Aspect Graphs

for conversation summarization

Nguyen Duc Duy

Supervisors: Prof. Giuseppe Riccardi
Dr. Evgeny A. Stepanov

Centre for Mind/Brain sciences
University of Trento

This dissertation is submitted for the degree of
Master of Science

October 2018

I would like to dedicate this thesis to my loving parents who gave me the eyes to sight, the mind to fulfil and a brave heart to love. . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Nguyen Duc Duy
October 2018

Acknowledgements

For half a year working on this thesis, I have received thoughtful and enthusiastic guidances from my supervisors, Prof. Riccardi Giuseppe and Dr. Evgeny A. Stepanov. Therefore, I would like to acknowledge them as well as researchers and students at Signals and Interactive Systems Laboratory, University of Trento for their precious helps. Additionally, this work would have not been done without the computing power supported by the University of Information Technology, Vietnam National University.

Abstract

Significant advances of the Internet have brought many human means of communication online, especially verbal conversations such as chat, messaging, discussion, etc. These online conversations contain information about every aspect of our life in abundance. Thus, automatically summarizing this large amount of data to identify most important information is useful for both industry and academic purposes, such as business intelligence and social research. Numerous methodologies for text summarization have been developed to date, however, far little attention has been paid on summarizing online conversation. The centre of a typical summarization process is the construction of an intermediation summarical representation of the text, from which summary can be generated using either abstractive or extractive approach. Special linguistic and structural features of online conversation, unfortunately, challenging current approaches for this representation thus refraining the production of a high-quality summary.

In order to tackle this problem, we propose Aspect Graph, an abstractive intermediate representation for online conversation specified for summarization purpose. Aspect graph represents conversation as a directed graph with community structure, each community is corresponding to one aspect of the conversation. Furthermore, we introduce an unsupervised methodology for constructing this structure based on dependency, frequency, sentiment and centrality information. Also, various variants of Aspect Graph are presented and compared to provide a better insight of the method.

We benchmark our approach against many state-of-the-art methods in two tasks that commonly used for deriving intermediate representation in automatic text summarization system: keyphrase extraction and topic modeling. These experiments are carried out on conversations annotated invoked by The Guardian online news article. Results in keyphrase extraction reveal that our method has strong advantages in phrase completeness while avoiding redundant and infrequent error, henceforth it works better state-of-the-art. In topic modeling experiment, we evaluate our method in comment topic clustering and classification tasks. The results obtained from clustering task show that aspect graph enhances with high consistency within clusters, completeness, and low error rate. Unfortunately, homogeneity of the clusters generated by our method is slightly lower than other candidates, therefore there is no clear

evidence of prominence. Classification task results offer confirms the effectiveness of aspect graph by outperforming state-of-the-art in term of accuracy, precision and recall.

All in all, our experimental results highlight the advantages of AG in representing summarical information of conversation. Thereby, it is reasonable to substitute current forms of intermediate representation with aspect graph in conversation summarization system. Moreover, this research provides a blueprint from which many variations of aspect graph can be developed to solve the increasing need for conversation summarization systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The landscape

The development of big data in the last two decades has redefined the way we think about the world[20] with game-changing opportunities for research and technology innovation. Despite its undeniable potentials, as said by Sarah Williams, an assistant professor of urban planning at Massachusetts Institute of Technology, "big data will not change the world unless it's collected and synthesized into tools that have a public benefit.". Certainly, the five key characteristics of big data, these are volume (high quantity of data generated and stored), variety (various types and nature of the data), velocity (demand for high speed on storing and processing data), variability (inconsistency of the data set) and veracity (great vary in data quality) [21, 86] brought many grand challenges in data complexity, computational complexity, system complexity[37], especially in data mining. Henceforth, it is increasingly becoming difficulty to manually mange and process the data in order to seek for valuable knowledge and information. This raised the need for automatic information extraction and summarization systems to narrow down the scope of information to be processed and point out important information. In fact, there are several reasons emerging the need to summarising text, to mention but a few:

- Summaries save reading and processing time, thereby incapacitate a larger volume data to be processed.
- Summarization improves the effectiveness of information searching and indexing.
- Compared to summaries produced by human, automatic generated summaries is less biased and more neutral. In reality, there are two phases of manual summary production:

understanding the source text and writing its concise a shortened version. Both these two phases depend not only in human linguistic and extra-linguistic skills but also his/her knowledge about the topic and understanding about summarization task.

- Query-focused summaries are useful for question-answering system to tackle general questions where information is distributed in many locations in a document or across documents.

Although the term information summarization is conventionally used for summarizing natural language data, big data winded its potential targets to various types of data, such as visual content (e.g. image, video, etc.) , geographic data, real-time data, time-series data, etc. However, summarizing textual data has never been an outdated field of study. So far, however, there has been little discussion on summarizing online conversations. A conversation is an exchanges of thoughts, feelings, news, ideas, information and between two or more people with questions and answers[85]. Typically, a conversation is triggered by a common concern or a topic under various forms, such as news, an oral or even an activity. The past decades have seen the rapid development of the internet and web-based service technology which generated platforms for conversations was carried on in cyber space and so called *online discussion*. Some of them are public, the others are private and together they cover every imaginable topic [22] from science to music, from climate to terrorism. Today, they can be seen at every corner of the word-wide-web, such as email clients, bulletin boards, newsgroups, internet forums, etc. Comments trigged by a post on Facebook or Twitter, comments on a video on Youtube or an article in online news can also exemplified the wide-spreading of online conversations. Early from 2012, a statistical research study [fb2] reported that every 60 seconds, there are 510,000 new posted comments and 293,000 status updated on Facebook. Due to the variety of topics, accompany with the freedom-of-speech and feeling of anonymous brought by the online environment, online conversations are becoming longer and more chaotic. A recent survey on Twitter revealed that there are 350,000 tweets sent every minute in 2016[3] and this number become 456,000 tweets per minute by 2017[2]. Consequently, seeking for information in these conversations, like reading tea leaves, is a challenging, time-consuming and expensive task. Figure 1.1 shows a typical online conversation following news topic *Researchers tackle link between climate change and public health*[1]. While manual or semi-automatic methods are overheated under this circumstance, there raised a need for automatic online conversation summarization system.

Automatic conversation summarization can be used for various applications in both academic and industry. In scientific research studies, it allows us to capture new thoughts and ideas

[1] Researchers tackled link between climate change and public health, The Guardian, 2014

that may spark out during discussion - a very essential academic activity. For example, one may want to know new ideas following innovations inspired by a talk on TED [2], or understanding the voice of research community behind the comments about their research published on ResearchGate[3]. Besides, industrial applications of conversation summarization cover a wide spectrum in various fields, such as business intelligence and dialogue technology. One of the most beneficial uses of the system is social listening which is a critical component of marketing. Social listening aims to understand the customers' opinions about a brand or industry by monitoring digital conversations online. This process allows marketer to test one campaign against another, monitor return on investment, or prove the value of social marketing resulting effective strategy planning and decision making. Similarly, the system can also be a powerful tool for market research which aims to systematically gather and interpret of information about consumers, competitors and the effectiveness of marketing programs via different channels, including social media. To the case of dialogue technology, automatic conversation summarization can be integrated with voice recognition systems generate automatic summarization of oral conversations, such as meeting, phone call or debates.

Challenges of processing online conversations

Despite the general problem (i.e. text summarization) has been investigated for more than six decades, there are not many researches specified to conversation due to its intrinsic characteristics:

- **Noisiness**: the majority of online conversations use expressions that are closer to spoken language than writing language, which means more noises can be spotted. There are several different kinds of noise, such as typo, spelling and grammatical mistake, etc. Moreover, conversation can contains a variety of external references, such as quotation, hashtag[4], mention[5] and hyperlink.

- **Vocabulary**: it is also reported that many words change their meaning when coming to the case of online conversation. A typical example of this phenomenon is the case of the word *troll* (verb) whose ordinary meaning is to pull a baited line through the water (behind a boat) in order to catch fish [85], today also means intentionally do provokes others into an annoying and uncomfortable emotional situation. Similar transformation of meaning can also be seen in many other words, such as *stumble*, *stream*, *bandwidth*,

[2] TED - Technology, Entertainment and Design

[3] ResearchGate - Advance your research

[4] A type of metadata tag used in news and social media for easy finding, categorizing content

[5] A type of metadata tags to refer to another person in the community. The referred person may not necessarily be in the conversation.

