Automated Financial Commentary Generation

Utkarsh Chaturvedi
National University of Singapore

Abstract
The past decade has seen an upsurge in research of unstructured data. At the forefront of this wave has been advancements made in processing, understanding and finally recreating and/or generating natural language. Commercial applications of this research have reached the market as organisations look to automate the mundane aspects of their repetitive tasks and allow employees to invest their effort on non-trivial problems. This project aims to automate the trivial aspects of writing financial commentaries, with the commentaries written in a language and syntax similar to that of organisation analysts. NLP techniques such as phrase modelling, NLTK similarity analysis and POS tagging, topic modelling along with word2vec models are used to explore the current language. A feasibility study is performed to understand whether neural network-based models can be used to perform the 'generation' task and finally, a natural language 'realiser' is used to generate commentaries.

1. Introduction

To briefly introduce the organisation, it is a healthcare giant operating across multiple geographies. Each region has a dedicated business-unit (henceforth referred to as 'BU') that performs the day-to-day operations for each type of product sold. With such a sprawling organisational structure, it becomes critical to keep a sharp eye on the financial performance across the board and highlight deviations or interesting phenomena as it occurs. To accomplish this task, a central team of financial analysts combs through the numbers and various metrics each month and generates commentaries that are then sent off to relevant stakeholders.

To allow the analyst to devote more time in performing value-added activities and increase the efficiency of the process of writing commentaries, the business would like this process to be automated to a certain extent. This automation framework should also take into consideration the following conditions:

- Identify the product-groups where commentary is required or allow the user to give an identification rule for this
- Generate more insightful commentaries with each iteration of the framework incorporating more data sources and advanced techniques
- Try to replicate the businesses 'style' of writing commentaries
- Ensure robustness to below factors:
    - Addition of more geographical regions
    - Levels of hierarchy in data
    - Scalability of solution across different business units

A long-term solution would incorporate a feedback loop as well, where corrections made by the analyst on generated commentaries would consider those correction in the next run.

2. Literature Review

The simplest example of an NLG implementation is that from a Markov Chain. Based on the ground-breaking work of mathematician A.A. Markov in 1906, it laid one of the foundations of modern probability theory by studying the influence of a past experiment to predict the outcome of future experiment. Its application to natural language was brought into prominence in the landmark Information Theory paper, A Mathematical Theory of Communication [Shannon, 1948], that proposed using a Markov Chain to create a statistical model of sequences of letters in a piece of English text. Variants of Markov chain models have since seen many real-world applications including speech-recognition, information retrieval, spam filtering and Google's Page Rank algorithm.

It is important to demarcate different types of NLG systems based on their purpose: the text-to-text systems and data-to-text systems. The former uses language already present or prompted to continue language generation based on the algorithm it is working on and has seen ground-breaking progress in the last 3 years with very rapid advancements from corporations and dedicated researchers. Chatbots, Q&A systems, information retrieval, classification problems, etc. all rely on algorithms developed for text-to-text systems. An end-to-end automation of the latter system though, continues to be an area of active research, primarily due to 2 fundamental problems that need to be addressed beyond just being able to generate language comprehensible to humans, which are:

- **Context Comprehension:** Given a table with a bunch of financial numbers, can the system switch its language when writing about profit vs. revenue? Is there a mechanism where it can learn to differentiate between profit and revenue and thus, switch between the lexicons used to describe either?
- **Accuracy:** Since neural networks do not have an optimal convergence point, it can cause data hallucinations where there are factual mistakes or sometimes incoherent rambling that does not tie back to any data. This also leads to a much more fundamental problem. How should we measure the accuracy of the generated commentaries with facts in data?

Measurement of data-to-text accuracy:

NLP researchers tend to use BLEU (Bilingual Evaluation Understudy) [Papineni et al., 2002] and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [Lin & Hovy, 2003] to benchmark their NLP algorithms. These metrics can be considered equivalent to precision and recall i.e. the former measures 'how many machine-generated n-grams appear in human-referenced summaries' and 'how many n-grams of human reference n-grams appear in machine-generated summaries' respectively. There are variants of these metrics created to penalise certain characteristics or specific to use-cases. The problem with using these evaluation metrics should be quite evident; they were developed for measuring accuracy of text-to-text systems. These metrics give a fair idea of performance in cases of language translation or text generation but fall short of measuring the accuracy of data to that of generated text. In a comprehensive study done by Ehud Reiter [Reiter et al., 2008], the authors determine that while BLEU-like metrics can be a fair judgement of linguistic 'quality', given two different generated texts with one being easier to read and the other being 'accurate', users preferred

the accurate version. It should also be noticed that the 'user preference' was requested on weather data in the previous study. When moving to domains such as healthcare or finance, due to exceptionally high cost of failure, the margin of error in data to text translation would be zero for any user. This study also explores the question of does correlation of generated text with human judgement imply correlation with task-effectiveness? Though the answer remains inconclusive due to lack of such studies, it is important to question that while measurement metrics are used to ensure alignment between human-generated and machine-generated text, was the human-generated text ever clear in its communicative goal?

This discussion about measurement of accuracy allows the topic of different paradigms of data-to-text systems to be broached. As of today, Data-to-text NLG systems are split into three different paradigms. They are:

- Templatization
- Pipeline Architecture
- End-to-end (E2E) neural network-based solution

Templatization

As evident from the name, this system involves building up templates where the data is input as a fill-in-the-blanks format. While this could be useful if there are very few scenarios to fill in data for, it becomes cumbersome and difficult to maintain as the complexity of task increases. One of the biggest advantages of this system, though, is ease in building it.

Pipeline Architecture

An older, but still very relevant paradigm is that of using a pipeline architecture to generate commentaries, proposed in 1997 [Reiter et al., 1997]. The pipeline consists of 3 components: Document Planning, Microplanning & Realisation (Also commonly known as Content Determination, Sentence Planning & Surface Realisation respectively), as shown in Fig.1 below.

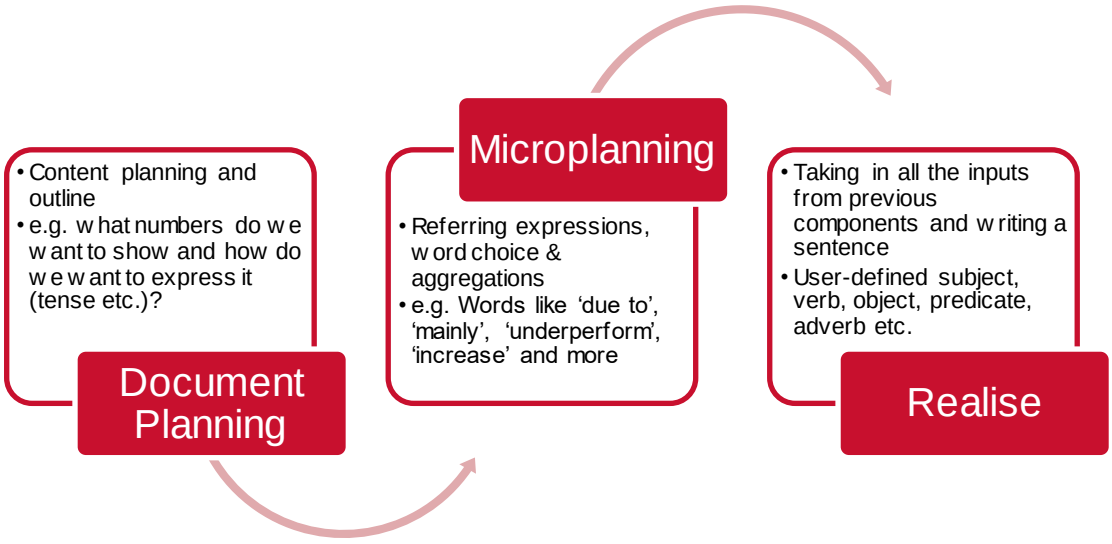

Figure 1: Process Flow of Pipeline Architecture of NLG systems

In the paper, NLG vs. Templates [Reiter, 1995], Reiter explores the 'value-add' of NLG over templates. The paper points to 3 key areas where NLG is superior to templates; **Maintainability** which discusses how template-based generators may be difficult to modify according to changing user-needs, **text-quality** particularly sentence planning and syntactic realisation (although it is to be noted that just good syntactic processing may not give a 'competitive advantage') over templates and lastly **a guaranteed conformance to standards.** Another paper published in 2003 by students and researchers across multiple universities [Deemter et al., 2003], arrives at a similar conclusion. This paper explores each aspect of language building, namely, Content Determination, Referring Expressions, Aggregation, Lexicalisation & Linguistic Realisation, and compares how NLG systems are superior to templates. It also acknowledges certain cases where template-based systems can be superior such as in cases where good linguistic rules are not (yet) available or for constructions which have unpredictable meanings or highly specific conditions of use.

These advantages make this type of system still relevant even after being in use for about two decades. There are some popular commercial applications that exist and build upon this system. ArriaNLG and Wordsmith are two popular NLG engines in the market today.

