Churnalist: Fictional Headline Generation for Context-appropriate Flavor Text

Judith van Stegeren
Human Media Interaction
University of Twente
Enschede, The Netherlands
j.e.vanstegeren@utwente.nl

Mariët Theune
Human Media Interaction
University of Twente
Enschede, The Netherlands
m.theune@utwente.nl

Abstract

We present Churnalist, a headline generator for creating contextually-appropriate fictional headlines that can be used as 'flavor text' in games. Churnalist creates new headlines from existing headlines with text modification. It extracts seed words from free text input, queries a knowledge base for related words and uses these words in the new headlines. Churnalist's knowledge base consists of a dataset of pre-trained word embeddings, thus requiring no linguistic expertise or hand-coded models from the user.

Introduction

The field of natural language generation (NLG) investigates how texts can be created automatically. NLG systems have been used to transform data, such as weather data, football match statistics and intensive care data, into texts for a specific audience. NLG is also used for generating fictional or creative text, such as poetry (Gonçalo Oliveira 2012), lyrics (Bay, Bodily, and Ventura 2017), advertising slogans (Gatti et al. 2015), and character dialogue in games (Lukin, Ryan, and Walker 2014; Schlünder and Klabunde 2013). It is in the latter type of applications that NLG has similar research goals as computational creativity, i.e. supporting or even completely replacing a human in the execution of a creative task.

In this paper, we present *Churnalist*, an interactive system for generating newspaper headlines for a given context. Our system is meant for generating fictional headlines that can be used in games. Most headline generators take a newspaper article as input and summarize it in one sentence. In contrast to these systems, Churnalist accepts any type of text as input and generates headlines based on nouns extracted from the input text. By reusing nouns from the input text in the generated headlines, we aim to make the headlines context-appropriate, by which we mean that readers will believe that the headlines are related to the input text. We want to exploit the human tendency to see connections between texts (input text and headlines) where there are none.

There are various games that use fictional news (in the form of headlines or newspaper articles) to provide narrative context to the player. For example, in city simulation game SimCity 2000 (Maxis 1996), the player has access to newspaper articles with information about important city issues, disasters and new technologies. Similarly, Cities Skylines (Colossal Order 2017) features a fictional social media website called 'Chirpy', where virtual citizens of the player's city express their (dis)satisfaction with the player's performance as mayor and city planner. In Deus Ex: Human Revolution (Eidos Montral 2011), the player can find ebooks and newspapers that provide background information on the social unrest that is driving the game's main storyline. Idle game Cookie Clicker (Thiennot 2013) has a news ticker with randomly generated headlines reflecting the player's game progress.

The fictional newspaper articles and headlines can be seen as examples of *flavor text*, i.e. text that is not essential to the main game narrative, but creates a feeling of immersion for the player. This is especially important for role-playing games and simulation games, as it gives players the impression that the virtual world they are interacting with is a living and breathing world.

Writing flavor text is a time-consuming task for game writers. Text generation can be a solution to this problem. Most games that incorporate text generation use simple, manually created templates or canned text. More complex NLG techniques rely on linguistic models, which often take considerable effort to create and require linguistic expertise. Statistical linguistic models can be created automatically from a dataset of texts. However, generators with underlying statistical models offer less fine-grained control over the output. Canned text and simple templates offer a balance between control over the output and ease of use, but have the disadvantage that players will figure out the underlying templates after playing the same game for a while, or after replaying the game (Backus 2017). We think that NLG techniques other than canned text and simple templates are worth investigating in the context of game development, especially data-driven approaches to text generation, as these can overcome the need for expensive, handcrafted language models. We propose a system that can generate fictional headlines in order to support game writers in the task of writing flavor text.

In the next section, we discuss related work. Then, we present Churnalist and describe the system goal, the architecture and the generation steps in detail, together with an example. Finally, we discuss our results and describe some ideas for future work.

Related work

In this section, we discuss work related to headline generation, text generation for games and generative systems that take context into account.

Headline generation

Headline generation is often seen as a document summarization task, where headline generators take a full article text as input and return a headline that describes the most salient theme or the main event of the text. The literature distinguishes between extractive summarization, e.g. (Jing 2000), and abstractive summarization approaches. Contrary to extractive systems, the output of an abstractive system does not have to correspond to a sentence from the input text. Abstractive headline generation systems may be rule-based (Dorr, Zajic, and Schwartz 2003), statistics-based (Banko, Mittal, and Witbrock 2000) or based on machine learning (Colmenares et al. 2015; Shen et al. 2017), with the latter winning in popularity in recent years.