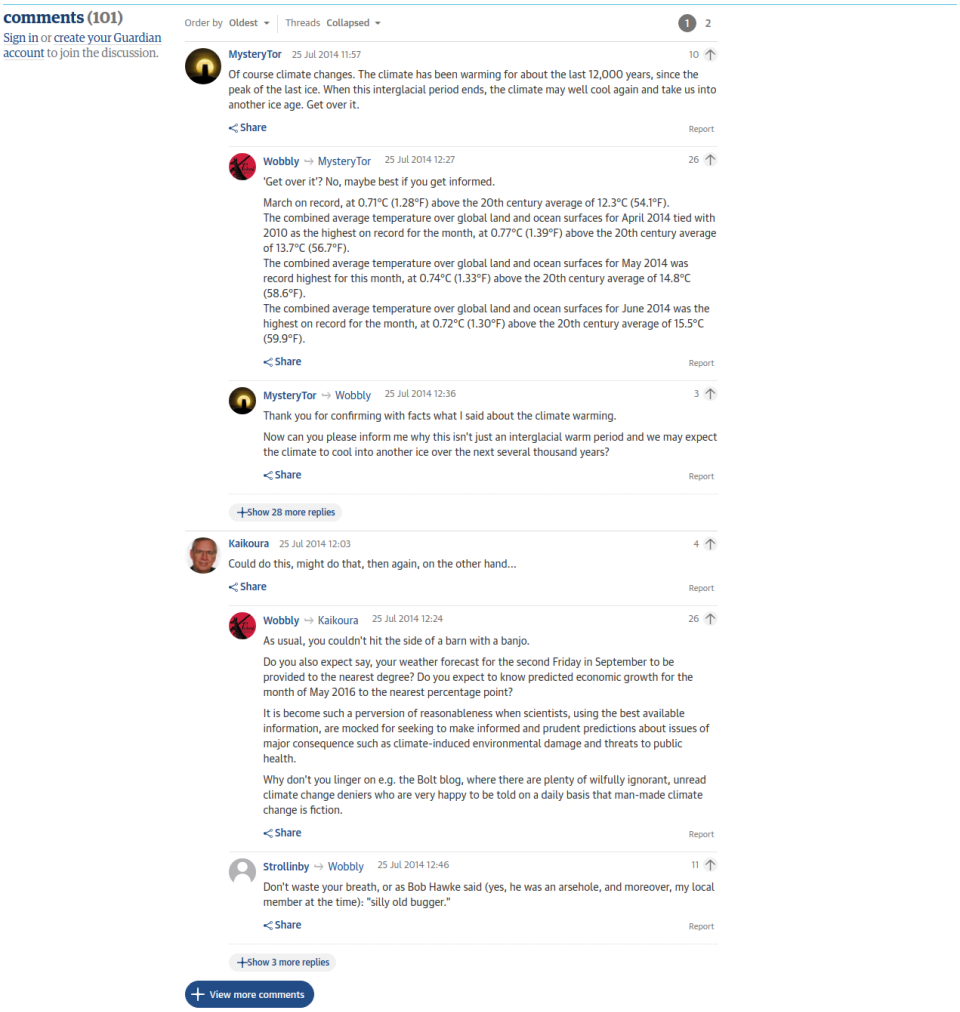


Figure 1.1 Example of conversation trigged by news article

etc. In addition, brand-new words, for instance google (verb), tweet (verb and noun) are enriching the lexicon of online conversations.

- **Shortness**: there are some theoretical and technical evidences leaded to character limitations enforced by many online services[11]. Such restriction make comments short and compact and sometime unexpectedly lead to malformed expression from grammatical perspective. Of course, short expression provides less information forthwith challenges many statistic-based methodologies.

- **Sentiment**: besides exchanging information, participants of an online conversation also express their opinion and feeling as the story go on. Conventionally, adverbs and adjective are responsible for this task, but in case of online conversation, there are more ways to

express one's sentiment. One of the most common practices is to use interjection, such as *uh, uhm* (used to indicate hesitation, doubt, or a pause), *uh-oh* (used to indicate dismay or concern), *ah* (used to express delight, relief, regret, or contempt), *boo* (used to scare someone or to voice disapproval) etc. Abbreviation (e.g. *lol* - laugh out loud), idioms and special expression (e.g. fake the snake - to add in a small specific detail in order to make a lie seem more convincing) are other way to show off the emotion. Another way expressing sentiment in conversation is emoticon (i.e. emoji, smiley) such as ☺, ☹, ☼, or just simple combination of characters like *:D* or *:(*. A research revealed that in 2016, users of Twitter sent 9,678 emoji-filled tweet every 60 seconds [3]. Furthermore, Facebook and Twitter recently introduce stickers, GIF (Graphics Interchange Format) and image comment.

- **Thread structure**: is a widely used feature in online conversation. Figure 1.1 illustrates a thread-based conversation in which a comment about the main topic triggered a sub-conversation. To distinguish comments in this sub-conversation to comments that directly referred to the main topic, they are call replies. The depth of this thread structure is varied among web-services, although the majority of them only support the depth of one (i.e. one sub-conversation for each comments accounted for the main topic). This branching structure is problematic because a reply now can easily go off-topic by discussing about the comment, not the main topic itself. In another word, as conversation are freely developed by participants, topics in the conversation drift further from the primary topic, therefore threads are no necessarily corresponding to topics.

The sophisticated nature of online conversation, of course, refrains any one-size-fits-all methods. Thus, we focus on one specific type of online conversation: online comment to news article. Burning topics on online news are attracting a tremendous number of readers every day. They leave comments as instant responses to an article; for example, many readers also ask questions, point out mistakes or even five new leads. A report from The Guardian [6] revealed that with one decade since 2006, there are 70 million comments left by readers in their online article. Interestingly, up to that date, these online conversations are mostly mediated by mediators and for ten years, also from the report, they manually removed 1.4 million comments (2% of the total). This fact exemplified the need for an online conversation summarization system for conversation monitoring.

[6] https://www.theguardian.com/technology/2016/apr/12/how-we-analysed-70m-comments-guardian-website

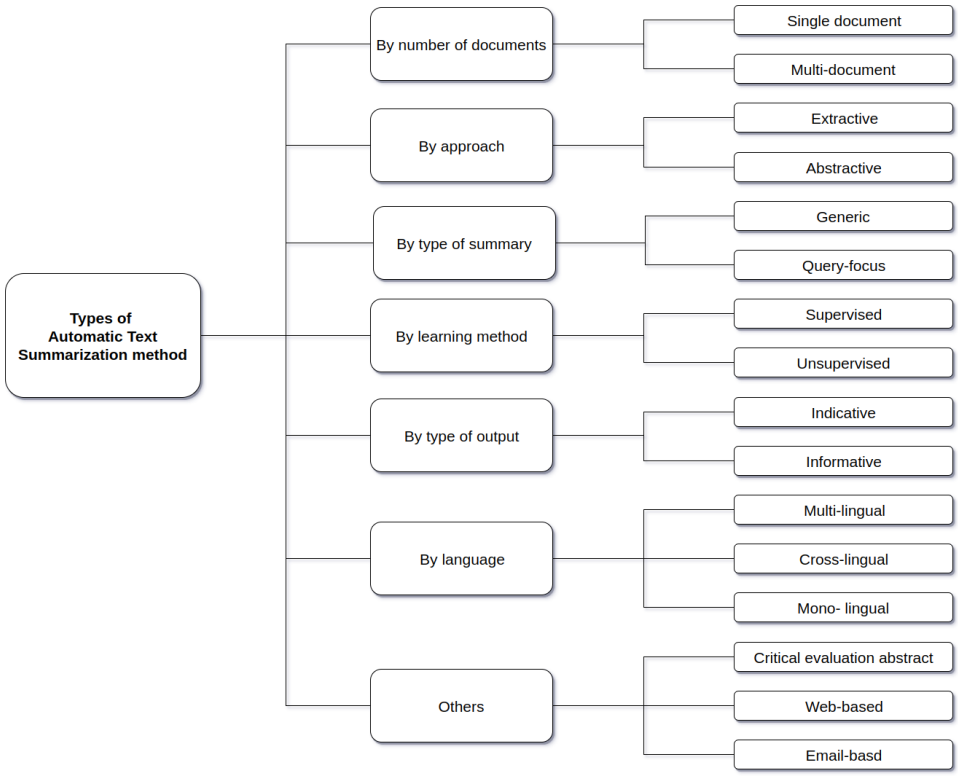


Figure 1.2 Categories of ATS

1.2 Automatic text summarization

Automatic text summarization(ATS) aims to generate a fluent and concise summary while containing the most important information from one or more texts. This definition inspired by the definition of *summary*, which according to Radev et al. [66], is a text produced from one or more texts that conveys important information and is significantly shorter than the original text(s). Since the task's advent in the 1950s, a considerable amount of literature has been published on automatic text summarization.