An ideal use-case of this system is shown in yet another paper by Reiter [Portet et al., 2007]. In this project, the team developed 4 different systems (named BT-45, BT-Burse, BT-Doc & BT-Family), targeted at each of the stakeholders in an ICU unit. A summary of the systems (directly referenced from the paper):

- BT-45: descriptive summary of 45 minutes data
- BT-Nurse: summary of 12 hours of data to serve as shift summary
- BT-Doc: similar to BT-Nurse but with the intention of supporting decision making by junior doctors
- BT-Family: a daily summary for the baby's family, adopted to emotional state of the recipient

The team had access to physiological time series data, along with some discrete data from different machines and actions of the medical staff. The architecture of BT-45 is as shown in Fig. 2 below:

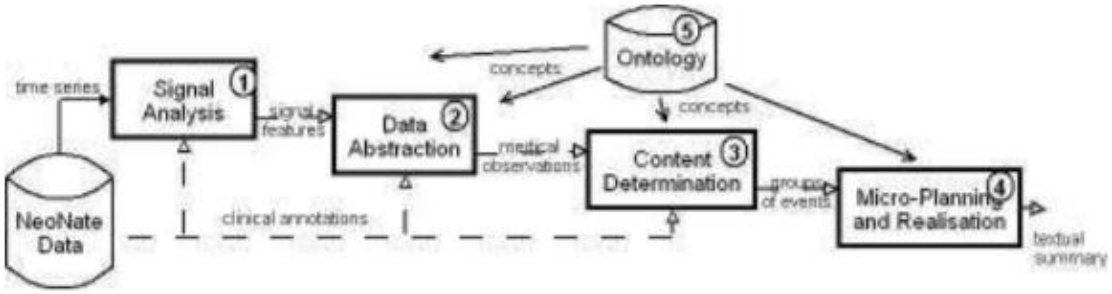

Figure 2: Architecture of BT-45 from Portet et al.

The content determination is the key here, and the paper explains it best: "Certain events need to be grouped and then the event groups are used to compose the tree for the document. The decision as to which groups to mention in the text is based on heuristics which try to produce a document of certain length. A pruning of events is also based on their importance and a notion of not mentioning events that the reader will infer by themselves".

Using this, the team was able generate realistic looking summaries, which are seen in Fig. 3:

BT-45	EXPERT
*** introduction ***	*** introduction ***
At 10:30 you see the baby.	You see the infant first at 1030.
HR = 148, mean BP = 28, central temperature = 37.5, peripheral temperature = 36.3 and sats = 96.	The transcutaneous OX/CO electrode is being re-calibrated.
*** 1 ***	*** 1 ***
Saturations fall for 10 minutes to 77 so FiO2 is decreased to 36.	In preparation for re-intubation, a bolus of 50ug of morphine is given at 1039 when the FiO2 = 35%. There is a momentary bradycardia and then the mean BP increases to 40. The sats go down to 79 and take 2 mins to come back up. The toe/core temperature gap increases to 1.6 degrees.
At 10:38 FiO2 is increased to 35 so saturations rise for 8 minutes to 96 and saturations drop to 78 and there is a bradycardia to 90.	
*** 2 ***	*** 2 ***
At 10:46 peripheral temperature falls for 13 minutes to 33.5 and then central temperature falls for 5 minutes to 36.5.	At 1046 the baby is turned for re-intubation and re-intubation is complete by 1100 the baby being bagged with 60% oxygen between tubes. During the re-intubation there have been some significant bradycardias down to 60/min, but the sats have remained OK. The mean BP has varied between 23 and 56, but has now settled at 30. The central temperature has fallen to 36.1°C and the peripheral temperature to 33.7°C. The baby has needed up to 80% oxygen to keep the sats up.
At 10:51 the baby is intubated and so there is a significant bradycardia to 61, there is a desaturation to 77 and mean BP jumps to 47. As part of this procedure, at 10:47 the baby is hand-bagged so there is a bradycardia to 125.	
At 10:52 toe/core temperature gap rises for 7 minutes to 2.4, at 11:02 FiO2 is decreased to 67 and FiO2 is increased to 79.	
*** 3 ***	*** 3 ***
Saturations drop to 79 and then at 11:05 saturations rise for 6 minutes to 99 so FiO2 is decreased to 80.	Over the next 10 mins the HR decreases to 140 and the mean BP = 30-40. The sats fall with ETT suction so the FiO2 is increased to 80% but by 1112 the FiO2 is down to 49%.
The baby is sucked out and at 11:09 FiO2 is increased to 61.	

Figure 3: Comparison of generated text vs actual of BT-45 from Portet et al.

Another challenge of language generation is choosing appropriate words to represent data. While developing 'SUMTIME-MOUSAM weather-forecast generator', Ehud Reiter's [Reiter et al., 2005] team analysed the corpora of words being used to forecast weather and identified major differences in how individuals performed this task. The paper also keenly analyses how something as simple as time phrases (by evening, by late evening, by morning etc.) can be written and interpreted by different people to mean different times. It looks at verbs (decreasing winds vs. easing winds) and shows how different experts use them in sometimes contradictory manner. Lastly, they also test two hypotheses regarding whether generated text is more 'appropriate' and whether readers 'comprehend' it better. Both these hypotheses turn out to be true and readers do in-fact prefer the SUNTIME-MOUSAM system more than humans. This shows how standardising the language can be advantageous to the reader and how word-usage which might seem a trivial matter is anything but that.

End-to-end (E2E) NLG

The last paradigm is the E2E NLG systems. With the advent of neural-network based algorithms, especially RNN architectures, NLP/NLG parsing algorithms breathed a new lease of life in this field. With dedicated research and lower hardware costs in the last decade, neural networks have just gotten bigger

with more complex architectures. Since 2017, when transformers (encoder-decoder) were introduced as attention mechanisms [Vaswani et al., 2017], almost all NLG algorithms have used this architecture in some form or the other.

Apart from the architectural advancements, progress has been made in application of these huge models as well. The technique of starting from a pre-trained model (on a large corpora of text) and then fine-tuning it to the required task has led to substantial gains in NLP/NLG benchmarks. This technique is known as 'Few-shot learning' and has seen extensive research and application in the domain of natural language in the last 2 or 3 years. The goal is to achieve one-shot and zero-shot learning where the model needs one or zero training data for fine-tuning. Given that this project does not have a lot of training data, few-shot NLG applications are of interest in this project.

With corporates getting into fray, there is unprecedented advancement happening in the field. Earlier in the year, Microsoft released Turing-NLG [Microsoft, 2020] for downstream NLP tasks having 17 billion parameters. This model beat the state-of-the art at the time on various tasks by substantial margins. Last month, OpenAI released their much-anticipated GPT-3 model [Brown et al., 2020] with a staggering 175 billion parameters. It has taken the natural language domain by storm with almost unprecedented number of applications and an uncanny ability to understand the user's prompts. The applications of GPT-3 were again tested using few-shot learning. A link to some of the exploration forays into GPT-3 is here (19th July, 2020). This paper also ponders upon the societal impacts of being able to generate language, which is difficult to distinguish from humans and how it can be misused. It also explores biases present in the model based on gender, race and religion.

While all the above advancements have substantially improved language quality & comprehension, accuracy of information in data-to-text systems remains a concern. A study by Intel AI and University of California, Santa Barbara [Chen et al., 2020] showcases how few-shot learning on pre-trained models can help achieve strong performance across various domains. When asked by humans to evaluate the 'Factual Correctness', it performs well compared to other studies but is still quite a distance away from the gold standard which is required when working on financial or healthcare data. In-fact, Wiseman et al. [Wiseman et al., 2017] performed a study exclusively on challenges faced in data-to-document generation, proposed some new techniques but still concluded that there is a significant gap between neural network generated test and templated systems. In 2019, researchers from the University of Edinburgh [Puduppully et al., 2019] tried to improve upon Wiseman's results and were successful in some regards but still unable to reach gold standards. An example of the neural network hallucinating facts from the paper is provided below (blue text is correct facts, erroneous texts in *red*):

TEMPL System (State-of-the-art Template Hybrid System)

The Golden State Warriors (10–2) defeated the Boston Celtics (6–6) 104–88. Klay Thompson scored 28 points (12–21 FG, 3–6 3PT, 1–1 FT) to go with 4 rebounds. Kevin Durant scored 23 points (10–13 FG, 1–2 3PT, 2–4 FT) to go with 10 rebounds.

Isaiah Thomas scored 18 points (4–12 FG, 1–6 3PT, 9–9 FT) to go with 2 rebounds. Avery Bradley scored 17 points (7–15 FG, 2–4 3PT, 1–2 FT) to go with 10 rebounds. Stephen Curry scored 16 points (7–20 FG, 2–10 3PT, 0–0 FT) to go with 3 rebounds. Terry Rozier scored 11 points (3–5 FG, 2–3 3PT, 3–4 FT) to go with 7 rebounds. The Golden State Warriors' next game will be at home against the Dallas Mavericks, while the Boston Celtics will travel to play the Bulls.

Neural Network System used in Puduppully et al.