Headylines (Gatti et al. 2016) is an example of a headline generation system that focuses on the creative side of writing headlines. It can be used to support editors in their task of writing catchy news headlines. Given a newspaper article text as input, it extracts the most important words from the text and uses these as seed words for generating a large set of variations on well-known lines, such as movie names and song lyrics. This research is a good example of combining NLG with techniques from computational creativity.

Text generation for games

Text generation for games is a form of *procedural content generation* (PCG). Procedural content generation, which refers to the creation of content automatically through algorithmic means, is a relatively new addition to the field of artificial intelligence. PCG for games studies the algorithmic creation of *game contents*, defined by Yannakakis and Togelius (2011) as all aspects of a game that affect gameplay but are not non-player character behavior or the game engine itself, such as maps, levels, dialogues, quests, music, objects and characters. Text generation techniques can be used for generating dialogue, stories, quests and flavor text for games. Although including generated game text in video games is winning in popularity, these texts are often generated with simple NLG techniques, such as canned text and simple templates.

On the other hand, within the natural language generation field, there are various publications that list game text as a possible application (Schlünder and Klabunde 2013; Strong et al. 2007; Lukin, Ryan, and Walker 2014). However, there are few cases where the implemented system is actively used in a games context. One example is Caves of Qud (Freehold Games 2018), which combines techniques from PCG and NLG to create a unique game world for every play-through. The developers of Caves of Qud used a hand-written knowledge base for their text generator, which links in-game themes to a set of words and phrases (see next section). In our research, we use a knowledge base for a similar purpose: to link seed words from the input text to a set of related words. Instead of creating it manually, we used word embeddings as the basis for our knowledge base.

Generative systems and context

For Churnalist, we were inspired by how other generative systems create text for a given context. Context is a slippery notion. Within PCG for games, generated artifacts are judged together with the rest of the assets of the game for which they were generated. In the case of Churnalist, context means the input text and, more generally, the game from which the input text is taken and for which the headlines are generated.

In slogan generation it is also important for the generated texts to fit a given context or domain. In BRAINSUP (Özbal, Pighin, and Strapparava 2013), the generated slogans must fit a domain for which the user manually supplies keywords as input to the system. In BISLON (Repar et al. 2018), keywords are automatically extracted from two sets of documents representing two domains that the generated slogans must fit. The pool of extracted keywords is expanded using FastText embeddings (Bojanowski et al. 2017). The keywords are then used to fill the slots in templates ('slogan skeletons') derived from a corpus of slogans. The BISLON approach to extracting and expanding the set of keywords is very similar to that of Churnalist, as we will see in the following sections. One way in which Churnalist differs from both BRAINSUP and BISLON is that those systems derive syntactic patterns or templates from a corpus and then fill their slots, whereas Churnalist takes the original texts from a corpus and applies word substitution to them. In that respect, Churnalist is more similar to the transformation-based approach to lyrics generation proposed by Bay, Bodily and Ventura (2017).

Another system related to Churnalist is O Poeta Artificial 2.0 (Gonçalo Oliveira 2017), a bot tweeting poems that are generated for trending hashtags on Twitter. It uses hashtags as topical seed words to generate poems that fit the hashtag. The bot is based on PoeTryMe (Gonçalo Oliveira 2012), a poem generation framework for Portuguese, which uses external data sources to enrich its output, such as a database of Portuguese poems, a semantic graph and lexical datasets. Churnalist has multiple things in common with O Poeta Artificial: both generate text for a specific context, work with seed words and external semantic resources, and need a method to deal with out-of-vocabulary words in the input.

Caves of Qud's text generation (Grinblat and Bucklew 2017) influenced our design for Churnalist as well. The game generates fictional biographies for mythical non-player characters called sultans. These biographies consist of randomly generated fictional events from the life of the sultan, such as starting a war, acquiring a mythical weapon or forging an alliance. To infuse a sense of coherence in these biographies a domain, such as 'glass','jewels', 'ice' or 'scholarship' is assigned to each sultan. To tie the life events in the biography together, the generator incorporates domain-specific elements in each event.

