NSTM: Real-Time Query-Driven News Overview Composition at Bloomberg

Joshua Bambrick, Minjie Xu, Andy Almonte, Igor Malioutov, Guim Perarnau, Vittorio Selo, Iat Chong Chan

1Bloomberg, London, United Kingdom
2OakNorth, London, United Kingdom

1{jbambrick7,mxu161,aalmonte2,imalioutov,gperarnau,vselo}@bloomberg.net
2iat.chan@oaknorth.com

Abstract

Millions of news articles from hundreds of thousands of sources around the globe appear in news aggregators every day. Consuming such a volume of news presents an almost insurmountable challenge. For example, a reader searching on Bloomberg’s system for news about the U.K. would find 10,000 articles on a typical day. Apple Inc., the world’s most journalistically covered company, garners around 1,800 news articles a day.

We realized that a new kind of summarization engine was needed, one that would condense large volumes of news into short, easy to absorb points. The system would filter out noise and duplicates to identify and summarize key news about companies, countries or markets.

When given a user query, Bloomberg’s solution, Key News Themes (or NSTM), leverages state-of-the-art semantic clustering techniques and novel summarization methods to produce comprehensive, yet concise, digests to dramatically simplify the news consumption process. NSTM is available to hundreds of thousands of readers around the world and serves thousands of requests daily with sub-second latency. At ACL 2020, we will present a demo of NSTM.

1 Introduction

In many domains, finding contextually-important news as fast as possible is a key goal. With millions of articles published around the globe each day, quickly finding relevant and actionable news can mean the difference between success and failure.

When provided with a search query, a traditional system returns links to articles sorted by relevance. However, users typically encounter (near) duplicate or overlapping articles, making it hard to quickly identify key events and easy to miss less-reported stories. Moreover, news headlines are frequently sensational, opaque, or verbose, forcing readers to open and read individual articles.

For illustration, imagine an analyst sees the price of Amazon.com stock drop and wants to know why. With a traditional system, they would search for news on the company and wade through many stories (307 in this case1), often with duplicate information or unhelpful headlines, to slowly build up a full picture of what the key events were.

By contrast, using NSTM (Key News Themes), this same analyst can search for ‘Amazon.com’, over a given time horizon, and promptly receive a concise and comprehensive overview of the news, as shown in Fig. 1. We tackle the challenges involved with consuming vast quantities of news by leveraging modern techniques to semantically cluster stories, as well as innovative summarization methods to extract succinct, informational summaries for each cluster. A handful of key stories are then selected from each cluster. We define a (story cluster, summary, key stories) triple as one theme and an ordered list of themes as an overview.

NSTM works at web scale but responds to arbitrary user queries with sub-second latency. It is deployed to hundreds of thousands of users around the globe and serves thousands of requests per day.

2 Design Goals

We focus on the scenario where a news search query can render many matching news articles, from tens up to hundreds of thousands. The task is to create a succinct overview of the results to help our users to easily grasp the gist of them without combing through the individual articles.

Since the matching articles often cover various aspects and events, NSTM must first cluster related stories to form a clear separation among them.

Furthermore, the system must extract a concise

---

1 The corresponding overview can be found in Appendix C.
(up to 50 characters, or roughly 6 tokens) summary for each cluster. It needs to be short enough to be understandable to humans with a single glance, but also rich enough to retain critical details from a minimal ‘who-does-what’ stub, so the most popular noun phrase or entity alone will not suffice. Such conciseness also helps when screen space is limited (for context-driven applications or mobile devices).

From each cluster, NSTM must surface a few key stories to provide a sample of its contents. The clusters themselves should also be ranked to highlight the most important few in limited screen space.

Finally, the system must be fast. It may only take up to a few seconds for the slowest queries.

**Main technical challenges:** 1) There is no public dataset corresponding to this overview composition problem with all the requirements set above, so we were required to either define new (sub-)tasks and collect new annotations, or select techniques by intuition, implement them, and iterate on feedback; 2) Generating summaries which are simultaneously accurate, informational, fluent, and highly concise necessitates careful and innovative choices of summarization techniques; 3) Supporting arbitrary user searches in real-time places significant performance requirements on the system whilst also setting a high bar for its robustness.

### 3 Related Work

A comparable system is Google News’ ‘Full Coverage’ feature\(^2\), which groups stories from different sources, akin to our clustering approach. However, it doesn’t offer summarization and its clustered view is unavailable for arbitrary search queries.