Categories of ATS

ATS system can be categorized in various ways, depending on the basis of number of documents, approach, types of summary, learning method, types of output, language, etc., as shown in figure 1.2

- By number of documents: while single document summarization tends to generate summary from a single document, multi-document summarization works on multiple document[24]. Despite the common held belief that the latter is just an extension of the

former one, in practice multi-document more difficult due to the volume of data, variety of the information and most importantly, the redundancy. There has been many methods proposed for tackling this issue [78, 89, 88], but yet, just a few of them is applicable to the case of conversation. For instance, in [70], Sakar et al. employed the topic sentence for each paragraph as anchor and disregards next sentences with low similarity to the anchor. However, the standard format beginning with topic sentence rarely exists in common conversation expression.

- By approach: extractive and abstractive summarization. Extractive text summarization generate a summary by selecting some important sentences from the original documents bases on some weights (i.e. saliency scores) assigned to each sentence. Researchers have suggested many different ways to compute these scores, mostly based on keywords, keyphrases or frequencies[14, 4]. So far, this approach has dominated ATS due to its robustness, simplicity and feasibility[27]. Abstractive summarization, on another hand, is much more complicated. It not only captures the keywords and keyphrases, but also re-interprets and reproduces the information in a particular form. Thus, an abstractive summarization does not necessarily contain the exact keywords and keyphrases from the original documents. Therefore, it requires many intensive natural language processing techniques and results in a flexible and comprehensive. Additionally, re-interpretation allows the participation of prior knowledge and reasoning that help the summary become more domain-specific and context-aware.

- By type of the summary, there are generic summarization providing general information about the original document, whereas query-focused summarization (i.e. topic-focused or user-focused) querying some specific aspects or contents from the document[60, 87].

- By training method: supervised and unsupervised methods. Supervised learning methods are commonly used for extractive approaches. The goal is to learn an efficient model that effectively assigns salience scores indicating the importance of sentences from a large amount of annotated data. At its simplest form, this is a two-class classification problem (a sentence is marked as *positive* as it is likely to appear in the summary, and *negative* otherwise [73, 60]) which can be done using various machine learning methods, such as Support Vector Machine [60], neural networks and Gaussian mixture model [24]. On the contrary, unsupervised methods directly analysis the original documents to seek for designated features and information for generating the summary. Two common approaches for this method are rule-based [24] and topic clustering[53]. Similar to the case of abstractive summarization, unsupervised summarization is relatively more complex than the other one. However, since this method doesn't require any training

data, it can be applied to any newly observed data, thus, suits the case of conversations and big data [53].

- By style of output: an indicative summary is description (metadata) of the document such as general topic or the scope. In contrast, an informative summary contains not only topics but also informative part of the original documents.

- By language: mono-language (both the documents and summary are in the same language), multi-language (the documents are in different language and the summary is in one of these language) and cross-language (documents are in one language and the summary is in another language).

- Other systems can serve at the back-end of information retrieval system (such as search engine like Google[7], Bing[8], etc.) for generating a summary from cluster of retrieved web pages that is known as Web-based summarization. Another type of ATS targets email conversations called email-based summarization. In addition, there is one type of summarization called *critical evaluation abstract* which is specific for reviewing academic papers. Indeed, before publishing a scientific work, authors will receive feedbacks from a group of reviewers which may consist of various information, such as opinion, recommendations and decision (either acceptance or reject). In many cases, the group of reviewers may consist of many members in different backgrounds, which raises the need of an automatic method for summarizing these feedbacks.

It is immersed from these categories that some methods usually comes together. For example, researchers often solve extractive summarization with unsupervised method to archive indicative summary. Abstractive summarization, on the other hand, is more likely to be solved using unsupervised methods with the ability to output both indicative and informative summary. In our research, we follow abstractive summarization approach which will be explained in the following section.

Abstractive text summarization

Back to the 1950s when automatic text summarization first discussed, its goal is to produce the most concise and credible summaries, which was originally called *the abstract*, in the same way human does. To the date, we are still far from this objective and the fact the research community is too much focusing on extractive approach exemplified the difficulties of abstractive text

[7] https://www.google.com
[8] https://www.bing.com/

summarization. Frankly speaking, extractive approach simplified the summarization process by disregard the problem of making a machine understand a text, which is still an unresolved problem. Consequently, unfortunately, HexTAC experiment at TAC'09 campaign revealed that extraction-based methods had already reached a ceiling in terms of their performance[80]. This urged the need for conducting more research into abstraction-based methods, starting to sentence summarization, short paragraphs summarization, single document summarization, multi-document and beyond. At the level of sentence and short paragraph, the question is similar to the problem automatic sentence compression that aims to condense the text to its most important content. This method is widely used for enhancing writing quality, satisfying document length constraints, and improving the accurate of document summarization systems[12].

Coming to the case of multi-document, the task is relatively more complicated, but resulting in more interesting application, such as article headline generation[77]. Recently, many techniques from neural machine translation have been applied to solve the problem at all levels, from sentence summarization to multi-document summarization. The intuition of this approach is to perceive abstractive text summarization as translating from the source language, where many sentences are need to express content, to another language where fewer sentences are needed. Furthermore, the acceleration of computing power and deep learning have successfully supported the neural approaches on abstractive text summarization. This approach aims to model the entire summarization process via one big artificial neural network. Typically this big neural network is separated into two smaller ones in an architecture known as encoder-decoder model, widely used in machine translation[9, 47]. These encoders and decoders can be built in different ways, and the most popular architectures are convolutional neural network, sequence-to-Sequence recurrent neural networks. There are several tricks proposed to improve the network's performance, such as neural attention model [Jobson and Gutiérrez, 19](i.e. focus the neural encoder to a a few words in sentence instead of all words) and black-out trick [36] (i.e. use discriminative loss to output layer in order to reduce computation while improving stability, sample efficiency, and rate of convergence).

A recent work of Ramakanth et al. [63] incorporated knowledge in natural language inference skills that was logically entailment, with neural document summarization and came up with promising initial improvements. One of the popular issues of sequence-to-sequence neural network models is the redundant repeating generations of words at the output of the decoder. It has been known in the last few years with the development of neural machine translation and also occurs in neural abstractive text summarization. A study of Jun et al. [76] managed to solve this problem by jointly estimate the upper-bound frequency of each target vocabulary in the encoder and control the output words based on the estimation in the

decode. The author claimed that significant improvement was made over a normal RNN-based encoder-decoder baseline. Despite there are debates about the idea of considering the whole summarization process as a back-box carried out by a big neural network, practically say, it has archived many remarkable results in the last few years.

Conversation summarization

The rapid development of neural text summarization with supervised methods requires a large amount of training data. So far, however, there has been too little annotated data for summarizing conversation, therefore most works on automatic conversation are still using supervised approaches, instead of deep learning. An early research by Benoit et al. [32] on generating summaries about product features and functions from customer reviews of products sold online. Although customer reviews do not have all essential characteristics of a conversation, this research has a straightforward approach that captures not only the information about the product but also the opinion of customers. The method consists of three phases: first the system finds out features of the product that customers have expressed their opinions on, the for each product's feature, it searches for reviews' sentences that give positive or negative emotion on; and finally, it generates a summary based on the discovered information. The determination of features and opinion words based on a dictionary (using Wordnet), word frequency and part of speech tag. Another work by Xiaodan et al. [91] reached further to spontaneous conversations where the source documents were auditory records, instead of textual content of the conversation. Therefore, beside natural language processing methods, they employed many speech recognition methodologies, such as automatic transcripts and prosodic feature extraction. The focus on utterance-level extraction uses various auditory and linguistic features to output both audio and text summaries. Research studies on educational dialogue were carried out in 2007 by Mahesh et al. [39] to produce summaries for instructors from student dialogues. They proposed a summarization process consists of 3 stages. In the first stage, they determine important characteristics of the conversations that should be displayed in the summary by extracting transactivity of discourse in order to identify segments of discourse which allow the system to predict effectiveness of the learning. In the second stage, they validated the design (i.e. the determined characteristics) on its ability to predict the learning effectiveness of the students. Finally, they reported this ongoing work to automatically identify transactive contributions in a discourse. The author claimed that they archived encouraging results despite the small dataset. Efforts on summarizing meetings and emails conversations were conducted a year later by Gabriel et al. [52] focused on conversation representation. Indeed, they introduced a set of 24 conversational features that is useful for both meeting

and email conversations. The experimental results showed that this features set archived a competitive result against domain-specific features.