The Golden State Warriors defeated the Boston Celtics 104–88 at TD Garden on Friday. The Warriors (10–2) came into this game winners of five of their last six games, but *the Warriors (6–6)* were able to pull away in the second half. Klay Thompson led the way for the Warriors with 28 points on 12–of–21 shooting, while Kevin Durant added 23 points, 10 rebounds, seven assists and two steals. Stephen Curry added 16 points and eight assists, while Draymond Green rounded out the box score with 11 points, eight rebounds and eight assists. For the Celtics, it was Isaiah Thomas who shot just 4–of–12 from the field and finished with 18 points. Avery Bradley added 17 points and 10 rebounds, while the rest of the Celtics *combined to score just seven points*. Boston will look to get back on track as they play host to the 76ers on Friday.

A simpler implementation of NLG for financial commentaries can also be found in a study by Mokhtari et al. [Mokhtari et al., 2019]. In this study, the team uses an RNN based encoder-decoder architecture to make decisions on whether a commentary is necessary and if so, how should it be written. A cause of concern here is the usage of evaluation metrics such as BLEU, ROUGE, and METEOR. As established before, while these metrics give a fair idea of the quality of language, the authors have not performed any study on the accuracy of information generated, which, in a business use-case outside of academia is important. It is also perplexing to the author of this project on why an RNN architecture is used for encoder which is fed a vector of financial numbers. Using an RNN style architecture for decoder is not disputed as the output is commentary and input is commentary embedded as a vector.

3. Solution: Description of Approach

3.1 Problem Breakdown

To solve a business problem, it is vital to understand the pain point. So, the aspect of improving efficiency in the commentary generation process is broken down as shown in Fig. 4. This breakdown prompts the business to consider the problem from different angles and helps prioritise the short-term and the long-term focus.

The problem was broken down to four components, as shown in Fig. 4:

1. Efficacy in number crunching: The conjecture here was that manipulating numbers to different levels of product hierarchy and in general keeping a track of Revenue and Profit & Loss balance-sheets might be difficult. The business, though, uses a specialised planning tool for this and as such, it might not be a great point of concern. However, certain questions here are: "What information is relevant?" and "Is there consistency in information being put out?". These questions originate from the fact that commentary generation is an entirely human process and it is possible that different analysts have slightly different perceptions of what is necessary and/or important.
2. Efficacy in the Art of Writing: This conjecture states that art of writing

and language is the pain-point in the process. Even if the analyst knows 'what' to convey, 'how' to convey it can be bothersome. It also leads into the broader questions of the words, adjectives and adverbs being used to describe certain phenomena and how it is interpreted in the market. As observed from the literature review [Reiter et al., 2005], this is not a trivial matter

3. Efficacy in collaborations with stakeholders: Coordinating with such high number of stakeholders can lead to a degree of inefficiency due to process mismanagement or distributed siloed data. It is also important to note that if this was the primary cause of inefficacy, then the business needs to approach it from a process mismanagement standpoint and not analytics.
4. Generating deeper insights: This is a much broader conjecture about the process. In an ideal scenario, the end point of this process should be commentaries that highlight the deviation from planned scenario and delves deeper into the causality of it. If the business is unable to achieve that, what could be the solution? Building an automated engine or ensuring technical competency or hiring more analysts to allow better time-management can just be some answers to this broader question

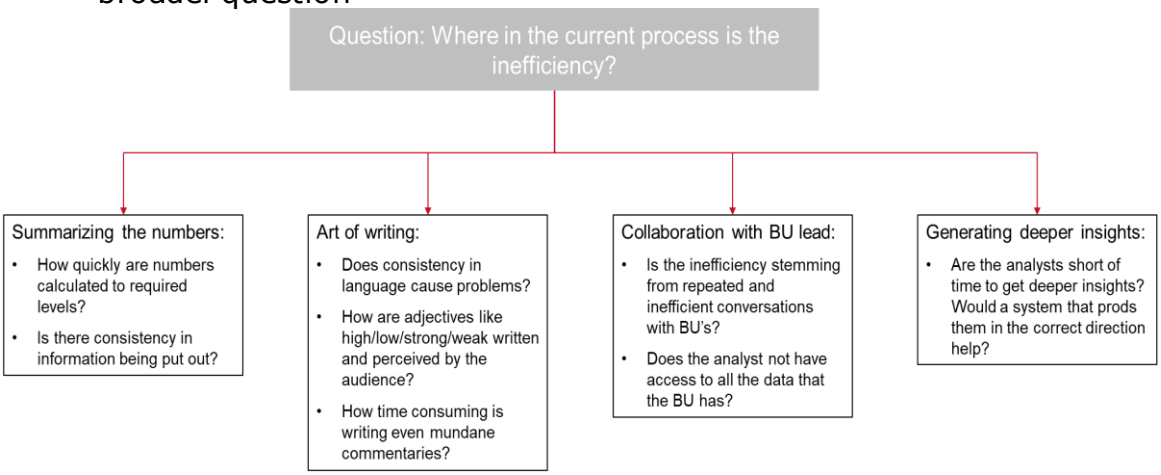

Figure 4: Breakdown of the business problem in process components

After the problem-breakdown, a meeting was scheduled with a financial analyst to get a first-hand account of their process of commentary writing. On discussing the breakdown, the analyst identified points (1) and (2) simultaneously as the areas of best marginal gain in efficiency for the short-term.

The process of writing a commentary was also discussed and it was much more iterative involving higher number of stakeholders than anticipated. The analyst coordinates with the in-house analytics team, along with the Business Unit (BU) folks and Regional teams to start identifying some of the causality of certain numbers. There are also interactions within the BU and regional teams which finally goes out to the Investor Relations team. This team finalises the comments which is finally put out to Wall-Street. Fig. 5 summarises the process. It also highlights where the project can be plugged in. This positioning was chosen for minimum disruption to the process whilst supplying the analyst with a starting point better than a blank slate. It would also hopefully allow the analyst to skip the 1st iteration of generating commentaries and jump straight into having discussions with the concerned

stakeholders.

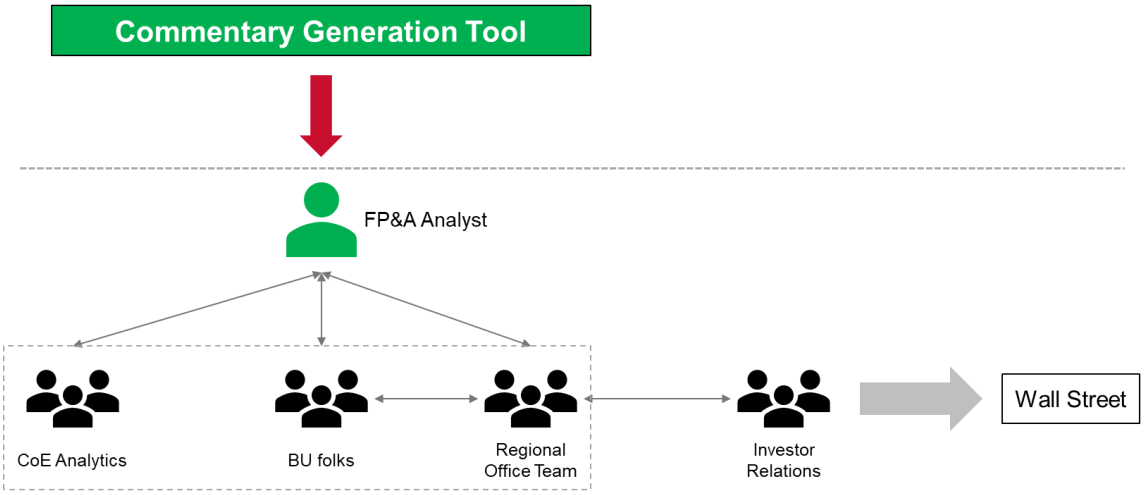

Figure 5: The process of commentary generation

3.2 Know the language

As one of the requirements of this project was that the commentary should be generated in a style replicating the business, it was necessary to analyse the language currently being used by the analysts. To perform this analysis, a suite of tools were used which are discussed below:

3.2.1 Phrase Modelling

There are some phrases (n-grams) in the English language that are spelled out or written separately but are spoken or implied as a single phrase (e.g. 'due to', 'because of'). So, these n-grams are clubbed together to create single phrases which is then analysed as part of a sentence. The below formula is used for phrase modelling:

$$\frac{count(AB) - count_{min}}{count(A) * count(B)} * N = threshold$$

where:

- count(A) is the number of times token (token is another word for vocabulary or word in NLP field) A appears in the corpus
- count(B) is the number of times token B appears in the corpus
- count (AB) is the number of times the tokens AB appear in the corpus in order
- N is the total size of the corpus vocabulary
- $count_{min}$ is a user-defined parameter to ensure that accepted phrases occur a minimum number of times (default value in Gensim's Phrases function is 5)
- threshold is a user-defined parameter to control how strong of a relationship between two tokens the model requires before accepting them as a phrase (default threshold used in Gensim's Phrases function is 10.0)