SUMMA (Liepins et al., 2017) is another comparable system which integrates a variety of NLP components and provides support for numerous media and languages, to simultaneously monitor several media broadcasts. SUMMA applies the online clustering algorithm by Aggarwal and Yu (2006) and the extractive summarization algorithm by Almeida and Martins (2013). In contrast to NSTM, SUMMA focuses on scenarios with continuous multimedia and multilingual data streams and produces much longer summaries.

### 4 Approach

#### 4.1 Architecture

The functionality of NSTM can be formulated as: given a search query, generate a ranked list (overview) of the key themes, or (news cluster, summary, key stories) triples, that concisely represent the most important matching news events.

Fig. 2 depicts the system’s architecture. The **story ingestion service** processes millions of published news stories each day, stores them in a search index, and applies online clustering to them. When a search query is submitted via a user interface (① in the diagram), the **overview composition service** retrieves matching stories and their associated online cluster IDs from the search index (②). The system then further clusters the retrieved online clusters into the final clusters, each corresponding to one theme (③). For each such cluster, the system extracts a concise summary and a handful of key stories to reflect the cluster’s contents (④). This creates a set of themes, which NSTM ranks to create the final overview. Lastly, the system caches the overview for a limited time to support future reuse (⑤) before returning it to the UI (⑥).

#### 4.2 News Search

The first step in the NSTM pipeline is to retrieve relevant news stories (① in Fig. 2), for which we leverage a customized in-house news search engine based on Apache Solr.\(^3\) This supports searches based on keywords, metadata (such as news source

---

\(^2\)[https://www.blog.google/products/news/new-google-news-ai-meets-human-intelligence/]

\(^3\)[http://lucene.apache.org/solr/]
and time of ingestion), and tags generated during ingestion (such as topics, regions, securities, and people). For example, **TOPIC:ECOM AND NOT COMPANY:AMZN**4 will retrieve all news about ‘E-commerce’ but exclude Amazon.com.

NSTM uses Solr’s *facet* functionality to surface the largest $k$ online clusters (detailed in Sec. 4.3.2) in the search results, before returning $n$ stories from each. This tiered approach offers better coverage and scalability than direct story retrieval.

### 4.3 Clustering

#### 4.3.1 News Embedding and Similarity

At the core of any clustering system is a similarity metric. In NSTM, we define the similarity between two articles as the cosine similarity between their embeddings as computed by NVDM (Miao et al., 2016), i.e., $\tau(d_1, d_2) = 0.5(\cos(z_1, z_2) + 1)$, where $z \in \mathbb{R}^n$ denotes the NVDM embedding.

Our choice is motivated by two observations: 1) The generative model of NVDM is based on bag-of-words (BoW) and $P(w|z) = \sigma(W^T z)$ where $\sigma$ is the softmax function, $W \in \mathbb{R}^{n \times V}$ is the word embedding matrix in the decoder and $V$ is the size of the vocabulary. This resembles the latent topic structure popularized by LDA (Blei et al., 2003) which has proven effective in capturing textual semantics. Additionally, the use of cosine similarities is naturally motivated by the fact that the generative model is directly defined by the dot-product between the story embedding ($z$) and a shared vocabulary embedding ($W$). 2) NVDM’s Variational Autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) framework makes the inference procedure much simpler than LDA and it also supports decoder customizations. For example, it allows us to easily integrate the idea of introducing a learnable common background word distribution into the generative model (Arora et al., 2017).

We trained the model on an internal corpus of 1.85M news articles, using a vocabulary of size about 200k and a latent dimension $n$ of 128.

#### 4.3.2 Clustering Stages

We divide clustering into two stages in the pipeline, 1) online incremental clustering at story ingestion time, and 2) hierarchical agglomerative clustering (HAC) at query time (3 in Fig. 2). The former is used to produce query-agnostic online clusters at a relatively low cost to handle the daily influx of millions of news stories. These clusters reduce the computational cost at query time. However, due to its online nature, over-fragmentation, among other quality issues, occurs in the resulting clusters. This necessitates further refinement at query time when an offline HAC step is performed on top of the retrieved online clusters. A similar, but more complicated, design was adopted in Vadrevu et al. (2011) for clustering real-time news search results.

At both stages, we compute the cluster embedding $z_c \in \mathbb{R}^n$ as the mean of all the story embeddings therein, and evaluate similarities between clusters (individual stories are taken as singleton clusters) using the metric $\tau$ defined in Sec. 4.3.1.