Since 2010, the development of new media, especially social media attracted the attention of the research community. Thereupon, many studies were performed on new source of data, such as social media and chat and call-center conversations. In a preliminary work by Alan et al.[67], the author proposed the first unsupervised approach for modeling Twitter dialogue acts in an open domain. They constructed three models: conversation model, conversation with topic model and Bayesian conversation model for conversation model based on Hidden Markov Models. These models support unsupervised induction of dialogue structure from naturally occurring open-topic conversational data. Although this work focused on modeling rather than summarization, the learned model also carries important information of the conversation, therefore it can be extended toward an open-domain conversation summarization system. This research is among first studies that applied topic modeling to support summarization purpose which brings a certain degree of abstraction to the summary, rather than merely pointing out important segment of information. Henceforth, many attempts in text summarization with the support of topic modeling have been made with encouraging results [42, 53]. In [74] the authors worked on summarizing real-time chat conversations. The interesting points of this type of conversation are the involvement of multiple participants and the shifts in topic occurred frequently. Their focus their attention on this topic transition and called the method as topic-focused summarization. The whole process was divided into two phases. First, they employed topic modeling to seek for primary topics, then generate the summary generation phase accordingly. A recent work of Stepanov et al. [75] proposed an abstractive method for summarizing call-center conversation. They used a rule-based approach, beginning with the definition of hand-written templates. Then, they used topic-dependent rules to extract domain knowledge from the transcript and fill them to the templates. They further used these knowledges to fill that automatically learned templates (i.e. frequent patterns from hand-written synopses). In addition, they demonstrated the work on a browser-based graphical system.

Turning now to the specific case of news article comment summarization, there are only a few works had been done so far. Zongyang et al.[48] introduced two topic models, namely, Master-Slave Topic Model and Extended Master-Slave Topic Model. In these models, news article plays the role of master document, and each of its comments is a slave document. The first model limits the summarization to topics that derived from the commenting news article, while the other support new topics arise from the comments themselves. Their experimental results show that Extended Master-Slave Topic Model significantly outperforms its limited version. Another common way to approach conversion summarization problem is via topic clustering. Despite summarization encouraging results, [48] shows no significant evidence for

its performance of topic clustering over Latent Dirichlet Allocation (LDA), and in fact, the authors claimed that LDA is the best performing system. Another interesting work by Llewellyn et al.[46] examined multiple approaches in clustering comment, including Unigrams, K-means, Cosine Distance, Topic Models (with LDA). The results again confirm the superiority of LDA. In 2016, Aker et al. [5], for the first time in conversation summarization proposed a system that claimed to outperform LDA in comment topic modeling. Their method employed graph-based approach where nodes are comments taken from multiple articles. For edges, they propose a linear regression model of similarity between the nodes based on similarity features and weights trained using automatically derived training data. This is a cross-conversations summarization solution with the assumption that all comments in one conversation (i.e. form the same article) convey the same topic. However, as we previously discussed, each conversation may contain many sub-topics, which we called aspect of the conversation. Therefore, our work focuses on single conversation rather than cross-conversations summarization problem.

1.3 Our approach

An abstractive summarization method for online conversation

So far extractive summarization approach overwhelmed the majority of current researches in text summarization and claimed state-of-the-art with robust performance [55]. However, there are far too little attention has been paid to abstractive summarization, and just a few of these works considered conversation data. One possible explanation for the situation comes from the fact that extractive methods do not work well with conversation. In fact, language in online conversation is spoken rather than writing certainly hinders the cohesion and clarity of the generated summary. In the world of spoken languages, besides context, sentiment is one of the most curial factors that modify the information in expression[81, 8], therefore it is reasonable to take into account this information while summarizing the conversation. In addition, besides addressing what were discussed in the conversation, it is also necessary to point out how participants felt about them. For instance, the expression: *"Interesting how someone still believes that being narcissistic is helpful!"* should be interpreted as *negatively speaking about narcissism* rather than *mentioning about narcissism*. Furthermore, taking sentiment to consideration also allows the summarization system to capture not only the content of conversation, but also participants' opinion. Of course, sentiment analysis is an uneasy task and outside the scope of this study but its usefulness in conversation summarization is rational.

For these reasons, we propose an unsupervised abstractive method for summarizing online conversations with the embed sentiment information. With this intention, we organize our

system to as illustrated in figure 1.3. The system takes a conversation consisting of comments as input and output the summary of the conversation.

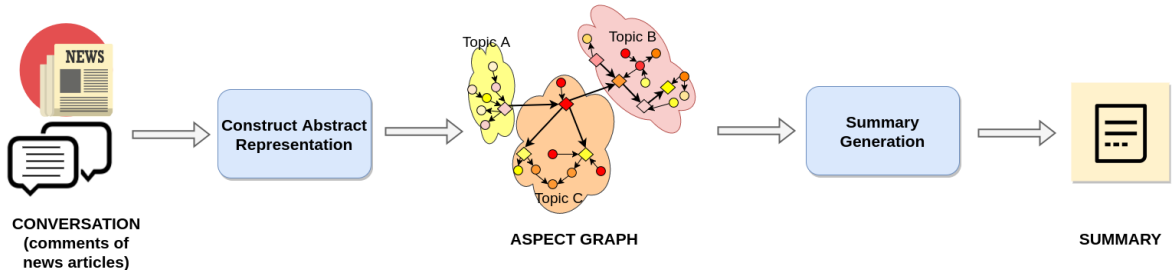


Figure 1.3 Overall organization of the proposed method

To begin, we construct an intermediate structure called Aspect Graph that captures the important information (i.e. sub-topics and sentiment information) of the conversation. Aspect graph is an abstract summarical representation for conversation and facilitates summary generation at the final step. In order to distinguish sub-topics from the main topic of the article, we call them *aspects*. By dictionary definition, aspect is a part of a situation, problem, subject, etc. [85]. Similarly, we define an aspect as one part of the main topic that was discussed in the conversation. For example, the topic *smart phone* has many aspects, some of them are concrete concepts, such as *screen*, *camera* or *price*; others are more abstract, such as *user experience*, *quality* while some concepts are cross-domain, such as *application*. In reality, as previously mentioned, there are cases that comments drift away from the primary topic as the conversation goes on, thus detected aspects may be unrelated to the topic of the article. According to the categories in figure 1.2, our method is multi-document, abstractive, generic, unsupervised, informative, mono-lingual text summarization.

The heart of the whole process is the aspect graph and its construction (which will be explained in details in the following section). In this research, we focus on proposing a definition and construction method for Aspect Graph from conversation. The later stage (i.e. text generation) is outside the scope of this work and will be carried out in future works. Since there has not been any graph-based method for conversation summarization so far, we benchmark our method on two basic tasks in text summarization: keyphrase extraction and topic modeling. The topic models allow us to further create a probabilistic model that performs topic clustering and classification for comments.

Scope of the study

As stated in early in this chapter, the research was undertaken to propose the definition and construction method for an abstract summarical representation of conversation that facilitate

future works on generating summary. The input of our system is a conversation with or without prior knowledge about the main topic, and it outputs a graph representation of the given input. This graph, named Aspect Graph points out aspects (i.e. subtopics) of the conversation with their importance in the conversation. Beside topic-modeling-like capability, aspect graph can play the role of a classifier that allows us to determine the link between each comment and each aspects of the conversation.

Although the final goal (for future works) would be constructing a complete system that can actually generate a summary, this objective is outside the scope of this study. Notwithstanding, since the construction of aspect graph aims to support the summary generation, there will be some discussion about the directions toward this system at the end of chapter 2. However, it is important to keep in mind the goal of this study is the definition and construction of aspect graph, rather than complete ATS system.

Contributions

This study calls into the need for developing novel methods on summarizing online conversation with can obviously be useful for many scientific and industrial applications. Furthermore, we proposed an organization for a system, whose heart is a generic structure that conveys summarical information of the conversation. These descriptions, however, can be considered as an essential guideline from which many derivatives can be further developed to fit different types of conversation. Finally, the proposed method on constructing aspect graph values not only in conversation summarization but also in keyphrase extraction and topic modeling task on multiple short documents.