Players of Caves of Qud will interpret the randomly generated biographies as coherent narratives, thereby creating their own logical explanation for the overarching theme in each biography. The developers call this human tendency to perceive patterns 'apophenia'. It is related to the 'charity of interpretation' effect studied by Veale (2016), who found that "readers will generously infer the presence of meaning in texts that are well-formed and seemingly the product of

an intelligent entity, even if this entity is not intelligent and the meaning not intentional." If humans see a text in a well-known form (or *container*), they are disposed to attribute more meaning to the text than it actually contains. A similar effect is the Eliza effect described by Hofstadter (1995), who noticed that humans will attribute intelligence or empathy to (text-producing) computer systems. This approach to evoking context is also related to the 'intention' aspect of framing information (Charnley, Pease, and Colton 2012) in computational creativity. With Churnalist, we want to exploit this effect too: by incorporating words from the input text in the output, we hope that readers will perceive the generated headlines as coherent with the input.

Description of Churnalist

Churnalist is a system for generating fictional headlines that are context-appropriate for the textual input. In this section, we discuss the goal of the system and the requirements for the output. We elaborate on the technical design of the system and provide a running example.

System goal

Game writers can use Churnalist for generating flavor text for video games, in the form of headlines. Instead of taking newspaper article texts as input, as is common practice for headline generators, Churnalist accepts user-supplied free text as input, in the form of English sentences from a game. For example, see the one-sentence input in Figure 1.

"Mario must save Princess Peach from Bowser's castle."

Figure 1: An example of valid input text. The names and noun phrases that Churnalist will incorporate in the output headlines are underlined.

Churnalist extracts a set of seed words from the input and creates new headlines by doing word-substitution on headlines from a database. The seed words consist of words from the input. We expand the set of seed words by querying a vector space of word embeddings for vectors close to the words from the input. By using words that have a link with the input text, or *context words*, we generate headlines that fit the context that is represented by the input. By inserting context words in the headlines from the database, we hope to exploit the Eliza effect (Hofstadter 1995), apophenia (Grinblat and Bucklew 2017) and the charity of interpretation (Veale 2016) in readers: readers should think that the headlines are related to the context. Churnalist's output is a set of fictional headlines, like in Figure 2. A more extensive example of using game text as input is provided in Figure 4.

Using free text input makes Churnalist usable for different games and different topics. Regardless of the content or the type of game, as long as the input text contains content words (nouns), Churnalist will extract these from the input and use them as seed words to generate headlines. Churnalist was developed using publicly available datasets, open source libraries and only simple text modification techniques, so that for future users no linguistic expertise is required.

Mario apologises to mother involved in car crash
Mario injured after Sicily volcano triggers earthquake
Mario says Arsenal return vs Qarabag was 'emotional'
Mario: 'My marriage is over because I voted to leave the EU'
Princess Peach unveils world's first Chromebook with AMD processors
Bowser's castle retains Border-Gavaskar trophy after cleaning up Australia on day five

Figure 2: Example output text for Churnalist, given the input text in Figure 1.

For Churnalist's output, we adopt similar requirements as Gonçalo Oliveira (2012):

1. The output texts must look like headlines. We are not generating news article texts. The content of the headlines does not have to be realistic or ground in reality. On the contrary: we aim for fictional output, as well as output that is not literally present in the database of headlines (for copyright reasons).
2. Headlines must be grammatical.
3. Headlines must feel context-appropriate (coherent, meaningful, relevant) for the input text to a not-too-discerning, not overly critical reader.

Architecture

We have implemented a prototype system with the architecture shown in Figure 3. Churnalist has a modular pipeline design so that every subtask can be implemented according to the requirements of the user, to make the system as flexible as possible. The pipeline consists of three modules, one for each step in the generation process. At the end of each step, the user of Churnalist can manually filter the output of the system, thus fine-tuning the nouns and noun phrases that are used in later generation steps.

The first module, the *keyword extractor*, reads the input text and extracts the most important words. These words are the *seed words*. The second module takes the list of seed words and expands this with a set of loosely related words, gathered from the *knowledge base*. The seed words and the related words form the set of *context words*. The *substitution module* takes a random headline from a headline database, runs it through a dependency parser and substitutes parts of the sentence with context words. The resulting new headline is the output of the system. Users can generate multiple headlines from one input text; the number of possible results is determined by (1) the number of seed words in the input text, (2) the size of the headline database and (3) the size of the set of user-approved context words.