For online clustering, we apply an in-house implementation which uses a distributed pool of workers to reduce latency and increase throughput. It merges each incoming story with the closest cluster if the similarity is within a parameterized threshold and otherwise creates a new singleton cluster.

For HAC, we apply fastcluster\textsuperscript{5} (Müller, 2013) to construct the dendrogram. We use complete linkage to encourage more congruent clusters and then form flat clusters by cutting the dendrogram at the same (height) threshold. To further

---

\textsuperscript{4}This is Bloomberg’s internal news search query syntax, which maps closely to the final query submitted to Solr.

\textsuperscript{5}https://www.jstatsoft.org/article/view/v053i09
reduce fragmentation where similar clusters are left un-clustered, we apply HAC twice recursively.

To find a reasonable similarity threshold, we manually annotated just over 1k pairs of news articles. Each annotator indicated whether they would expect to see the articles grouped together or not in an overview. We then selected the threshold which achieved the highest F1 score on this binary classification task, which was 0.86.

4.4 Summary Extraction

Clustering search results (Vadrevu et al., 2011) is a meaningful step towards creating a useful overview. With NSTM, we push this one step further by additionally generating a concise, yet still human-readable, summary for each cluster (4 in Fig. 2).

Due to the unique style of the summary explained in Sec. 2, the scarcity of training data makes it hard to train an end-to-end seq2seq (Sutskever et al., 2014) model, as is typical for abstractive summarization. Also, this technique would only offer limited control over the output. Hence, we opt for an extractive method, leveraging OpenIE (Banko et al., 2007) and a BERT-based sentence compressor (Devlin et al., 2019) to surface a pool of sub-sentence-level candidate summaries from the headline and the body, which are then scored by a ranker.

4.4.1 OpenIE-based Tuple Extraction

Open Domain Information Extraction (OpenIE) presents an unsupervised approach to extract summary candidates from an input sentence.

First, we construct a dependency parse tree of the sentence, using a model based on Kiperwasser and Goldberg (2016) (1 in Fig. 3).

From this tree, we extract predicate-argument n-tuples using an adapted reimplementation of PredPatt (White et al., 2016) (2). The tuples represent nested proto-semantic parses of the sentence, and typically correspond to well-formed phrases. This method applies rules cast over Universal Dependencies (Nivre et al., 2016) so syntactic patterns are unlexicalized and language-neutral.

We then prune these tuples (3), applying rules which reduce the arguments to their syntactic heads, while heuristics keep named entities and multi-word expressions intact. We recursively intersect the resulting tuples to create more tuples.

Finally, to render summary candidates, we create a titlecased surface form of each tuple (4).

4.4.2 BERT-based Sentence Compression

In addition to the rule-based OpenIE system, we apply a Transfer Learning-based solution, using a novel in-house dataset specific to our sub-task. In particular, we model candidate summary extraction as a ‘sentence compression’ task (Filippova et al., 2015), where each story is split into sentences and tokens are classified as keep or delete to make each sentence shorter, while retaining the key message.

We oversaw the manual annotation of a dataset which maps sentences to compressed equivalents that correspond to summaries. When presented with a news story, annotators selected one sentence and deleted words to create a high quality summary. This rendered 10k annotations which we randomly partitioned into train (80%) and test (20%) sets.

The task is formulated as sequence tagging, whereby each sub-token (1 in Fig. 3), defined using the BERT vocabulary, is classified as keep or delete (2). We implement this using a feedforward layer on top of a Bloomberg-internal pre-trained neural network, akin to the uncased English BERT-Base model, applying an adapted implementation.

To create a compression, we stitch sub-tokens labelled keep together (3). Lastly, we use post-processing rules to improve formatting (4), such as titlecasing and fixing partial-entity deletion (where only some sub-tokens of a token/entity are deleted).

4.4.3 Summary Candidate Ranking

Tuple generation and sentence compression provide a pool of summary candidates for individual news stories. These are further aggregated across stories within a cluster to form the final pool. To identify the best summary for the cluster, we trained a sequence-pair model \( s_\theta(a, c) \) to score each candidate \( c \) given an article \( a \). Such article-level scores for a candidate are computed against all the stories in a cluster and then aggregated (e.g., averaged) to produce the final cluster-level scores, which we use for ranking.

For this purpose, we collected an in-house annotated dataset. We sampled a few thousand news articles and generated 33k summary candidates from them using OpenIE.\(^6\) Then we asked internal annotators to label each as Great, Acceptable or Terrible were it to be used as a summary for the article, considering both readability and informativeness.