1.4 Thesis structure

We organize this report in three for chapters. First of all, a glance about the need and related works and applications of online conversation will be provided in the *Introduction* chapter. In addition, our approach will also be explained in this chapter, accompany with the focus of this research: definition of aspect graph and its construction. To the second chapter *Aspect Graph*, definition of aspect graph, its variations, advantages and pitfall will be discussed. The third chapter *Construct Aspect Graph* will propose an effective method for constructing aspect graphs. Moreover, a predictive model derived from aspect graph will also be explained. Then, we evaluate the proposed AG construction method and models in fourth chapter *Experiments and results*. These experimental results allow use to come up with the fifth chapter, entitled *Conclusion and Discussion*. Finally, some we will discuss about future work toward a complete

conversation summarization system at the last chapter *Future works*. Two appendices complete this report. The first one provides result of comment clustering and classification in detail, while the second is a guideline for using online visualization of AG.

Chapter 2

Aspect Graph

In this chapter, we will discuss about why an abstractive representation of conversation is necessary for summarization task. More important, the chapter will provide definition of the AG, including its structure and the information that it convey. Additionally, we will also discuss about the use AG in summarization task and other types of possible applications. Furthermore, although summary generation is not inside the scope of our work, we will briefly describe how this graph can be done based on AG.

2.1 Summarical representation of conversation

2.1.1 Intermediate representation in text summarization

One common step that most ATS shared is that they derive some intermediate representation of the text that they want to summarize, from which they can identify the most important content then generate the summary accordingly [55]. Even the simplest system represents the text as list of keywords, complex systems like neural ATS represents the document as a fixed-length vector generated by the encoder. In topic-based systems, the original documents are converted into structures in which topics can be interpreted, and possibly accompany with relevant keyphrases. The whole summarization method, for this reason, critically relies of the construction of such representation. There has been many proposals for this structure, with a remarkable number of variations, complication and representation powers, including three common categories:

- **Frequency-based**: term frequency, TF-IDF, etc The representation is a simple table of keyphrases and their weights. Keyphrase with higher weight are more indicative of the topic. This simple approach requires minimal amount of computing resource. However, they are incapable to identify semantically similar words, as well as suffering from infrequent words errors.

- **Co-occurrence-based**: this approach can be visually imagine as constructing a graph in which nodes are keyphrases and edges are their co-occurrences. Frequency of these keyphrases and co-occurrences can also be captured in this graph.
- **Lexical chain**: thesaurus (e.g. Wordnet) is used to recognize semantically similar concepts and weighting them accordingly.
- **Latent semantic analysis**: patterns in co-occurrence of keyphrases are used for identify topics and giving weight for each pattern.
- **Full blow Bayesian topic models**: each document is thought to be a mixture of phrases belonging to many topics. These keyphrases are assumed to be drawn from distribution, such as Dirichlet distribution. In this approach, each topics appeared as a table of word probability on how it attributed to the topic. The representation, thus, is a table with rows whose topics and columns are keyphrase or keywords in the dictionary. The best-known method related to this approach is Latent Dirichlet Allocation [15] which is widely believed to be the best topic modeling method at the moment [48].
- **Vector-based**: this method is widely adapted by researches with the involvement of neural network (see section 1.2). The idea is to let the encoder output a fix-length vector that represents the whole input documents. This structure, however, does not have a direct interpretation, thus we can only assets it power by the generated summary.
- **Indicator representation**: arranges all sentences in the document(s) as ranking list bases on some importance features (i.e. indicators) such as sentence length, position in the paragraph or document, number of keyphrases, existent of specific phrases, etc.
- **Sentence graph**: represents the input text as a graph where nodes are sentences and edges are their relation which is defined in various ways. A famous method with this approach is LexRank[23].

2.1.2 Summarical representation for conversation

The research to date has tended to focus on regular documents rather conversation. To the case of conversation, the importance of a topic covers not only by its frequency, but also by the sentiment of people who participated the conversation. An example of this can be taken from real scientific conversation, in which some problem was pointed out in the meeting. Some unimportance, but highly agreeable topic may not take much time therefore the segment of the conversation in which these topics are discussed is relatively short. In contrast, there are

topics that less important but require a debate, therefore they are discussed longer. Therefore, frequency only partly solves the question about the importance of aspects of the conversation. The other haft of the problem can be addressed by sentiment analysis, also known as opinion mining.

Under those circumstances, a reasonable intermediate representation for summarizing conversation need to convey the following information:

- **Aspects** (i.e. sub-topics): a conversation can mention about many aspects about the main topic, given before the departure of the conversation. Although some participants may know about the main topics, there is often the repose directly to the previous comment or talk, which may refer to the old or trigger a new aspect. In common terms, aspects are sub-topics of the conversation. However, the term of *sub-topic* is partially incorrect in the context of conversation, because the conversation may turn to new issues that is neither an instance nor a specific case and a side of the main topic. Therefore, in this research we use the term of *aspect* instead.

- **Keyphrases**: are important key words or phrases that attributed to the aspects. These keyphrases should be not only characterized for the conversation, but also has a universal integrity. For example, the aspect of *boat* in conversation A owns the keyphrase *stainless steel propeller*, and the same aspect should not have the keyphrase *vertical stabilizer* in conversation B. This indicates a certain amount of semantics agreement between keyphrases within an aspect.

- **Sentiment** (i.e. opinion): is the information of how participants think about the aspect and every possible keyphrase that it comprises. Sentiment information can be indicated in various levels. The simplest level is polarity between positive and negative. More complex categorizing strategies would include subjective-objective scale, multiple class sentiments, such as *angry*, *sad*, and *happy*, etc. Sentiment analysis is broad and trending study field, therefore there a many ways to extend the representation from this direction.

- **Summarical information**: this information characterizes the representation for summarization task, compare to native topic modeling methods. This information is the indication of importance of aspect (and possibly keyphrase) from quantitative and qualitative point of view. There are various ways to define this information in different degrees of sophistication and summarization purpose. The simplest and most general score is frequency. However, summarical information should rely on the application and summarization target. For example, a presentation for politic conversation among multiple parties should contain the information about the *influence* of ideas and aspects.

On another hand, in a conversation about an entertainment, they summarical information should be the aspects' applause.

Among these four characteristics for an ideal summarical representation for discussion, the aspects and keyphrase hold the contents of the conversation. It answers the question *what did people discuss?*, therefore two conversations may end up with a very similar representation. Sentiment and summarical information are identical for every conversation, even when these conversations were carried out by the same group of people. It captures the opinion of participants, that is *How did they talk about these contents?*.

Since we are the first people who define this structure, we will propose a base architecture for a presentation, namely *Aspect Graph*.

2.2 Aspect graph definition

2.2.1 Basics of AG

Aspect Graph is a graph-based abstractive summarical representation for conversation. It carries the semantic and sentiment information of the conversation as well as suggestion for summary generation. Furthermore, AG can visually demonstrate the conversation and support a wide range of graph-based operations.

At a glance, aspect graph G consist of a set of nodes V and set of edges E that connect nodes together:

- **Node** v_i denotes a *group* of keyphrase with who are semantically equivalence in the context of the document. For example, in a conversation about trash collection, three keyphrases *trash bin*, *wastebin* and *bin* are semantically equivalent and therefore, they are grouped into a single node. Each node contains three attributes: frequency, label and sentiment score. Frequency of a node is the sum all its members' frequencies, while the sentiment score is a harmonic average of the sentiment of its members. Each node has a label taken by the most frequent keyphrase in the whole group.

- **Edge** e_i represents a potential relation between two nodes. Every edge has two attributes: direction and frequency. All edges are directed, starting from the subject to the object of the potential relation. For example, the expression *The dog chases the cat* can be represented with two nodes each contains only one member *cat* and *dog*. The edge between these two nodes originates from *dog* to *cat* and the label for the potential relation is *chase*. There could be multiple edges associated to two nodes, however every node could not have a ring edge. It's also important to keep in mind that the edge's label is

expected to be an action that has a subject and an object. However, such action may not always explicitly exist between entities, therefore we also interpret edges as *dependency relation* where target node is the dependant. Frequency for an edge is counted by the occurrence of the relations from any member of source node to any member of target node.

Although the standard structure of AG was introduced with nodes as group of keyphrase and edges as directed potential relation, it is possible to simplify the graph to least complex or make it more sophisticated. For instance, a minimal version of AG contains keywords as nodes and co-occurrence as undirected edges. More complex derivations can introduce more syntactic information to the graph, such as word order in sentence, rule-based regex chunking, etc. Technique speaking, AG can be referred as a family of sentiment-embedded representation method for aspects discussed in a conversation.