Application of this phrase model once reveals bigrams in the corpus. Applying it on bigram data reveals trigrams which is an industry standard to check. The threshold in the formula above was decided by bit of trial & error after looking at some phrases which were known to be single phrases and conversely, ensuring that phrases which are separate remain separate. The example in Fig. 6 should clear any confusion with the process:

(Please note that any numbers have been removed from this analysis).

```
THRESHOLD:  1
Original Sentence:  [[redacted], 'vs', 'py', 'since', [redacted], 'launched', 'in', 'sep']
Bi-gramed sentence:  [[redacted], 'vs_py', 'since', '[redacted]_launched', 'in_sep']
Tri-gramed sentence:  [[redacted], 'vs_py', 'since', '[redacted]_launched_in_sep']
```

```
THRESHOLD:  2
Original Sentence:  [[redacted], 'vs', 'py', 'since', [redacted], 'launched', 'in', 'sep']
Bi-gramed sentence:  [[redacted], 'vs_py', 'since', [redacted], 'launched_in', 'sep']
Tri-gramed sentence:  [[redacted], 'vs_py', 'since', '[redacted]_launched_in', 'sep']
```

```
THRESHOLD:  5
Original Sentence:  [[redacted], 'vs', 'py', 'since', [redacted], 'launched', 'in', 'sep']
Bi-gramed sentence:  [[redacted], 'vs_py', 'since', [redacted], 'launched', 'in', 'sep']
Tri-gramed sentence:  [[redacted], 'vs_py', 'since', [redacted], 'launched', 'in', 'sep']
```

Figure 6: Examples of joined phrases with thresholds of 1,2 & 5 respectively (brands redacted)

From the example above, it is quite evident that a threshold of 5 gives the best bigrams (trigrams cannot be found in the example). It has learnt that 'vs_py' is a very frequently occurring combination of words while ensuring that 'product_launched_in_month' is not really a frequently occurring phrase especially when the product and month are specified as well. After visual examination of multiple randomly chosen sentences (not shown here), a threshold of 5 was decided to be the best threshold and all the commentaries were run through the phrase modeller.

3.2.2 Word-Cloud

Word-clouds could seem to be trivial, but they offer a first look at the data and tokens being used. They are an essential feature to start-off and help identify some high-level trends. Word-clouds in this project have been used to try and understand whether there exist differences in themes across dimensions of data. A figure of the word-cloud of all the commentaries is shown in Fig. 7. It is important to notice the prominent phrases. There are 'causality' conjunctions (i.e. conjunctions which are connecting phrases to identify the cause of a certain phenomenon) which come out as most prominent (driven by, due to), there are adverbs or adjectives (strong, high), the phenomenon itself (growth, decline, performance, etc.), some causality as well (sale, promo, promotion) and lastly the reference point of these commentaries (vs_py, vs_nu). These phrases provide some direction of the elements composed in a commentary.

Figure 7: Word-cloud of all commentaries (brands redacted)

Below are some high-level observations regarding analysing word-cloud

across different dimensions of data. Note that some of them are just trivial observations while others point to more systemic commentary-writing process:

- *Commentaries across Years (trivial observation):* 2017 comments focus on JU, FBP & PY; while 2018 has significant NU & PY comparison.
    - *FBP*, *JU*, *NU* & *PY* are comparative plans against the actual sales figures. FBP & PY are full business plans and Previous Year indicating comparison with a model that predicts for the year as well as against actuals of last year. JU & NU and June & November updates to the FBP based on trends observed in the first half of the year
- *Across Different Plans*: No significant difference is observed
- *Across Periods (non-trivial observation):*
    - Although this observation could be an anomaly due to lower number of quarter-to-date (QTD) commentaries, it would seem that QTD is much more focused on reporting growth numbers rather than getting too much into the detail of 'which' product. Words like 'sale', 'momentum', 'decline', 'support', 'promotion', 'campaign' are prominent. The QTD report looks to be a commentary with slightly different focus compared to year-to-date/month-to-date (YTD/MTD).
    - The element of launch of products seems prominent for 'YTD' commentaries compared to other commentaries (words like 'launch in', 'npd' etc.). There are commentaries with launch in MTD as well, but 'npd' seems to be exclusive language for YTD
        - NPD stands for 'new product development'
- *Across Months:* The primary purpose of this word-cloud was to check if the language changes as commentaries move across the year or if there are significant changes in commentary as analysts move through quarters - that does not seem to be the case

3.2.3 Natural Language Toolkit (NLTK) Analysis

NLTK is popular and powerful NLP library. The website [NLTK, 2020] boasts of the following capabilities: "It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries". It is clear the library has a myriad of functions and features. In this project, two of them are utilised. They reveal more information about the commentaries while confirming some of the observations from the section above.

Before jumping into the features though, a brief glance at the lexical diversity of the corpus to understand the richness of tokens is necessary. In the article, "Capturing the Diversity in Lexical Diversity" [Scott, 2013], noted linguist Scott Jarvis identifies six different properties of lexical diversity - variability, volume, evenness, rarity, dispersion and disparity. This paper looks only at the 'variability' and finds that even without stemming/lexicalisations of words, there are ~225,000 tokens across the commentaries with a lexical

diversity of about 3.45% only. It establishes the lack of variety in tokens. For reference, the Book of Genesis has a lexical diversity of 6.23% and a text-chat corpus has a diversity of 13.48% [NLTK, 2020].

Once the diversity was established, the project looked at two features of the library, which are:

- **Word Similarity:** With the most prominent words now known due to word-cloud, the next step was to identify similar words to those prominent words. While there are multiple ways to go about this, *text.similar()* naïve method was deemed the best due to its ease of interpretability. The method takes a word w, finds all contexts w_1w w_2, then finds all words w' that appear in the same context, i.e. $w_1w'w_2$.

```
Similar words to due:
thanks compared leading continues owing from contributed continue
contributing resulting led in back delay top unable is lead support
expected
```

```
Similar words to growth:
performance decline momentum py shortfall fbp sales ju
underperformance is mainly offtake gap overperformance pipeline nu in
trend slowdown promotion
```

Figure 8: Similar words to '*due*' & '*growth*'

If '*due*' is taken as the focus word (from the phrase 'due to' in word cloud), some tokens look to be quite similar is usage as due ('thanks to', 'owing to', 'leading to', 'continues to' etc.) but others looks random ('delay', 'top', 'unable' etc.). Something similar is seen with the word '*growth*'. Other words explored can be found in Appendix 1.

- **Part-of-Speech Tagging:** This exercise involves assigning a tag to each word in the corpus with a "word class" or lexical category. NLTK combines several methodologies and studies from different papers to provide the user a simplified "Universal Tagset". This project uses this tagset to identify the most commonly occurring tokens across each category, which then feeds into the NLG system developed later.

```
[wt[0] for (wt, _) in word_tag_fd.most_common() if wt[1] == 'VERB']
executed in 37ms, finished 12:22:39 2020-07-28

['driven',
 'is',
 'was',
 'offset',
 'expected',
 'has',
 'may',
 'be',
 'phasing',
 'coming',
 'contributed',
 'grew',
 'been',
 'are',
 '+',
 'launched',
 'offtake',
 'including',
 'declined',
 'came',
 'impacted',
```

```
[wt[0] for (wt, _) in word_tag_fd.most_common() if wt[1] == 'ADJ']
executed in 43ms, finished 12:22:38 2020-07-28

['due',
 'strong',
 'lower',
 '+',
 'higher',
 'total',
 'new',
 'general',
 'high',
 'slower',
 'last',
 'low',
 'positive',
 'key',
 'soft',
 'in-store',
 'off-take',
 'operational',
 'good',
 'competitive',
 'negative',
```

Figure 9: Most Frequently occurring 'Verb' & 'Adjective'

Other lexical categories can be found in Appendix 2.