In the rest of this section, we describe these steps in more detail and provide an example. Figure 4 shows an input text taken from a dilemma-driven serious game. It features both a situational description and some lines of NPC text. The rest of this paper will feature examples that were generated with this input text.

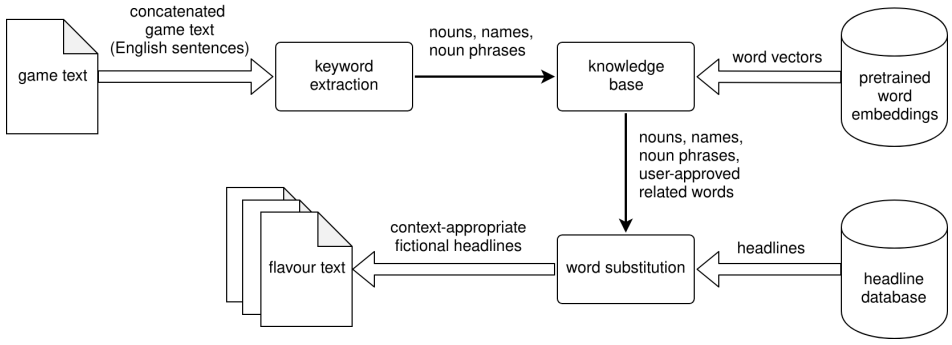

Figure 3: Churnalist: system architecture

"You are the system administrator of SuperSecure ltd, a hosting company. At four o'clock in the afternoon, your manager storms in. Apparently, there has been a break-in in your computer network. The CEO has been receiving anonymous emails from a hacker that demands a payment of $100,000 before midnight. If SuperSecure does not pay, they threaten to publish sensitive company documents online. The manager is worried, since the hacker claims to possess important intellectual property. Manager: Can you find out how the hackers got into our systems? CSIRT: We recognize this mode of operation. We will share some relevant IOCs with your company. Can you contact us if you have finished your forensical analysis? Security officer: There has been a nation-wide increase in phishing attacks in the past few days. System administrator: I can't find any traces of active malware on our Windows server. I will check the network log files for malicious activity."

Figure 4: Representative input text for Churnalist: game text from a dilemma-based serious game, consisting of a description and a few lines of NPC text. Names and noun phrases are underlined.

Keyword extractor

The start of the pipeline is the keyword extractor. We assume that the input text consists of grammatical sentences, so that it can be parsed by a sentence tokenizer and a dependency parser. The keyword extractor runs the input text through the NLTK sentence tokenizer[1] and the spaCy[2] part-of-speech-tagger and dependency parser trained on spaCy's default corpus for English.[3] It uses spaCy's noun phrase extraction to extract all English noun phrases from the input text. The keyword extractor saves all noun phrases that occur in the input text, together with the head of each noun phrase. Churnalist uses the head nouns as seed words and saves the noun phrase itself so it can be reused later, during the word substitution phase. See Figure 5 for an example of seed words and the corresponding noun phrases as extracted from the input text in Figure 4.

Knowledge base

In order to get more variety in our output, and not limit the words used in our output to nouns extracted from the input text, we extend the list of seed words with related words. Note that we mean 'related' in a broad sense; not just synonyms. To obtain these words, Churnalist queries the knowledge base for words similar to the seed words. We took this idea from the procedurally generated biographies in Caves of Qud (Grinblat and Bucklew 2017), which evoked a feeling of coherence because of the related domain words that were put into the biography. For example, for the seed word 'ice' the words 'lightblue', 'frost', 'cold' or 'winter' would all be suitable related words. The word lists for the various domains in Caves of Qud were written manually by the game developers. However, we do not want to build large content models by hand. Instead, we want to focus on generating text from data that can be obtained automatically. Like Repar et al. (2018), we used the English dataset of FastText's pre-trained word embeddings (Bojanowski et al. 2017) as a knowledge base, similar to the external semantic datasets used by the PoeTryMe framework (Gonçalo Oliveira 2012).

Word embeddings are a method for encoding words as their context, based on a corpus. The FastText dataset is based on a corpus of Wikipedia articles. It contains words represented as vectors that encode the context of these words. Words (vectors) that are close to each other in the resulting vector space, are words that occur in similar contexts.