2.2.2 Topics

One of the most interesting features of graphs is community structure. A community (i.e. cluster) comprises of nodes that own many edges joining nodes within the same cluster and comparatively few edges connecting to other clusters. Each community is an adequately independent compartment of a graph, which can be morphologically imagined as organs in the human body[25]. This definition of community in graph fits the intrinsic properties of a topic, thus, topics in AG are defined equivalent to communities. In another word, each community in AG represent a topic. Within community, there are nodes that connected to each other and the importance of nodes indicates the importance of all the keyphrases that it contains. There are several ways to qualify the importance of a node. The most common indicator is centrality score that can be computed with various methods, such as PageRank algorithm, degree, closeness, etc.

Community detection in graph is a hard-clustering process; hence there is no node that any two communities share in common. As a result, the set of keyphrases for topics are totally distinct from each other. This is an important different between AG and Bayesian topic models (e.g. LDA) where topic is represented as a vector contains the probability of every word in dictionary contribute to the topic.

2.2.3 Sentiment and summarical information

Sentiment information in AG is the polarity between positive to negative. This sentiment score is stored as properties of nodes and attributes to the cluster's sentiment score. Indeed, the

sentiment information of community (i.e. general participants' opinion about the aspect) is compute as the average value of all the nodes inside that community. Likewise, we compute the frequency of a community by summing up all nodes that it contains. The harmonic value between sentiment score and frequency is considered as the indicator for the importance of a topic. This same pattern can also be applied for determining important nodes inside each community. Frequency and sentiment score together make up the summarical information of the whole conversation all four levels: conversation, aspect, node (i.e. group of equivalent keyphrases) and individual keyphrase. Furthermore, the relation between nodes depicts the historical usage of relevant keyphrase, which is useful for generating summary.

In order to support AG visualization, we assign two additional properties to nodes: size and colour. The size of a node illustrates its' important in the conversation, larger size indicates a more significant role the node played in the conversation. In a similar manner, nodes are coloured by its sentiment score in a gradient from green (strongly positive) to red (strongly negative). Communities also have these mentioned properties, depending on the frequency and sentiment score of is nodes.

2.2.4 The two-layer structure

As was discussed in 2.1.2, information in AG can be arranged in two groups according to their functional utilities. Therefore, we structure AG in two layers: semantic layer and sentiment layer.

Semantic layer

Similar to the role of the skeleton of human body, semantic layer is a base of an AG. It answers the question about *"what did the conversation talk about?* and contains most of the semantic information of the graph. With a similar appearance and function as a concept network, the semantic layer encompasses nodes and edges without sentiment and frequency information. Also in semantic layer, communities are visualized as cluster of node with a specific label that indicates the general semantic information of the corresponding topic. Figure 2.1 exemplifies the semantic layer of an aspect graph about Brexit[1]. We can interpret from the graph that the conversation was surrounding five aspects of Brexit, namely politic, social, education, economics and culture. From politic aspect, people concerned that the *referendum* has *leaded* to *David Cameron resigns* as well as *Brexit contagion* can *affect* the *Greece*. They also mentioned

[1]The popular term for the prospective withdrawal of the United Kingdom from the European Union (EU) following the national referendum on 23 June 2016. After the event, the United Kingdom's president at that time, resigned his duty as prime minister and the government was taken over by Theresa May the leader of the Conservative Party.

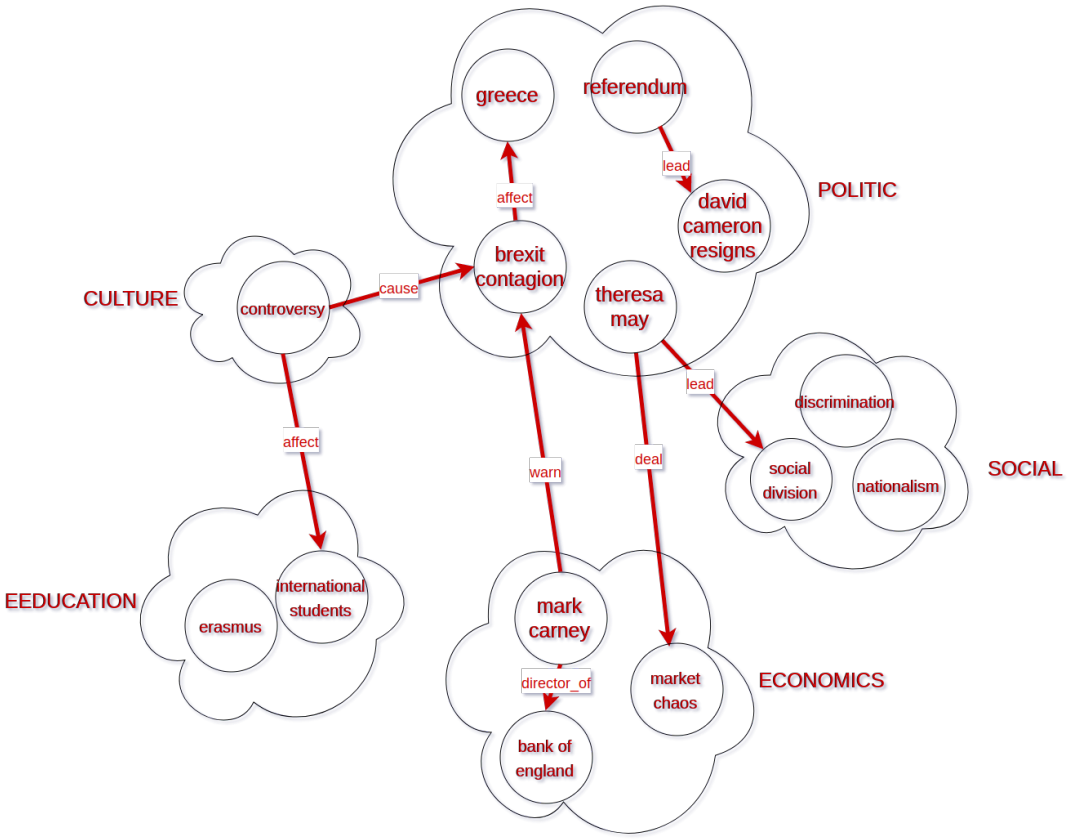


Figure 2.1 Semantic layer of an example AG about Brexit

about *Theresa May* who *leaded* the *social division*, which is one *social* aspect of the event. At the same time, people talked about *economics* aspects related to the event related to *Mark Carney* who is the *director of bank of England* and mention about how *Theresa May deals* with *market chaos*. In the same manner, one can easily understand the content of the conversation while viewing the semantic layer.

Sentiment layer

Acting like the muscles or skin of human body, sentiment layer contains the summarical information built on the AG base (i.e. semantic layer) to answer the question *"What is the participants' opinion about aspects of the conversation?"*. It captures information about the importance of topics, nodes and support the visualization with node color and size. For example, figure 2.2 is the sentiment layer of an aspect graph whose semantic layer was early explained in

figure 2.1. In sentiment layer, node labels, edges and communities are exactly the same as those at the semantic layer, except that the color and size of node and communities are visualized according to their sentiment and frequency. In the example, we can interpret from sentiment layer along with information from the semantic layer that most participants had negative feeling about *politic* and *social* aspects about Brexit. In contrast, they thought positively about *culture* and *economics* aspects of the event while most comments are neutral about *education*. To be specific about politic aspect, people expressed their *unfavorability* about *Thresa May*, characterized by very negative sentiment score, although they think the *referendum*. Similarly, a simple rule-based system can easily interpret the content of the summarical information represented in this layer to generate the complete summary. Additionally, sentiment layer provides an intuitive viewpoint at AG and its content, hence, insight of the graph can be taken directly from the graph, rather than reading the final summary which relied on the summary generation step.

2.3 Operations on AG

The separation of AG into semantic and sentiment layer is due to the different in their operational benefits. Semantic layer contains units that carry the content of the conversation. At this layer, two conversations talking about the same topic may end up to be very similar because concepts and their relation are many, but limited. On the contrary, people tend to react differently time by time, even when they are talking about the same topic. Therefore, there would be numerous possible variation of sentiment layer even when their semantic layers are similar. This variation also depends on the length and the number of participants of the conversation. This suggests that every operation in AG may suit only one of two layers, corresponding the purpose of the operation: content or sentiment information.