3.2.4 Topic Modelling

Latent Dirichlet Allocation (LDA) is a commonly used unsupervised technique for topic-modelling of text corpora. Introduced in 2003 [Blei et al., 2003], it is a 3-level hierarchical Bayesian model, where documents are represented as a mixture of a pre-defined number of *topics*, and the *topics* are represented as a mixture of the individual tokens in the vocabulary. The number of topics is a model hyperparameter selected by the practitioner. LDA makes a prior assumption that the (document, topic) and (topic, token) mixtures follow Dirichlet probability distributions. This assumption encourages documents to consist mostly of a handful of topics, and topics to consist mostly of a modest set of the tokens. The architecture along with the parameters are part of Fig. 10:

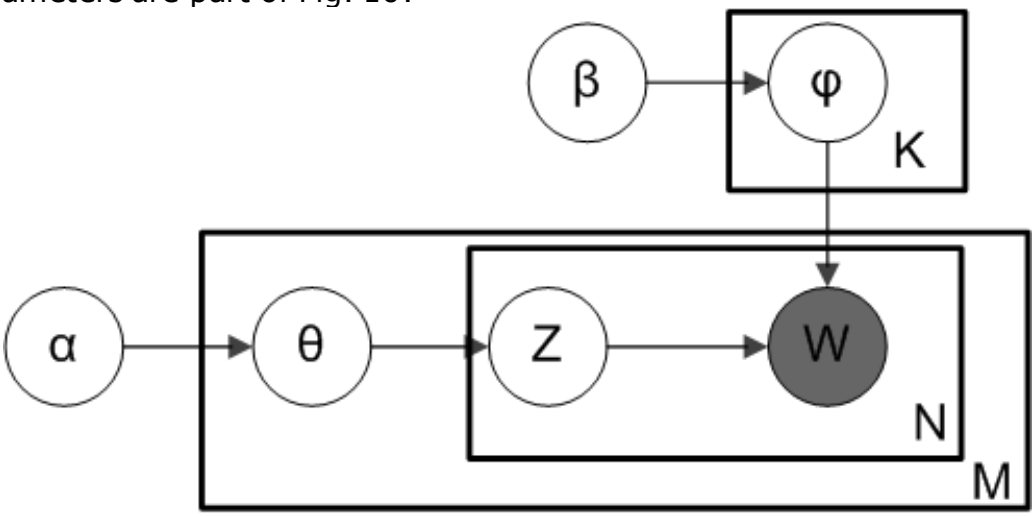

Figure 10: Smoothed LDA from Wikipedia

Above is what is known as a plate diagram of an LDA model where:

- α is the per-document topic distributions,
- β is the per-topic word distribution,
- θ is the topic distribution for document *m,*
- φ is the word distribution for topic *k,*
- z is the topic for the n^{th} word in document *m*, and
- w is the specific word

This project had a very clear aim for performing topic modelling onto the commentaries. It was to classify each commentary based on the topic of their 'intent' and 'depth' i.e. what information is the comment conveying and to what degree is the causality identified for it? Examples are provided below:

- Reporting numbers as facts and generic statements (i.e. Baby powder went down by x%) – Level 1 comment (L1)
- Reporting numbers that dig a bit deeper into the data i.e. goes down maybe a level or 2 to report what caused the movement (i.e. Baby Powder went up by y% due to strong growth in Minor 1 and Minor 2 but offset by Minor 3 due to inventory problem) – Level 2 comment (L2)
- Commentaries that are made from insights through BU-F and for which data is NOT present in consumer sales (i.e. Baby powder continued its growth momentum due to XXXX campaign and YYYY sale along with a favourable climate and weak competition) – Level 3 comment (L3)

3.2.4.1 Topic Modelling Interpretability

LDA models operate on 'Perplexity'. It is a statistical measure of how well a model predicts a sample i.e. It captures how surprised a model is of new data it has not seen before and is measured as the normalized log-likelihood of a held-out test set. Just like log-likelihood of any ML model, its power lies in the ability to allow comparison across multiple models (In this case, comparison across number of topics). But, Chang et al. (2009) presented the first human-evaluation of topic models by creating a task where humans were asked to identify which word in a list of five topic words had been randomly switched with a word from another topic. This work showed some possibly counter-intuitive results, where in some cases humans preferred models with higher perplexity. This served as a motivation for work on '*Topic Coherence*' [Newman et al., 2010].

Topic Coherence measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artefacts of statistical inference. There are six different types of coherency scores, but this project uses the most popular one 'C_v' defined as a measure based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized pointwise mutual information (NPMI) and the cosine similarity.

This project checks to see if classifying the comment corpus into 3 topics generates a high degree of coherency & is interpretable by humans or whether a different number of topics generates better coherency. The results of this process are a mixed bag. It is necessary to note that hyperparameter tuning of filtering tokens was performed using perplexity score, before checking for coherency across number of topics. A list of stop words was added to stop brand and product names or types appearing in the topic-model. The other parameters were thresholds for too rare & too common words (Table 1).

# Topics	No_below (Rarity threshold)	No_above (Commonality threshold)	Stop-words Removed	Perplexity
3	#10	75%	No	-6.49
4	#10	75%	No	-6.485
5	#10	75%	No	-6.508
2	#60	75%	No	-5.039
3	#60	75%	No	-5.015
2	#60	60%	Yes	-
3	#60	60%	Yes	-
3	#30	50%	Yes	-5.64
4	#30	50%	Yes	-5.66
5	#30	50%	Yes	-5.68

Table 1: Perplexity scores across parameters (lower the better)

The rarity thresholds of 30 & 60 were selected as a token found in 0.5% & 1% of all comments being analysed respectively.

From Table 1 above, it appears that 2 or 3 topics with thresholds of 60 and 75% gives the best perplexity, but on analysing the PCA of topic distribution in the case of 3 topics, it becomes very evident that there seems to be 1 huge topic and very small two other topics (shown in Figure 11). This also continues with 60% commonality threshold (for which perplexity is not shown as a similar phenomenon was observed).

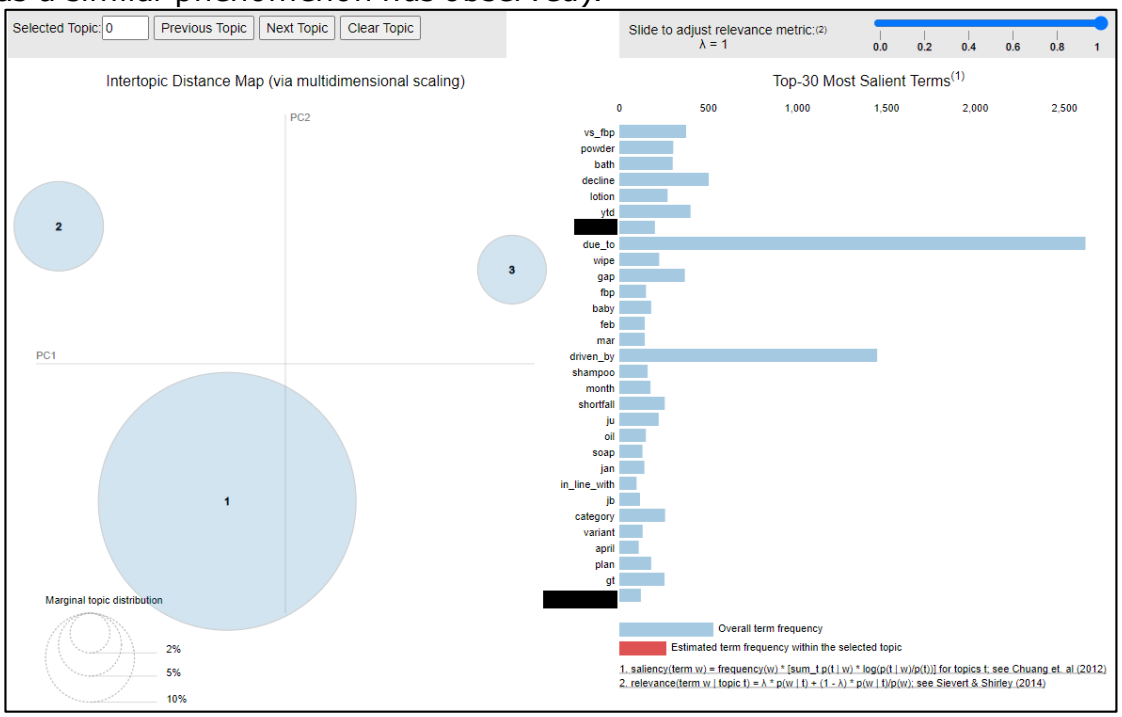

Figure 11: PCA for #60 & 75% thresholds (size denotes marginal topic distribution)

This prompted the switch to check topic distribution with thresholds 30 and 50%, which produced better distributions. So, using these thresholds, coherency (C_v) scores were checked across different number of topics (Fig. 12)

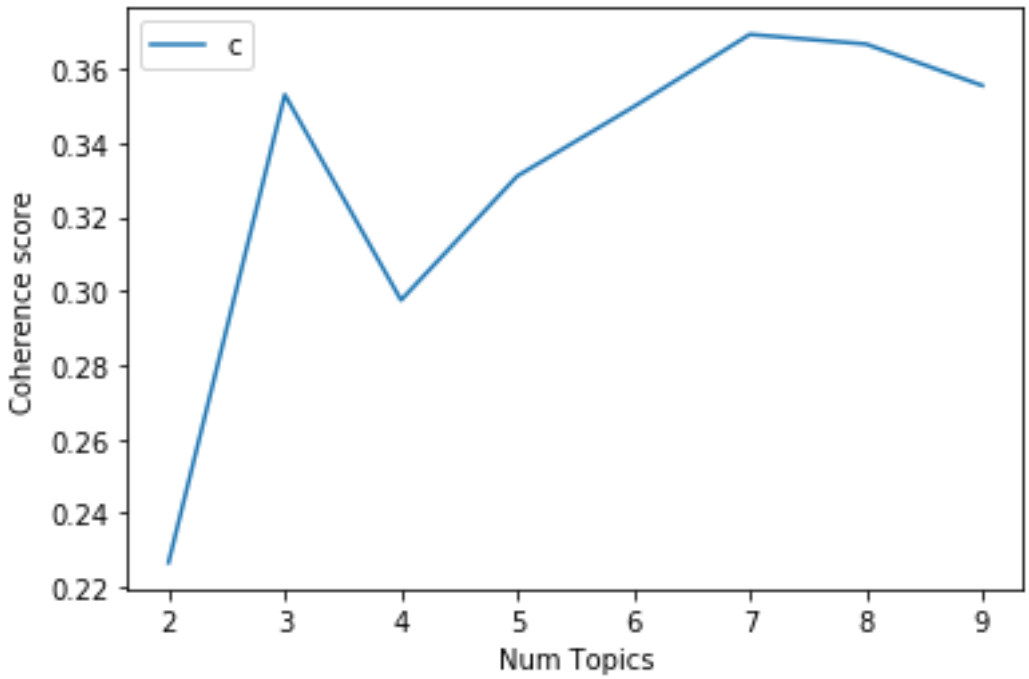

Figure 12: Coherence Score (C_v)