A useful property of the FastText dataset is that it contains word embeddings that encode subword information: the vector of a word is created from the vectors of its subwords of length n. As a consequence, the dataset can be used to obtain vectors for out-of-vocabulary words: we only need to create a vector for them by looking at the vectors of their subwords. This allows us to deal with words that are not present in the semantic resources being used. Consequently, we bypass a problem similar to O Poeta Artificial's out-of-vocabulary hashtags (Gonçalo Oliveira 2017). Another advantage of using FastText is that its datasets are available in multiple languages, which allows us to port our system to languages other than English (for instance, Dutch).

The seed words are passed on to the knowledge base, which tries to assign a vector to each word and find its closest neighbours. If the seed word is an out-of-vocabulary word,

[1]NLTK 3.3, `https://www.nltk.org`

[2]spaCy 2.0.16, `https://www.spacy.io`

[3]Language model en_core_web_sm 2.0.0

Head noun	noun phrase
administrator	system administrator
company	hosting company, company
network	computer network
CEO	CEO
emails	anonymous emails
documents	sensitive company documents
manager	manager
property	important intellectual property
hackers	hackers
IOCs	relevant IOCs
analysis	forensical analysis
officer	Security officer
traces	traces
malware	active malware
server	Windows server
files	network log files
activity	malicious activity

Figure 5: Noun phrases and their head noun that the keyword extractor extracted from the input text from Figure 4. Generic phrases, such as 'days', 'o'clock' and 'afternoon', were removed manually from the list of seed words.

Word	distance	remark
companiess	0.7817	typographical error
subsidiary	0.6847	
telecompany	0.6821	too specific
companywide	0.6773	not a noun
ecompany	0.6668	too specific
webcompany	0.6496	
corporation	0.6315	
firm	0.6218	

Figure 6: Examples of suggestions from the knowledge base for the word 'company', together with the distance between the word vector for 'company' and the word vector for the suggestion. The knowledge base lists results in descending order of distance to the seed word. The final column lists reasons for rejecting this word. Not all suggestions by the knowledge base are shown.

the system calculates a new vector for the word based on the word embeddings of its subwords, and uses this new vector to find related words. The user can set a minimum distance for suggestions from the knowledge base and select which suggested words should be passed on to the substitution step. For an example of the results of the knowledge base, see Figure 6.

The knowledge base contains a machine-learned word embeddings model that was trained on Wikipedia dumps. Consequently, there are words in the model that are unsuitable for inclusion in Churnalist's output, such as words with crowdsourced typographical errors. For example, the words closest to 'company' are 'companiess', 'companythe' and 'companyx', which result from typographical errors (and possibly pre-processing errors) in the Wikipedia dataset. Additionally, some words are very similar to one of the seed words but have no connection to the way that seed word is used in the input text. Take the compound noun 'security officer', which means someone who defines and enforces the information security policy in a company. Its head noun is 'officer', for which the KB will list 'sergeant', 'quartermaster' and 'sublieutenant' as related words. However, these words have little connection with the term 'security officer' and should not be used in the output. The user of Churnalist can filter such unsuitable suggestions for related words from the knowledge base.

The set of seed words together with the set of related words from the knowledge base forms the set of *context words*. Figure 7 shows the final set of context words for the seed words from Figure 5.

Substitution module

The substitution module receives the list of context words from the knowledge base module and produces new headlines that contain one or more context words. It creates new headlines by substituting the subject of an existing headline from the headline database. This approach is similar to that of Headylines (Gatti et al. 2016), which inserts keywords from a newspaper article into existing sentences.

As external dataset of headlines, we use a collection of headlines scraped with the API from News API.[4] This API returns headlines and article excerpts from several large news websites. We collected 3629 headlines from media from the UK and the US in December 2018 and January 2019.

The substitution module starts by picking a random headline from the headline database. This headline is used as the starting point for one new headline. The headline is run through spaCy's part-of-speech-tagger and dependency parser, trained on spaCy's default English corpus. From the information of the parser, Churnalist tries to find the subject of the sentence. This is the substitution target. If the parser cannot determine what the subject of the sentence is, a different headline is drawn randomly from the headline database.

Next, Churnalist chooses a random seed word. Each seed word has a set of context words associated with it: the seed word itself, noun phrases from the input, and the user-approved related words from the knowledge base. Churnalist randomly chooses one of these as a substitution candidate. If the substitution target is of a different number than the substitution candidate, Churnalist converts the candidate to the right number (singular to plural or vice versa). Finally, the target is substituted by the candidate and the new headline is presented to the user.