From this dataset, we constructed about 48k pairwise samples \((c, c')|a\) where \(c\) is labelled more

\(^6\) At this time, we hadn’t considered sentence compression.
Automaker ST is investing $2B in electric vehicles (EVs), atoning for the 2018 scandal

Figure 3: Illustrations of the symbolic OpenIE (left) and neural sentence compression (right) candidate extraction pipelines. We apply both, to render a diverse pool of candidate summaries, and use a ranker to select the best.

favorably than $c'$ for a given common article $a$, and the model $s_\theta(a, c)$ was then trained to match such preferences using pairwise margin loss, i.e., $\max(0, 1 - s_\theta(a, c) + s_\theta(a, c'))$.

We considered a few models, including a parameter-free baseline which scores candidate-article pairs as the dot-product of their NVDM (Sec. 4.3.1) embeddings, i.e., $s = z^\top_a z_c$. We also considered this model’s bilinear extension $s = z^\top_a W z_c$ where $W$ is the learnable weight matrix. Lastly, we tried neural network models, such as DecAtt (Parikh et al., 2016). We evaluated these models on a held-out test set with metrics such as pairwise ranking accuracy and NDCG. We opted to productionize the baseline model, since it was the simplest and performed on par with the others. We considered a few models, including a parameter-free baseline which scores candidate-article pairs as the dot-product of their NVDM (Sec. 4.3.1) embeddings, i.e., $s = z^\top_a z_c$. We also considered this model’s bilinear extension $s = z^\top_a W z_c$ where $W$ is the learnable weight matrix. Lastly, we tried neural network models, such as DecAtt (Parikh et al., 2016). We evaluated these models on a held-out test set with metrics such as pairwise ranking accuracy and NDCG. We opted to productionize the baseline model, since it was the simplest and performed on par with the others.

Because NVDM uses a bag-of-words model, this ranker ignores syntax entirely. We believe that its empirical success owes to both the well-formedness of the majority of the candidates and the averaging effect that amplifies the ‘signal-noise ratio’ when the scores are averaged over the cluster.

Empirically, this approach tends to surface ‘informational’ summaries, in contrast to headlines which are often ‘sensational’. We posit that this is because high-ranked summaries must also be representative of story bodies, not just headlines.

4.4.4 Combining Summary Candidates

OpenIE and sentence compression offer distinct ways to extract candidates, and we experimented with each as the sole source of summary candidates in our pipeline. On the basis of ROUGE scores (Lin and Hovy, 2003; Lin, 2004) (details in Appendix B), the latter provides superior results.

However, in a production system which informs business decisions, we must consider factors which aren’t readily captured by metrics which compare generated and ‘gold’ outputs. For example, changing a single word can reverse the meaning of a summary, with only a small change in such scores. Hence, we consider a range of pros and cons.

The sentence compression method is supervised and is trained to produce summaries which can take advantage of news-specific grammatical styles. However, the OpenIE system is much faster and offers greater interpretability and controllability.

Since the neural and symbolic systems provide different advantages, we apply both. This renders a diverse pool of candidate summaries from which the ranker’s task is to select the best. At the pooling stage we also impose a length constraint of 50 characters and exclude any longer candidates.

4.5 Key Story Selection

As a sample from the full story cluster, NSTM selects an ordered list of key stories which are deemed to be representative. We select these using a heuristic based on intuition and client feedback.

Our approach is to re-cluster all stories in the cluster using HAC (see Sec. 4.3.2), to create a parameterized number of sub-clusters. For each sub-cluster, we select the story that has maximum average similarity $\tau$ (as per Sec. 4.3.1) to the other sub-cluster stories. This strategy is intended to select stories which represent each cluster’s diversity.

We sort the key stories by sub-cluster size and time of ingestion, in that order of precedence.
4.6 Theme Ranking

We have described how (story cluster, summary, key stories) triples, or themes, are created. However, some themes are considered to be more important than others since they are more useful to readers. It is tricky to define this concept concretely but we apply proxy metrics in order to estimate an importance score for each theme. We rank themes by this score and, in order to save screen space, return only the top few (‘key’) themes as an overview.

The main factor considered in the importance score is the size of the story cluster – the larger the cluster, the larger the score. This heuristic corresponds to the observation that more important themes tend to be reported on more frequently. Additionally, we consider the entropy of the news sources in the cluster, which corresponds to the observation that more important themes are reported on by a larger number of publishers and reduces the impact of a source publishing duplicate stories.