It is critical to note that the graph observed in Fig.12 is not ideally what is seen in practice. Normally, the coherency graphs start low, reach a peak at certain gradient and then flatten or point down. From this graph, 3 topics and 7+ topics appear to show best human coherency. But, the range of coherency values is troubling, at mere maximum of ~0.37, this is significantly below the normal industry standards of 0.5. To understand whether coherent topics can be extracted at such a low score, the model with 3 topics was explored in detail using 'gensim' library's pyLDAvis function (Fig.11 was also extracted using the same function).

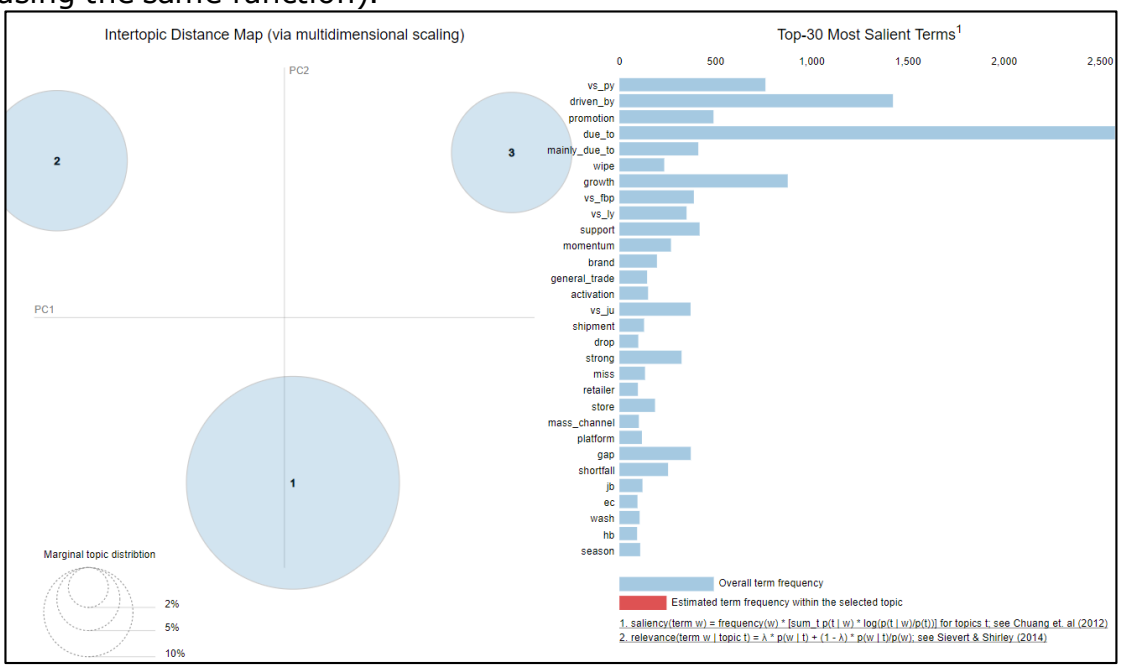

Figure 13: 2-D PCA for #30 & 50% thresholds of 3 topics

This visualisation is a very powerful tool for topic modelling. Along with giving an idea of how separated the topics are (the farther the circles, the better reach of topics being captured), the visualisation also allows the user to understand the most prominent (or salient) tokens of each topic. A slider at the top right also allows the user to adjust the importance of a token based on term-frequency. Similar to introducing the '*Inverse Document Frequency*' in *tfidf* to measure token importance, this measure allows the user to adjust the importance of a token to a topic based on its occurrence and proposes a novel measure 'relevance'. Introduced in a study in 2014 [Sievert et al., 2014], it concluded that ranking terms purely by their probability under a topic is suboptimal for topic interpretation.

With regards to this project, the exploration of tokens at different lambda values (relevancy measure) and across different topics did not reveal a lot of coherency. It was also very difficult to try and match the topics to the initial premise of comment classification into the 3 levels of L1, L2 & L3. So, this exercise was dropped with a conclusion that either the model needs more data to train on or in more probability, the lexical diversity between the 3 levels of comment is not enough to be picked up by this model.

3.2.5 Word2vec model

Developed by Google, word2vec (and an extension doc2vec) models were

the one-of-the-first representations of the power of neural network for NLP tasks [Mikolov et al., 2013], with multiple extensions explored in the subsequent paper [Mikolov et al., 2013]. The paper uses a 2-layer LSTM architecture to present each word in an n-dimensional vector space, where learning is achieved through two different means: Continuous Bag-of-Words (CBoW) or Continuous Skip-gram model. The former tries to reduce the loss function by correctly identifying a token using context words around it whilst the latter tries to predict the context words using one token which is in focus at a time. The skip-gram model was used in this project as it is perceived to have better contextual grasp of the vocabulary as even rare words with similar context to very common words can be recognised as similar using this technique which does not happen for CBoW. This gives the model a very powerful ability to learn words (and their usages) in context to the words around it!

After multiple iterations over window size (i.e. the number of context words around the word in focus) and vector size, it was determined that a window size of 3 and vector count of 50 gives the best results (Fig. 14). To arrive at this configuration of "best" results, most commonly occurring words were picked from the word-cloud and most similar words were obtained for them (from the word2vdc algorithm), which were then checked against human knowledge. While this was a less-than-ideal technique to arrive at the best parameters, it clearly highlighted the lack of sufficient training data (Fig. 15 shows this) for the algorithm to converge at an appropriate minimum and provide similar answers to identical queries.

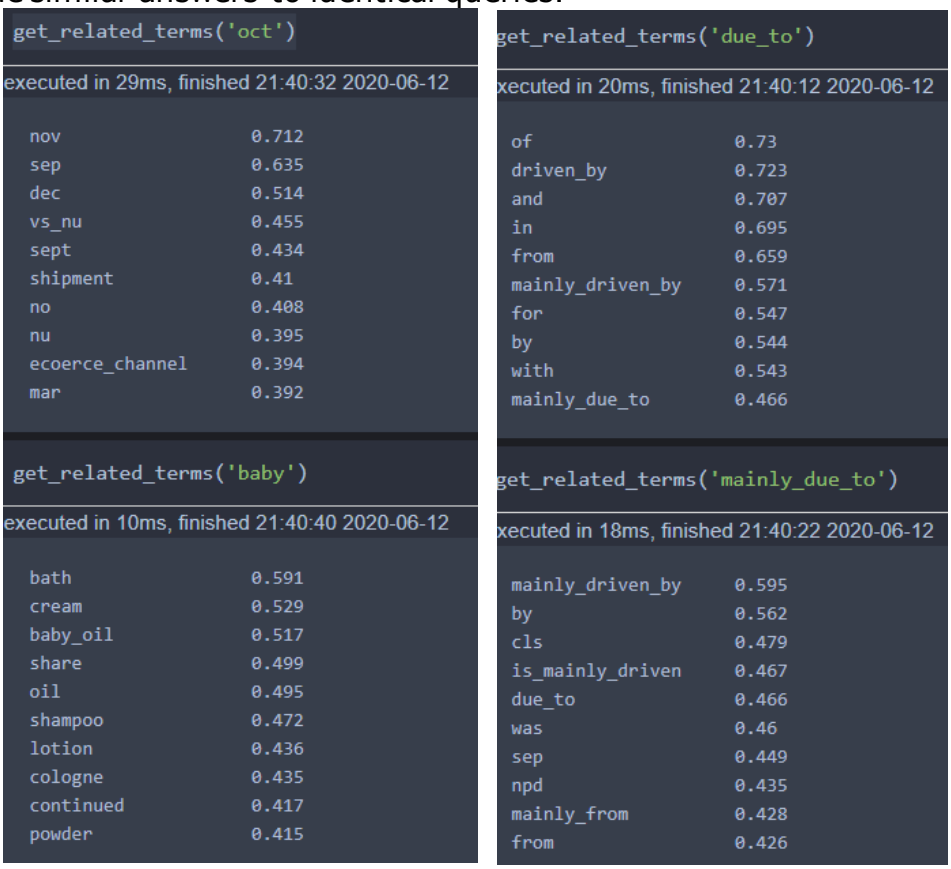

Figure 14: Similar words for window size 3 & vector size 50 of word2vec

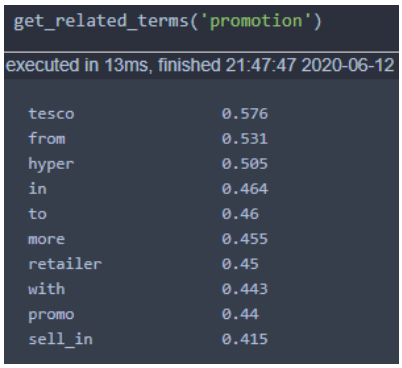

Figure 15: Similar word to 'promotion' for the parameter values as Fig. 14 – 'tesco' is a retailer & should not be related to promotion

This technique is a more powerful than the lexical similarity detailed in *'NLTK Analysis'* section to find similar words and improve the lexical diversity once the generation part of the project begins.