Results

In this section, we discuss our results. Figure 8 shows examples of generated headlines, together with the original headline and seed word. Consider the requirements we mentioned earlier: generated headlines should have an appropriate form, should be grammatical and should be context-appropriate for the input text.

Firstly, applying text modification to the headlines will lead to texts that again look like headlines. Informal inspection of the headlines generated suggests that this requirement is

[4]News API, `https://newsapi.org`

Seed word	approved suggestions	rejected suggestions
administrator	-	administratorship, admininistrator, nonadministrator
company	subsidiary, webcompany	companiess, companythe, companynew
CEO	executive, shareholder, entrepreneur, investor	CFO, COO, CTO
network	-	networky, networkx, networknbc
emails	-	emailings, voicemails, emailers
documents	documentation, memos, archives	documentations, documen, documentries
manager	teammanager	managership, imanager, managerin
property	-	poperty, propert, propertyless
hackers	hacktivists, cybercriminals, scammers	hackings, blizzhackers, hackery
IOCs	-	ligtvoet, zeijst, lennaert
analysis	-	analyses, analysises, analysist
officer	-	underofficer, officerer, commander
malware	spamware, botnet, vulnerabilities	spyware, malwarebytes, antivirus
server	-	iserver, vserver, pvserver
files	folders, fileserver	fileset, filesmy, filespace
activity	-	activitiy, activitism, activin, reactivities

Figure 7: Seed words and examples of knowledge base suggestions for related words. Not all words suggested by the knowledge base are shown. The results were approved and rejected manually by the first author. Words in the 'approved' column are added to the set of context words.

Seedword	system administrator
Headline	Revealed: 500k number plate conman is a convicted people smuggler
Output	Revealed: system administrator is a convicted people smuggler
Seedword	hosting company
Headline	Pelosi has edge over Trump on budget negotiations, CBS News poll shows
Output	Hosting company has edge over Trump on budget negotiations, CBS News poll shows
Seedword	computer network
Headline	Met Office issues ice warning as snow hits UK
Output	Computer network issues ice warning as snow hits UK
Seedword	hacker
Headline	Uber loses latest legal bid over driver rights
Output	Hacker loses latest legal bid over driver rights
Seedword	sensitive company documents
Headline	Investigators revise cause of escape room fire that killed 5 girls
Output	Sensitive company documents revise cause of escape room fire that killed 5 girls
Seedword	forensical analysis
Headline	MPs' threat to block government's tax without second brexit referendum
Output	MPs' threat to block forensical analysis without second brexit referendum

Figure 8: Generated headlines for the input text in Figure 4.

fulfilled sufficiently. The headlines are often grammatical, but not always. Sometimes, the dependency parser has trouble selecting the full noun phrase in both the input text and in the headlines from the headline database, which leads to only partially substituted objects and subjects. Since Churnalist is meant for supporting game writers, we rely on the user to filter and discard ungrammatical output.

In the current version of the system, where seed words and headlines are selected and matched at random, many of the generated headlines would probably not yet be considered context-appropriate. For example, readers will not necessarily relate headlines mentioning 'company documents' to the stolen company documents from the game text. We have not yet formally evaluated the output of our system for the 'context-appropriate' property. We plan on making further improvements to the system and evaluating both the system and the outputs. It could be that readers behave according to Veale's 'charity of interpretation' and are more generous in their interpretation than we anticipate.

Some seed words have a stronger connection to the game story and will evoke a stronger sense of coherence than others. For example, we expect that headlines that mention 'hackers' will be easier for the readers to connect to the story than headlines that mention 'managers'. Most companies have managers; few companies have problems with malicious attacks from hackers. We expect that incorporating a stricter filter for seed words will lead to headlines with a stronger link to the game story from the input text. For example, we could rank seed words based on their term frequency-inverse document frequency (tf-idf). This would take into account that seed words that occur frequently in general are probably less representative for the input text than seed words that occur rarely in other English texts. For now, we leave the task of filtering the seed words and generated headlines to the human user of Churnalist.

