in Rodriguez et al. (2010) [9]. These are likely due to the source of external driving factors found to influence social network diffusion as identified by Myers et al. (2012) [10]. 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.

We demonstrate the scalability of the approach by extending it to the considerably larger Hillary event (see Figure 8).

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, figures such as those found in Figure 9 do give on an appreciation for the enormous complexity of the news media we consume every day.

VI. Conclusion

In this paper, we demonstrate a totally unsupervised platform with the capability to reveal the semantic content and implicit interrelated structure of news events for the purposes of differential and network analysis at a fractal level. Through an analysis of the field from a wide variety of perspectives, we focused on the analysis of semantic content propagation, that was both generalizable and useful at both paragraph and sentence levels. By combining document vectors with clustering algorithms at an index level on every aspect of content within all the articles of a news event, we generate views on the implicit content graph to support a range of end-user applications.

We address issues of scalability and generalizability through the use of high-performance document vector models and illustrate the robustness through evaluation on existing online news datasets showing the diversity and full end-to-end capability. We demonstrated this capability on the Signal 1M News Dataset, which despite having numerous quality issues endemic to online news media, with numerous duplications, mixed in code and poor formatting, with a smaller coverage set, we were able to rapidly approach state of the art for in-domain datasets. We demonstrate this with an evaluation of multiple years of SemEval STS English tasks against reference models.

We demonstrate the effectiveness of document vector models at producing vector content indexes from news event article clusters found in real-world systems. By leveraging existing news event clustering systems used in practice, we were able to develop several novel applications. We did so by generating implicit graphs on the semantic content indexes by using combinations of HAC and k-NNs. We found in practice that k-NNs operated exceptionally well in exploration applications and differential reporting, whilst HAC was good at extracting out salient groups of entities for analysis.

We demonstrated the platform through support for both exploration analysis (differential reporting) as well as higher level graph analysis (source analysis). Exploration analysis was aided by k-NNs in multiple demonstrator applications. We then demonstrated how by aggregating k-NNs and HAC for salient entity extraction, we were able to develop novel visualization into source to source interactions within a news event. In doing so, we are able to give novel insights into how online news propagates all around us.

REFERENCES

[1] "Google news," http://news.google.com, 2016, accessed: 2016-10-30.

[2] "Event registry," http://eventregistry.org, 2016, accessed: 2016-10-30.

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

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

[5] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the Dynamics of the News Cycle," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. Paris, France: ACM, 2009, pp. 497–506.

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

[7] G. Colavizza, M. Infelise, and F. Kaplan, "Mapping the Early Modern News Flow: An Enquiry by Robust Text Reuse Detection," in *Social Informatics*, L. M. Aiello and D. McFarland, Eds. Cham: Springer International Publishing, 2015, vol. 8852, pp. 244–253.

[8] S. Vakulenko, M. Göbel, A. Scharl, and L. Nixon, "Visualising the Propagation of News on the Web," in *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, pp. 60–62.

[9] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring Networks of Diffusion and Influence," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. Washington, DC, USA: ACM, 2010, pp. 1019–1028.

[10] S. A. Myers, C. Zhu, and J. Leskovec, "Information diffusion and external influence in networks," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. Beijing, China: ACM Press, 2012, p. 33.

[11] A.-L. Minard, M. Speranza, E. Agirre, I. Aldabe, M. van Erp, B. Magnini, G. Rigau, and R. Urizar, "Semeval-2015 task 4: Timeline: Cross-document event ordering," in *9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015, pp. 778–786.

[12] M. Rospocher, M. van Erp, P. Vossen, A. Fokkens, I. Aldabe, G. Rigau, A. Soroa, T. Ploeger, and T. Bogaard, "Building event-centric knowledge graphs from news," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 37-38, pp. 132–151, Mar. 2016.

[13] G. A. Miller, "WordNet: a lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.

[14] C. Analytics, "Anaconda software distribution [software]," https://continuum.io, 2016, accessed: 2016-10-30.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

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

[17] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.

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

[19] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome Res.*, vol. 13, no. 11, pp. 2498–2504, 2003.

[20] D. Corney, D. Albakour, M. Martinez-Alvarez, and S. Moussa, "What do a Million News Articles Look like?" in *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, pp. 42–47.

[21] J. H. Lau and T. Baldwin, "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation," in *Proceedings of the 1st Workshop on Representation Learning for NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 78–86.

[22] "Google news coverage of the "syrian ceasefire" story," 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, Sep. 2016, accessed: 2016-10-30.

[23] "Event registry coverage of the "tax soda to fight obesity, WHO urges nations around the globe" story," http://eventregistry.org/event/5151232?lang=&tab=articles, Oct. 2016, accessed: 2016-10-30.

[24] "Event registry coverage of the "amazon launches voice-driven on-demand music service" story," http://eventregistry.org/event/5156472?lang=&tab=articles, Oct. 2016, accessed: 2016-10-30.

[25] "Event registry coverage of the "hillary clinton, mocking and taunting, turns the tormentor" story," http://eventregistry.org/event/5178877?lang=&tab=articles, Oct. 2016, accessed: 2016-10-30.

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

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

[28] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, vol. 14, Beijing, China, 2014, pp. 1188–1196.

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

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

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

[32] E. Agirrea, C. Baneab, C. Cardiec, D. Cerd, M. Diabe, A. Gonzalez-Agirrea, W. Guof, I. Lopez-Gazpioa, M. Maritxalara, R. Mihalceab, G. Rigau, L. Uria, and J. Wiebe, "SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado, 2015, pp. 252–263.

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

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

[35] N. Bhatia, "Survey of nearest neighbor techniques," in *International Journal of Computer Science and Information Security, Vol. 8, No. 2*, 2010.

[36] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[37] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. 100, no. 1, pp. 68–86, 1971.