3.3 Generate the commentary

As explored in the literature review section, there are currently two techniques for NLG currently used: the pipeline architecture & the E2E neural network-based solution. The benefits and drawbacks of both these techniques are explored in detail in the same section. For this project, the 'pipeline architecture' paradigm was decided due to three reasons:

1. Information Hallucination: The E2E paradigm can hallucinate information, which is unacceptable in this project
2. Lack of structured data: The E2E architecture requires data to be fed in a very structured manner with the model being able to learn the connection between input data and output commentary. While dealing with so many regions and different product-groups with the decision on each commentary being taken by an analyst, the input data (which includes certain facts, their causalities and magnitude) is not structured
3. Lack of training data: A glimpse of the lack of sufficient training data is seen in the word2vec model. Although few-shot NLG models exist to circumvent this very issue, the issues mentioned in points 1 & 2 above made this a less appealing prospect to go forward this

While the three points discuss the disadvantages of E2E network, the biggest advantage of the pipeline architecture is also important to discuss, which is 'flexibility'. Once the phrases picked out from the Section 3.2 are decided, they are implemented using a Python realiser (i.e. the last stage of pipeline architecture). It might be worth nothing that Python does not have many open-source language realisers, which can be a significant liability to this language and the field in general.

3.4 Building a flexible system – Dash-Plotly based front-end

One of the parameters of the project was the requirement of being able to generate commentaries in a robust input environment. That required coding in a manner where none of the columns of the input data could be hard-coded.

To allow for ease of using the commentary-generation application, a front-end was decided to be set-up; with the primary purpose of taking in certain

input from the user and using that it to manipulate data and generate the commentaries. For this purpose, it was decided that a Dash frontend would be used. This would allow the entire application to be restricted to just a software – Python, thus reducing user-effort to operate this program. It also allows for any python user to quickly incorporate any necessary changes for future commentaries.

According to the website, Dash is a Python framework for building analytical web applications. No JavaScript required. Built on top of Plotly. js, React and Flask, Dash ties modern UI elements like dropdowns, sliders, and graphs directly to your analytical Python code. With Dash Open Source, Dash apps run on your local laptop or workstation, but cannot be easily accessed by others in your organization. Fig. 16 shows a snippet of the simple but effective front-end built for this project.

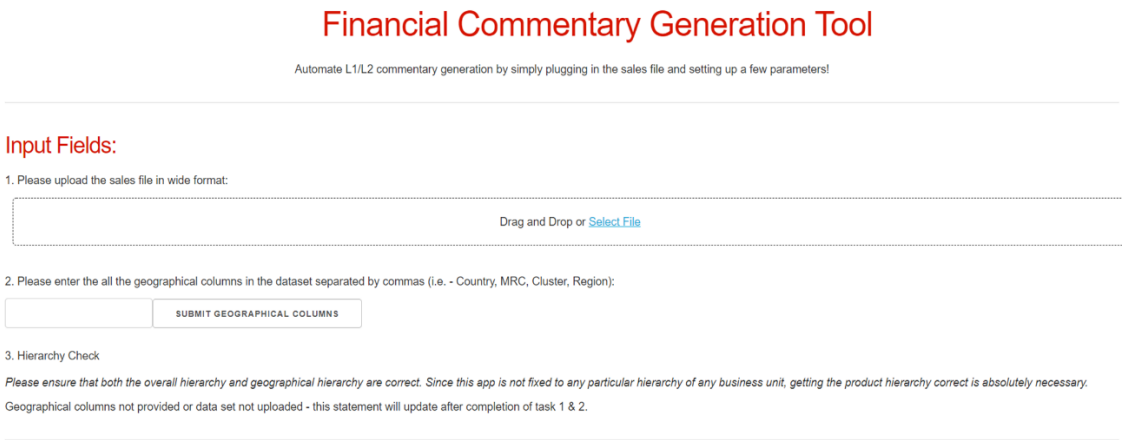

Figure 16: Snippet of Dash based front-end for the user on any browser

3.5 When is a commentary written?

Beyond the aspect of writing a commentary, the project also unsuccessfully tried to model 'when' a commentary is being a written. Discussions with financial analysts revealed the commentaries are written when certain thresholds of deviation (between actual and predicted figures) are crossed, but this threshold might vary based on different regions, levels of products of even different business units. This project still attempted to understand whether information from a combination of inputs can be used to arrive at a decision of writing a commentary.

Previous works on this project had tried to use different regression or tree-based techniques to model when a commentary is necessary, but due to the lack of any promise in the results of these commentaries, a different approach was tried. This approach is normally employed to determine mechanical failures in machines when input from multiple sensors are combined. The steps are as follows:

1. Input from required values are readied in a tabular format. This project used revenue deviation of the actual against predicted as well as deviations of certain other P&L channels such Brand Marketing Expenses, Inventory adjustments & Transportation value
2. The input is converted to vectors using Principal Component Analysis of a dimension capturing sufficient variation in data. For this projects, 12 sets input data were converted to 6 vectors capturing 98% of the total variation in the dataset. The PCA converts the data into vectors

such that only the datapoints which show the maximum variation after being scaled are part of the vector, thus retaining most of the useful information whilst throwing out the non-useful stuff

3. Distance between points is calculated and plotted on a frequency distribution chart. The 'distance' here can be any distance which measures the physical between two points. In this project, Mahalanobis distance was used as it works better on a non-spherical data. The Mahalanobis distance captures the distance of a point from the center by also incorporating the range of values of an axis along a particular vector (e.g. in an ellipse, an anomalous point across the longer radius will have different distance than across the shorter radius from the center and Mahalanobis distance takes care of this)
4. A frequency distribution of the distance is plotted. In an ideal scenario, the normal points (in the case of this project, points with no commentary) should fall within a threshold distance (decided by the user). Any point beyond this threshold is an anomaly (or needs a commentary). From Fig.17, it is clear that no such threshold could be derived between distances where commentary was present vs. absent (Fig. 17). This went on to show that there is indeed no broad brush or rule that can be used to identify whether a commentary is necessary given a set of data-points. The decision to write any comment is based on the analyst's judgement and rules defined while the exercise is underway.

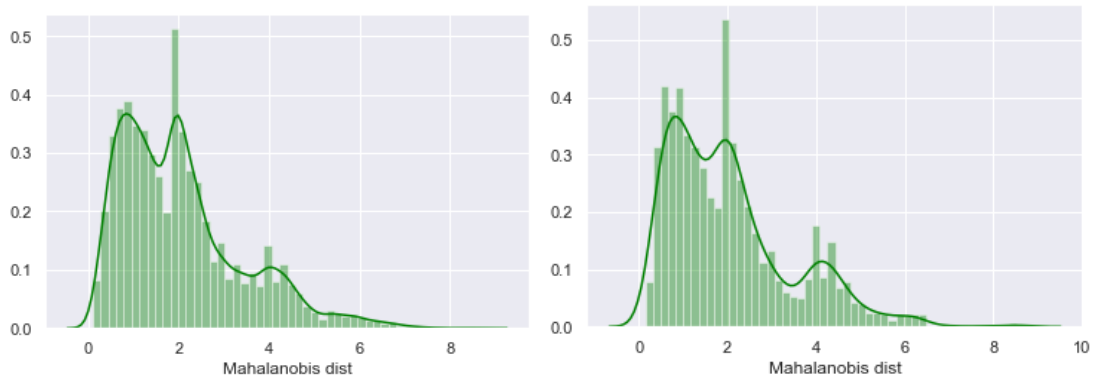

Figure 17: Frequency distribution of distance of where commentary is absent vs. present respectively – the distributions are almost identical and do not allow for any threshold distance to be set that separates the 2

4. Discussion

With increasing focus on deriving value from unstructured data and rapid advancements in natural language processing, organisations are moving swiftly to take advantage of these capabilities. But, outside of academic settings, implementation of accurate and advanced NLG systems is still very few and far between. This problem is compounded further by the lack of any advanced natural language realiser in Python or R, or even a project in development which offers any promise of such a feature. This leaves the community overly reliant on either using basic realisers or using Neural Networks, which can have problems discussed in detail in the earlier sections.