Finally, choosing a random headline from the database for

substitution is a mixed blessing. On the one hand, combining a context word with the randomized headline can lead to surprising, unexpected and creative outputs. On the other hand, sometimes the link with the context word that was chosen for substitution is far-fetched or even downright ridiculous.

The application domain of Churnalist is supporting game writers in their creative task. Since Churnalist requires no linguistic knowledge, it is an accessible tool. Instead of relying on hand-written linguistic models, it requires external datasets for its text modification functionality, By using News API for collecting headlines and using the FastText dataset as knowledge base, Churnalist can run fully on publicly available data. Similarly to PoeTryMe, users can choose to use different datasets for their particular application, for example for a different language than English.

However, using external data for NLG has some caveats. Reusing headlines has as advantage that we do not have to write templates. The disadvantage is that quality of the output headlines will never be better than the quality of the headlines from the headline database. Using headlines from low-quality news outlets with click-bait headlines, the output headlines will show similar clickbait properties.

Conclusion

We have presented Churnalist, a system for generating fictional headlines. The content of the headlines is determined by the noun phrases present in the input text. Churnalist creates new headlines by taking keywords from the input as seed words. It expands the list of seed words by querying a knowledge base of word embeddings for related words and injecting these into existing headlines via word substitution. We circumvented problems with out-of-vocabulary seed words by using word vectors based on subword information. The user can fine-tune the quality of Churnalist's output by filtering the intermediary output of each step in the system pipeline.

Churnalist can be used by game writers, as an authoring aid for writing flavor text in the form of headlines. We have provided example outputs for every step in the system pipeline, generated from game text from a dilemma-based game. Since our system was developed using publicly available datasets, open source libraries and only simple text modification techniques, Churnalist requires no linguistic expertise from its users. Although Churnalist is currently implemented for English, the use of external datasets allows us to adapt the system to other languages and use cases with minimal effort. This makes Churnalist suitable for different languages and game types.

Future work

There are three main directions for future work on Churnalist. Firstly, we can improve the implementation by using better and more appropriate tools and resources. We used an open source dependency parser for Churnalist that was trained on a standard corpus of English. Training the parser on a set of headlines could improve its accuracy, which might lead to better quality text transformations. Similarly, we expect that the quality of text transformation will improve if the headline database consists of better quality data, such as the annotated GigaWord corpus (Napoles, Gormley, and Van Durme 2012).

Secondly, there are several possibilities for expanding Churnalist's approach to generation. Like related systems (Gonçalo Oliveira 2012; Özbal, Pighin, and Strapparava 2013; Repar et al. 2018), Churnalist could use a generate-and-test strategy, where multiple candidate headlines are generated and a fitness function determines the best candidate headline or headlines from this set as output. Instead of applying word substitution to randomly chosen headlines, Churnalist could select headlines that show semantic similarity with the input text and use these as a basis for transformation. More advanced methods for keyword extraction from the input texts could be used, going beyond simple noun extraction. Using additional semantic resources, such as Wordnet or ConceptNet, could also help Churnalist in suggesting more valid words to the user.

Thirdly, we would like to expand Churnalist's outputs to also include social media messages, to make it possible to automatically generate social media messages as flavor text. To generate social media messages, we could take a similar text transformation approach as we have used for the headlines. As social media is often used to share news headlines, we can incorporate headlines as a specific type of social media messages. In fact, some malicious Twitter bots use text modification techniques and news headline sharing to disguise the fact that they are bots (Hegelich and Janetzko 2016). Additionally, both headlines and social media messages are interesting vehicles for exploring affective language generation. For example, we could generate headlines with a particular political slant or social media messages that express a particular emotion.

Finally, we still need to evaluate our approach to text generation for games. We plan to do so in various ways. We want to ask human judges to assess the output of Churnalist on properties such as grammaticality and 'context-appropriateness' (see our headline requirements), and draw comparisons with a baseline. Given the current popularity and quality of neural generation systems, we would also like to compare the output of Churnalist to state-of-the-art neural headline generation systems, given the same game text as input.

Acknowledgments

This research is supported by the Netherlands Organisation for Scientific Research (NWO) via the DATA2GAME project (project number 055.16.114). We would like to thank Dr. Lorenzo Gatti and all reviewers for their useful remarks.

References

Backus, K. 2017. Managing output: boredom versus chaos. In Short, T. X., and Adams, T., eds., *Procedural Generation in Game Design*. AK Peters/CRC Press. chapter 2, 13–21.