Summary generation

As AG was deliberately built for summarization purpose, its primary use is for generating summary from *sentiment layer*. Generating text from AG can be addressed in many ways and able to generate both abstractive and extractive summary. Indeed, it can be considered as a graph traversing problem, such that community and nodes are visit in the order of importance. As a result, producing a good summary is equivalent find out the best traversing strategy that maximizes the coverage while minimizing the number of steps. In our case, coverage can be understood as the total importance of all visited nodes. For each step of this process, both extractive and abstractive approaches can be applied. With the abstractive approach,

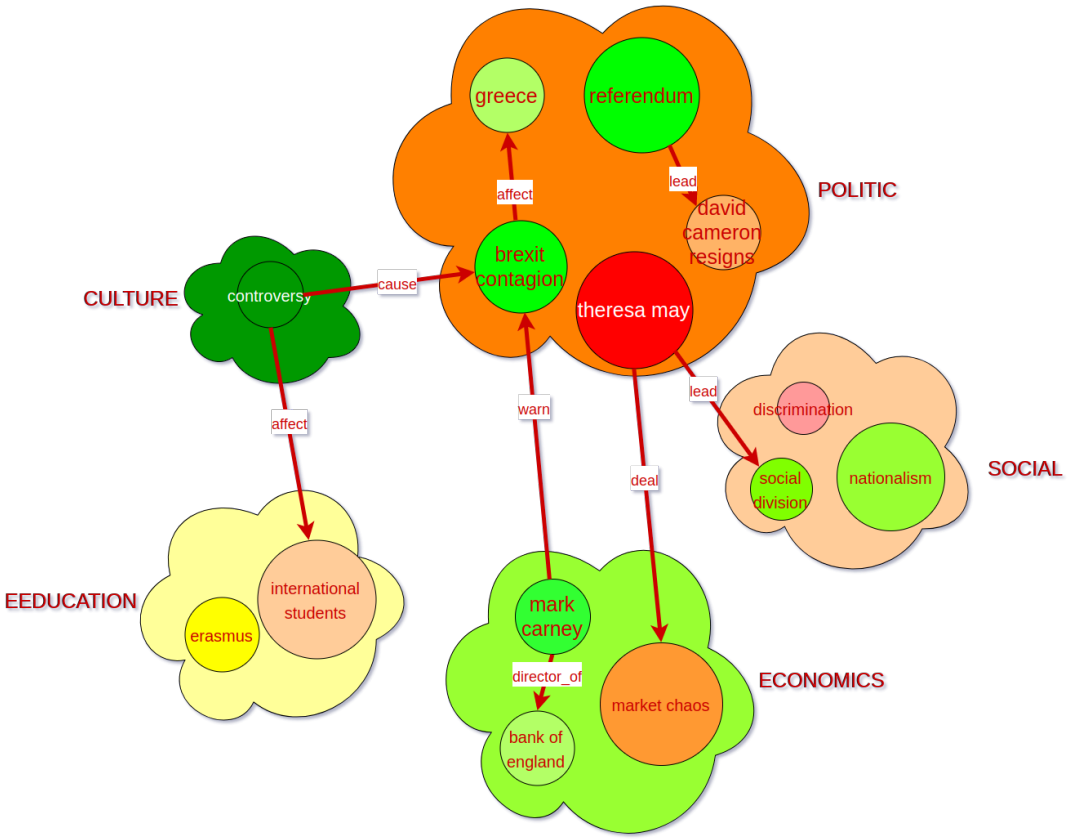


Figure 2.2 Sentiment layer of an example AG about Brexit

sentences are generated according to the subject-relation-object pattern suggested from the graph. External knowledge sources (e.g. dictionary, ontology, etc.) and language models can be employed to enhance the performance of this process. Furthermore, if we translate the AG to another language before applying the explained text generation procedure, it would be possible to generate the summary in a language that is different from the source language. For this reason, AG can be used for cross-lingual summarization systems.

Besides the use for generating an abstractive summary, AG can be used for extractive summarization. In this case, instead of generating text, the system only needs to pick up sentences where paraphrases generated pointed out by AG. However, it is not difficult to realize that extractive summarization is not fit the characteristics of online conversation. Furthermore, extractive approaches do not make use of the sentiment information, therefore they work well at semantic layer.

Conversations composition

When it comes to the case of long conversation (e.g. congress meeting, Negotiation Meeting) that contains multiple sub-conversations, it is necessary to compose the content and sentiment of participants across these individual conversations. Another practical example for this need is when you want to summarize not only the comment of one news article but also multiple articles. Another important application is real-time conversation summarization where we have to summarize a conversation as it is going on. Thanks to the development of graph theory in nearly 3 centuries since its first publication of Leonhard Euler on the Seven Bridges of Königsberg, various graph composition methods has been proposed. The simplest method is literally union of the node sets and edge sets (i.e. graph contraction), while more complicated methods such as series-parallel graph composition can also be used.

Theoretically, we can perform infinite number of compositions between conversations, which mean AG is a scalable method. However, if the number of conversations is large, composition should only be applied at semantic layer, rather than sentiment layer. It is because sentiment layer is biased by the frequency and length of the conversation. Therefore, composing a long and short document together with a vast number will bias the summarical content toward some long conversation. Henceforth, it is more reasonable to do such composition at semantic layer, and when the number of conversations is large enough, AG will look like a semantic network, which support reasoning.

Reasoning on AG and question answering

One of the most significant characteristics of abstractive text summarization is it ability to re-interpret the information therefore can produce a more general summarization. An example for this ability is grouping keywords together and representing them as a single unit using a more abstract term. For instance, an abstractive summarization system would refer all *Toyota*, *General Motors*, *Daimler* and *Ford* as *automobile companies*. An AG, built from sufficient large number of conversation is similar to a semantic network that supports inference. This advantage is also useful for producing query-focused summarization systems for conversation.

2.4 Advantages and disadvantages of AG

Advantages

As graph-based representation for conversation, AG is more powerful than other ranking list-liked representations, indeed, AG

- Contains participants opinions for the conversation's aspect and for each keyphrase.
- Groups semantically similar keyphrases in groups, so they can be treated as equal important unit and avoid redundancy.
- Specifics for conversation and short document summarization.
- Scalable with the capability to handle long conversation.
- Supports re-interpretation words and, to a certain extend, simple reasoning.
- Conveys some meaning in a transparent manner.
- Can be intuitively visualized, easy to read and get information.
- Has potential relation that supports summary generation.

Furthermore, AG can be applied to both abstractive and extractive summarization types; therefore existing methods in extractive summarization can easily extend their work with AG as intermediate representation

Disadvantages

Due to the fact that AG encompasses many natural language processing techniques, many weaknesses remain:

- Aspects contain a unique set of keyphrase, which is not always true in practice. For example, both two topics *animal* and *airplane* have *wing*. As a result, the community detection method will either merge both two topics into one, or leave the keyword *wing* to one of two topics. In either of these two cases, the result is unexpected;
- Despite the variety of entities, they can be shown in the same node;
- Edges do not have a standard set of name. They are learnt automatically and not all of them form a function or predicate;
- Edges only represent binary relation, while there exist unary, ternary and n-ary relations.

Chapter 3

Construct Aspect Graph

In the previous chapter, we proposed an abstractive summarical representation of conversation called Aspect Graph(AG). AG comes under the graph structure where every node is a set of semantically equivalent keyphrases, and edges are potential relation indicating the dependency relation between nodes. Nodes carrying frequency and sentiment information are clustered into fairly independent compartments called community, which is equivalent to an aspect (i.e. sub-topic) of the conversation. These communities inherit sentiment and frequency from the members which define the importance of its corresponding aspect in the conversation.

Before we present our experiments, this section describes the design and techniques used for building AG from conversation. The section 3.1 illustrates the overall landscape of the system as well as points out output for each phase. Follow this, the next four sections discuss how we build a raw graph from input before performing community analysis to find aspect, then assign an abstractive label to each aspect and finish by computing sentiment information for elements of the graph. This chapter closes with a discussion on deriving a predictive model from AG for clustering and classification fresh comments.

3.1 Pipeline

Our system takes comments in a conversation as input and output an AG with a process consist for four phases. An overview of our idea is demonstrated in figure 3.1. The procedure begins by constructing a base graph, which will facilitate community detection in the latter phase. It produces a base graph consisting of nodes and edges as previously discussed in previous chapter. Meanwhile, sentiment and frequency are extracted and stored as nodes and edges' property (i.e. history). Following this, we run a community detection algorithm to find communities from the base dependency graph. Detected communities, each comprises a set of nodes and edges make up a potential aspect, illustrated by the boundary surrounding nodes in figure 3.1.