In this project, current technologies with their capabilities and limitations are leveraged to understand the language being used currently by the analysts in their commentaries and replicate the style of writing to a certain extent. It also hopes to provide a comprehensive package, with each component

explored in detail, to build an NLG framework with exploration into the language.

5. Impact

In terms of its business impact, the immediate short-term value is the manhours and effort saved to complete repetitive trivial commentary tasks. This, in turn, could give the analysts more time to improve upon their current process and explore deeper into the causality of certain phenomena, translating into improved quality of commentaries over the long-term with the hope to drive better decision-making with these commentaries. It also lays a foundation of adding more analytical tools to the commentary-generation process with further iterations.

A vital component missing from the current product is a comprehensive review system. As explored earlier, there are considerable problems with using metrics such as BLEU for data-to-text generation systems, which make it difficult to evaluate the 'quality' of the commentaries. Human evaluations are necessary to understand whether the task the project set out to do is being done satisfactorily or not, but it can be expensive and time-consuming. A continuous evaluation framework which incorporates new trends in analyst languages and corrections made on top of the output of this system would be necessary to ensure longevity of this project.

6. Conclusions and Further Work

In conclusion, a data-to-text NLG framework has been implemented after performing a detailed analysis on the past language. For ease-of-use and flexibility to use across different environments, the system has a web based front-end created from Dash-Plotly which can be easily tweaked to incorporate any changes. The framework does not use E2E neural-network paradigm, but rather the older but more reliable pipeline architecture to generate language. It also highlights the very limited open-source options available today to generate coherent language at any level.

From a business viewpoint, there are some immediate next-steps necessary to incorporate feedback and understand the quality of the language generation with a more long term-plan to enable adoption of this framework in the BU, add more components to identify anomalies and causalities and then incorporate them into the commentary. They are succinctly summarised below:

1. Showcase current capabilities and try to incorporate feedback from the end-user(s)
    a) Set up interviews with analysts to understand their level of satisfaction with the generated commentaries and understand where they would like to improve
    b) Set up meetings with multiple stakeholders to raise awareness and get their buy-in
2. Diversify data sources and work towards generating L2 level commentaries
    a) Further discussions with the analyst on the sources of information and how it is processed to arrive at the comment
3. Incorporate smart analytical techniques into the process to increase value-proposition over generic commercial NLG tools
4. Identify a framework for continuous assessment and improvement

Bibliography

C.E. Shannon, *A Mathematical Theory of Communication*, The Bell System Technical Journal, 1948

Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei Jing, *BLEU: a Method for Automatic Evaluation of Machine Translation*, 10.3115/1073083.1073135, 2002

Chin-Yew Lin & Eduard Hovy, *Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics*, 2003

Reiter, E., & Belz, A., *An Investigation into the Validity of Some Metrics for Automatically Evaluating Natural Language Generation Systems,* Association for Computational Linguistics, 2008

Ehud Reiter and Robert Dale, *Building applied natural language generation systems*, Natural Language Engineering, 3(1):57–87, 1997

Ehud Reiter, *NLG vs. Templates*, 1995

Kees van Deemter, Emiel Krahmer, Mariet Theune, Real vs. template-based natural language generation: a false opposition?, 2003

François Portet, Ehud Reiter, Jim Hunter, Somayajulu Sripada, *Automatic Generation of Textual Summaries from Neonatal Intensive Care Data*, AIME, Amsterdam, Netherlands. pp.227- 236. ffhal-01006111f, 2007

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, Ian Davy, *Choosing words in computer-generated weather forecasts,* Artificial Intelligence, Volume 167, Issues 1-2, Pages 137-169, 2005

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, *Attention Is All You Need*, arXiv:1706.03762v5, 2017

Microsoft, *Turing-NLG: A 17-billion-parameter language model by Microsoft*, Microsoft Research Blog, 2020

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei, *Language Models are Few-Shot Learners*, arXiv:2005.14165v3, 2020

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, William Yang Wang, *Few-Shot NLG with Pre-Trained Language Model*, arXiv:1904.09521v3, 2020

Sam Wiseman, Stuart M. Shieber, Alexander M. Rush, *Challenges in Data-to-Document Generation*, arXiv:1707.08052, 2017

Ratish Puduppully, Li Dong, Mirella Lapata, *Data-to-Text Generation with Content Selection and Planning*, arXiv:1809.00582v2, 2019

Mokhtari, Karim El, John N. Maidens, Ayse Basar Bener, *Predicting Commentaries on a Financial Report with Recurrent Neural Networks*, Canadian Conference on AI, 2019

Jarvis, Scott, *Capturing the Diversity in Lexical Diversity*, Language Learning. 63: 87–106, 2013

David M. Blei, Andrew Y. Ng, Michael I. Jordan, *Latent Dirichlet Allocation*, Journal of Machine Learning Research 3 (993-1022), 2003

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, David M. Blei, *Reading Tea Leaves: How Humans Interpret Topic Models*, In Proc. of NIPS, 2009

David Newman, Jey Han Lau, Karl Grieser, Timothy Baldwin, *Automatic Evaluation of Topic Coherence,* Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 100–108, 2010

Carson Sievert, Kenneth E. Shirley, *LDAvis: A method for visualizing and interpreting topics*, 10.13140/2.1.1394.3043, 2014

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv:1301.3781v3, 2013

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, *Distributed Representations of Words and Phrases and their Compositionality*, arXiv:1310.4546v1, 2013

Appendix 1:

Words for which similarity was explored using NLTK – all the words here come from prominent words in the word-cloud.

```
Similar words to due:
thanks compared leading continues owing from contributed continue
contributing resulting led in back delay top unable is lead support
expected
Similar words to mainly:
py is ju fbp shortfall by growth nu ly decline was target
underperformance gap coming primarily largely majorly and of
Similar words to launch:
pipeline launched decline npd phasing deletion underperformance growth
overperformance oos delisting sales delay start shipment availability
gap momentum performance softening
Similar words to driven:
offset impacted contributed caused short supported py ly fbp declined
ju target is mainly by grew led mitigated came was
Similar words to growth:
performance decline momentum py shortfall fbp sales ju
underperformance is mainly offtake gap overperformance pipeline nu in
trend slowdown promotion
Similar words to decline:
growth gap shortfall underperformance mainly overperformance is py
performance increase ffw fbp pipeline oos drop promotion delay powder
wipes launch
```

```
Similar words to promotion:
promo support growth promotions pipeline increase inventory and
performance sales decline [redacted] powder atl delay shortfall the [redacted]
[redacted] gap
Similar words to promo:
promotion promotions the pipeline atl digital lotion soap py trade
[redacted] sales twin [redacted] ec promos inventory in cologne clean
Similar words to gap:
decline growth shortfall the underperformance this mainly drop
momentum brand overperformance target which sales increase is py by
pipeline oos
Similar words to support:
promotion in phasing campaign pipeline stock growth investment
activation performance lotion and impact decline powder the oos sales
delay [redacted]
Similar words to impact:
growth phasing performance and pipeline support delay discontinuation
promotion momentum sales underperformance shortfall effect in strong
deletion softening decline lack
Similar words to offset:
driven impacted supported contributed fbp short caused py drive ly ju
the in declined led offsets compensated cover complemented of
Similar words to impact:
growth phasing performance and pipeline support delay discontinuation
promotion momentum sales underperformance shortfall effect in strong
deletion softening decline lack
```

Appendix 2:

Most frequently occurring words for lexical categories (apart from adjectives & verbs which can be found in Fig. 9):

```
[wt[0] for (wt, _) in word_tag_fd.most_common() if wt[1] == 'NOUN']
```

executed in 37ms, finished 12:22:37 2020-07-28

```
['%',
 'py',
 'growth',
 'performance',
 'support',
 'channel',
 'ju',
 'trade',
 'promotion',
 'inventory',
 'sales',
 'fbp',
 'momentum',
 'baby',
 'sale',
 'impact',
 'campaign',
 'launch',
 'ly',
 'decline',
 'gap',
```

```
[wt[0] for (wt, _) in word_tag_fd.most_common() if wt[1] == 'ADV']
```

executed in 34ms, finished 12:22:40 2020-07-28

```
['mainly',
 'partially',
 'also',
 'well',
 'as',
 'not',
 'especially',
 'still',
 'nu',
 'ahead',
 'majorly',
 'however',
 'where',
 'ytd',
 'slightly',
 'primarily',
 "n't",
 'yet',
 'back',
 'forward',
 'largely',
```

```
[wt[0] for (wt, _) in word_tag_fd.most_common() if wt[1] == 'CONJ']
```

executed in 164ms, finished 12:22:42 2020-07-28

```
['and',
 '&',
 'but',
 'ytd',
 'plus',
 'nicorette',
 'or',
 'nu',
 'tylenol',
 'neu',
 '/',
 'npd',
 'yet',
 'lqtr',
 'ssd',
 'less',
 'neutrogena',
 'labo',
 '-0.4',
 'npi',
 '/+0.2',
```