Banko, M.; Mittal, V. O.; and Witbrock, M. J. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, 318–325.

Bay, B.; Bodily, P.; and Ventura, D. 2017. Text transformation via constraints and word embedding. In *ICCC*, 49–56.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics* 5:135–146.

Charnley, J. W.; Pease, A.; and Colton, S. 2012. On the notion of framing in computational creativity. In *ICCC*, 77–81.

Colmenares, C. A.; Litvak, M.; Mantrach, A.; and Silvestri, F. 2015. Heads: Headline generation as sequence prediction using an abstract feature-rich space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 133–142.

Colossal Order. 2017. *Cities: Skylines*. Game [PC]. Paradox Interactive, Stockholm, Sweden.

Dorr, B.; Zajic, D.; and Schwartz, R. 2003. Hedge Trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, 1–8.

Eidos Montral. 2011. *Deus Ex: Human Revolution*. Game [PC]. Square Enix, Shinjuku, Tokyo, Japan.

Freehold Games. 2018. *Caves of Qud*. Game [PC/Mac/Linux]. Freehold Games, USA.

Gatti, L.; Özbal, G.; Guerini, M.; Stock, O.; and Strapparava, C. 2015. Slogans are not forever: Adapting linguistic expressions to the news. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2452–2458.

Gatti, L.; Özbal, G.; Guerini, M.; Stock, O.; and Strapparava, C. 2016. Heady-lines: A creative generator of newspaper headlines. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces, IUI 2016*, 79–83.

Gonçalo Oliveira, H. 2012. PoeTryMe: a versatile platform for poetry generation. In *Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence (C3GI at ECAI 2012)*.

Gonçalo Oliveira, H. 2017. O Poeta Artificial 2.0: Increasing meaningfulness in a poetry generation twitter bot. In *Proceedings of the Workshop on Computational Creativity in Natural Language Generation (CC-NLG 2017)*, 11–20.

Grinblat, J., and Bucklew, C. B. 2017. Subverting historical cause & effect: generation of mythic biographies in Caves of Qud. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 1–7.

Hegelich, S., and Janetzko, D. 2016. Are social bots on Twitter political actors? Empirical evidence from a Ukrainian social botnet. In *Tenth International AAAI Conference on Web and Social Media*.

Hofstadter, D. 1995. Preface 4: The ineradicable Eliza effect and its dangers. In *Fluid concepts and creative analogies: computer models of the fundamental mechanisms of thought*. Basic Books, New York.

Jing, H. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, 310–315. Seattle, Washington, USA: Association for Computational Linguistics.

Lukin, S. M.; Ryan, J. O.; and Walker, M. A. 2014. Automating direct speech variations in stories and games. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Maxis. 1996. *SimCity 2000*. Game [PC]. Maxis Software Inc./Electronic Arts.

Napoles, C.; Gormley, M.; and Van Durme, B. 2012. Annotated GigaWord. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, 95–100. Association for Computational Linguistics.

Özbal, G.; Pighin, D.; and Strapparava, C. 2013. BRAINSUP: Brainstorming support for creative sentence generation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 1446–1455.

Repar, A.; Martinc, M.; Žnidaršic, M.; and Pollak, S. 2018. BISLON: BIsociative SLOgaN generation based on stylistic literary devices. In *ICCC*, 248–255.

Schlünder, B., and Klabunde, R. 2013. Greetings generation in video role playing games. In *Proceedings of the 14th European Workshop on Natural Language Generation*, 167–171.

Shen, S.-Q.; Lin, Y.-K.; Tu, C.-C.; Zhao, Y.; Liu, Z.-Y.; Sun, M.-S.; et al. 2017. Recent advances on neural headline generation. *Journal of Computer Science and Technology* 32(4):768–784.

Strong, C. R.; Mehta, M.; Mishra, K.; Jones, A.; and Ram, A. 2007. Emotionally driven natural language generation for personality rich characters in interactive games. In *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 98–100.

Thiennot, J. O. 2013. *Cookie Clicker*. Game [PC/Browser]. `http://orteil.dashnet.org/cookieclicker/`. Played September 2018.

Veale, T. 2016. The shape of tweets to come: Automating language play in social networks. *Multiple Perspectives on Language Play* 1:73–92.

Yannakakis, G. N., and Togelius, J. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 2(3):147–161.