4.7 Caching

Since many user requests are the same or use similar data, caching is useful to minimize response times. When NSTM receives a request, it checks whether there is a corresponding overview in the cache, and immediately returns it if so. 99.6% of requests hit the cache and 99% of requests are handled within 215ms. In the event of a cache miss, NSTM responds in a median time of 723ms.

We apply two mechanisms to ensure cache freshness. Firstly, we preemptively invoke NSTM using requests that are likely to be queried by users (e.g., most read topics) and re-compose them from scratch at fixed intervals (e.g., every 30 min). Once computed, they are cached. The second mechanism is user-driven: every time a user requests an overview which is not cached, it will be created and added to the cache. The system will subsequently preemptively invoke NSTM using this request for a fixed period of time (e.g., 24 hours).

5 Demonstration

NSTM was deployed to our clients in 2019. Using the UI depicted in Fig. 1, users can find overviews for customized queries to help support their work. From this screen, the user can enter a search query using any combination of Boolean logic with tag- or keyword-based terms. They may also alter the period that the overview is calculated over (this UI offers 1 hour, 8 hour, 1 day, and 2 day options).

This interface also allows users to provide feedback via the ‘thumb’ icons or plain-text comments. Of several hundred per-overview feedback submissions, over three quarters have been positive.

Tables 1 and 2 show example theme summaries generated for the queries ‘Facebook’ and ‘U.K.’. Note that the summaries are quite different from what has previously been studied by the NLP community (in terms of brevity and grammatical style) and that they accurately represent distinct events.

In addition to user-driven settings, NSTM can be used to supplement context-driven applications. One example, demonstrated in Appendix D, uses themes provided by NSTM to help explain why companies or topics are ‘trending’.

6 Conclusion

We presented NSTM, a novel and production-ready system that composes concise and human-readable news overviews given arbitrary user search queries. NSTM is the first of its kind; it is query-driven, it offers unique news overviews which leverage clustering and succinct summarization, and it has been released to hundreds of thousands of users.

We also demonstrated effective adoption of modern NLP techniques and advances in the design and implementation of the system, which we believe will be of interest to the community.

There are many open questions which we intend to research, such as whether autoregressivity in neural sentence compression can be exploited and how to compose themes over longer time periods.

Table 1: Ranked theme summaries and cluster sizes for ‘Facebook’ (1,176 matching stories) from 31 Jan. 2020.

| Summary                                      | Size |
|----------------------------------------------|------|
| 1 Facebook to Settle Recognition Privacy Lawsuit | 90   |
| 2 Facebook Warns Revenue Growth Slowing     | 79   |
| 3 Facebook Stock Drops 7% Despite Earnings Beat | 70   |
| 4 Facebook to Remove Coronavirus Misinformation | 49   |
| 5 Mark Zuckerberg to Launch WhatsApp Payments | 19   |

Table 2: Ranked theme summaries and cluster sizes for ‘U.K.’ (13,858 matching stories) from 31 Jan. 2020.

| Summary                                      | Size  |
|----------------------------------------------|-------|
| 1 Britain to Leave the EU                   | 459   |
| 2 Bank of England Would Keep Interest Rate Unchanged | 141  |
| 3 Sturgeon Demands Scottish Independence Vote | 71    |
| 4 Pompeo in UK for Trade Talks              | 45    |
| 5 Boris Johnson Hails ‘Beginning’ on Brexit Day   | 63    |

8 Computed for all requests over a 90-day period.
9 Computed for the top 50 searches over a 7-day period.
References

Charu C Aggarwal and Philip S Yu. 2006. A framework for clustering massive text and categorical data streams. In Proceedings of the 2006 SIAM International Conference on Data Mining, pages 479–483. SIAM.

Miguel Almeida and André Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 196–206, Sofia, Bulgaria. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In Proceedings of the 5th International Conference on Learning Representations, ICLR’17. OpenReview.net.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI’07, page 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 360–368, Lisbon, Portugal. Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. Transactions of the Association for Computational Linguistics, 4:313–327.

Renars Liepins, Ulrich Germann, Guntis Barzdins, Alexandra Birch, Steve Renals, Susanne Weber, Peggy van der Kreeft, Hervé Bourlard, João Prieto, Ondřej Klejch, Peter Bell, Alexandros Lazaridis, Alfonso Mendes, Sebastian Riedel, Mariana S. C. Almeida, Pedro Balage, Shay B. Cohen, Tomasz Dwojak, Philip N. Garner, Andreas Gießer, Marcin Junczys-Dowmunt, Hina Imran, David Nogueira, Ahmed Ali, Sebastián Miranda, Andrei Popescu-Belis, Lesly Miculichic Werlen, Nikos Papasarantopoulos, Abiola Obamuyide, Clive Jones, Fahim Dalvi, Andreas Vlachos, Yang Wang, Sibo Tong, Rico Sennrich, Nikolaos Pappas, Shashi Narayanan, Marco Damonte, Nadir Durraní, Sameer Khurana, Ahmed Abdelali, Hassan Saïjad, Stephan Vogel, David Shepperd, Chris Hermon, and Jeff Mitchell. 2017. The SUMMA platform prototype. In Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pages 116–119, Valencia, Spain. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pages 150–157.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pages 1727–1736. JMLR.org.

Daniel Müllner. 2013. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. Journal of Statistical Software, Articles, 53(9):1–18.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16), pages 1659–1666.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, pages 1278–1286.
Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pages 3104–3112.

Srinivas Vadrevu, Choon Hui Teo, Suju Rajan, Kunal Punera, Byron Dom, Alexander J. Smola, Yi Chang, and Zhaohui Zheng. 2011. Scalable clustering of news search results. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM’11, pages 675–684, New York, NY, USA. ACM.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.
A  Acknowledgements

This has been a multi-year project, involving contributions from many people at different stages.

In particular, we thank Miles Osborne, Marco Ponza, Amanda Stent, Mohamed Yahya, Christoph Teichmann, Prabhanjan Kambadur, Umut Topkara, Ted Merz, Sam Brody, and Adrian Benton for reviewing and commenting on the manuscript; We further thank Adela Quinones, Shaun Waters, Mark Dimont, Ted Merz and other colleagues from the News Product group for helping to shape the vision of the system; We also thank José Abarca and his team for developing the user interface; We thank Hady Elsahar for helping to improve summary ranking during his internship; Finally, we thank all colleagues (especially those in the Global Data department) who helped to produce high quality in-house annotations and all others who contributed valuable thoughts and time into this work.

B  End-To-End Evaluation

We evaluate the end-to-end NSTM system when using the OpenIE (Sec. 4.4.1) and the BERT-based sentence compression (Sec. 4.4.2) algorithms as the sole source of candidate summaries. We also conducted one experiment where both were used to create a shared pool of candidates (as per Sec. 4.4.4).

We test the system end-to-end using the manually-annotated Single Document Summarization (SDS) test set described in Sec. 4.4.2. To implement SDS, our experimental setup assumes that only one story was returned by a search request (as per Sec. 4.2). We evaluate the output from each system with ROUGE (Lin and Hovy, 2003; Lin, 2004)

| Metric       | OpenIE | BSC  | Both  |
|--------------|--------|------|-------|
| ROUGE-1 F1   | 0.831  | 0.863| 0.851 |
| ROUGE-2 F1   | 0.609  | 0.701| 0.667 |
| ROUGE-3 F1   | 0.530  | 0.640| 0.599 |
| ROUGE-4 F1   | 0.492  | 0.603| 0.562 |
| ROUGE-L F1   | 0.621  | 0.706| 0.670 |

Table 3: ROUGE scores for the Single-Document Summarization task in the end-to-end system, when using OpenIE, BERT-based sentence compression (BSC) and both to construct the pool of candidate summaries.

https://github.com/google/seq2seq/blob/master/seq2seq/metrics/rouge.py
C Screenshots of A Query-Driven User Interface

Figure 4: Screenshot (taken on 29 January 2020) of a query-driven interface for NSTM showing the overview for the company 'Amazon.com'.

Figure 5: Screenshot (taken on 29 January 2020) of a query-driven interface for NSTM showing the overview for the topic ‘Electric Vehicles’.
Figure 6: Screenshot (taken on 29 January 2020) of a query-driven interface for NSTM showing the overview for the region ‘Canada’.

Figure 7: Screenshot (taken on 29 January 2020) of a query-driven interface for NSTM showing the overview for a complex query, including a keyword.
Figure 8: Screenshot (taken on 29 January 2020) of a context-driven application of NSTM. In the ‘Security’ column are the companies that have seen the largest increase in news readership over the last day. Each entry in the ‘News Summary’ column is the summary of the top theme provided by NSTM for the adjacent company.

Figure 9: Screenshot (taken on 29 January 2020) of a context-driven application of NSTM. In the ‘News Topic’ column are the topics that have seen the largest volume of news readership over the past 8 hours. Each entry in the ‘News Summary’ column is the summary of the top theme provided by NSTM for the adjacent topic.