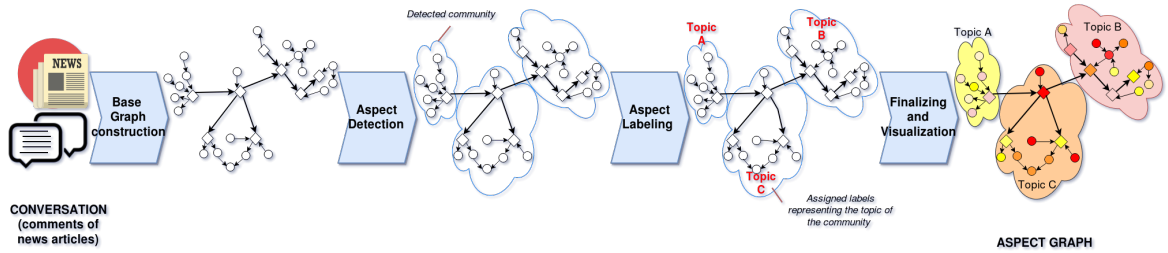


Figure 3.1 Pipeline for AG construction

This followed by assigning an abstract label covering the main idea of the aspect according to its member. Up to this point, the system completed conducting the semantic layer of the AG. At the last phase, the system computes information about importance of nodes and aspect by putting frequency and sentiment information of nodes together. Finally, it compute sentiment scores for aspects and decorates the graph with color according to the computed information and generate description file for online visualization. In the next four section, we will discuss in detail the design and techniques used in every phrase in this pipeline.

3.2 Base Graph construction

Conversation about news article consists of a series of comments arranged in temporal order. Thus, the graph construction for the whole conversation comprises the construction of elemental graphs for each of the comments following by composition step. We use the term dependency graph to call these elemental graphs. Each dependency graph a directed graph that consist of nodes which are keyphrases and edges are the dependency relations between these keyphrase (see 3.2.1 for details about this type of relation). Nodes of dependency graph convey frequency and sentiment information of the node in the text (i.e. the comment). It is important to avoid misconception between the dependency graph and AG. Dependency graph contains information (e.g. keyphrase, frequency, sentiment) about a document, not an entire conversation as what AD does. Additionally, a node in dependency graph literally represents a single keyphrase, while node in AG is a set of keyphrase that semantically equivalent. To this point, the base graph is still noisy and contains keyphrases that are similar in semantic. Therefore, we unify these keyphrases together before applying filtering to end up with a clear base graph. This process is demonstrated in figure 3.2

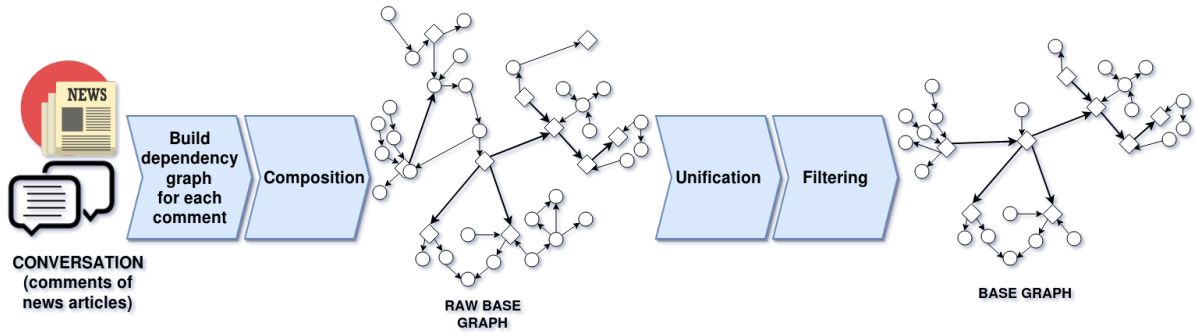


Figure 3.2 Base graph construction process

3.2.1 Build dependency graph for each comment

Dependency

A vast number of researches to date has employed co-occurrence of words as indicator of relation between units of information (e.g. words, phrases, entities, etc.) in text. Co-occurrence refers to the above-chance frequent occurrence of two units appear alongside each other in a text document corpus. There has been a many proposal about how to understand and interpret co-occurrence, however there have not been a standard definition yet. One of the widely adapted interpretations is that it indicates the semantic proximity or an idiomatic expression between linguistic elements [41] which became the assumption for distributed semantic and other applications in computational linguistic. Still, the debates go on, as we know there existing many cases where co-occurrence may not fully address the semantic relation between terms. Perhaps the best counter examples for the above-mentioned interpretation are synonyms and plural because they rarely appear together in the same sentence or text although they are obviously semantically equivalent. To overcome the deficiency of naive co-occurrence based approaches, researchers add syntactic features to their method to make the relation between grammatical units more meaningful. One of the most effective way to define this relation is *dependency*, which was first concretely present in the by Sámuel Brassai. Lucien Tesnière, a French linguist, laid the lay the foundation for class of modern syntactic theories based on dependency widely known as dependency grammar, which opposes to constituency grammars. In simple terms, dependency is the notion that linguistic units (e.g. words, phrases...) are connected to each other by directed links. There are four types of dependency, including:

- **Semantic dependencies** come in terms of predicates and their arguments where the arguments are semantically dependent on that predicate. A set of standard dependencies was recently proposed Sebastian Schuster et al. [72] consists of 40 dependency relations that capture common grammatical relations and relations between content words. How-

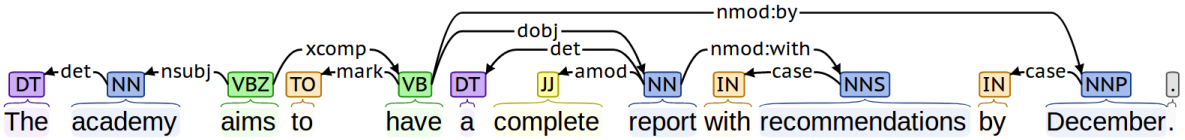


Figure 3.3 Example of dependency tree

ever, the set of dependency relations are not limited to grammatical function because predicates can be formed in various ways. For instance, from the sentence *the dog chases the cat.*, we can form a relation from *the dog* to *the cat* with the relation *chases*.

- **Morphological dependencies** come when words or parts of a words influence each other. In that that case, we say the latter is morphologically dependent on the former.
- **Prosodic dependencies** address the the behavior of clitics. Clitic is a syntactically autonomous element (i.e. morpheme) that has syntactic characteristics of a word, but depends phonologically on its host(e.g. word or phrase). Therefore, clitics are the dependant of its host. An example of clitic in English is *'s* in *it's* which stands for both *it has it is*;
- **Syntactic dependencies** focus on the presence and the direction of syntactic dependencies. So far, the most supported presence of dependency grammar is hierarchy, but the problem is of course often open to debate.

Figure 3.3 demonstrates an example dependency tree for the sentence *"The academy aims to have a complete report with recommendations by December"*. The corlored rectangles contain part-of-speech tag for every word in the sentence. Arrows indicated the dependency relationship between them which labels follow Universal Dependencies version 2 [1]. For instance, *nsubj* (nominal subject) is a nominal which is the syntactic subject and the proto-agent of a clause; *dobj* (direct object of a verb phrase) is the noun phrase which is the (accusative) object of the verb.

Among these different types of dependency relation, semantic dependency is the most suitable for constructing the dependency graph for many reasons. First, it depicts the relationship between lexical units, e.g. words, phrases, which are useful for detecting aspects rather than grammatical units. Second, the predicate-arguments efficiently support text forming sentences which is useful for summary generation. Last and most important, predicate-arguments pattern allows us to consider keyphrase as *terminal* instead of a single lexical unit. This is a prominent approach because the important of a word or phrase now can go beyond its occurrence in text,

[1] see more at http://universaldependencies.org/

to be its role when forming relation with other keyphrases. In another word, it is the use of keyphrase justifying its importance, rather than its existence.

Build dependency graph from text

Turing now to the procedure of build up dependency graph from text (e.g. comment, paragraph,etc.). Since comments often have noise and special characters (e.g. escape characters, emotions, etc.), we first clean the text by removing hyperlinks, quotation, truncating number, and unnecessary unicode characters. Then, the system detects noun phrases and named-entities in order and treat them as a single word. This follows the algorithm 1.

Algorithm 1 (A1) describes the steps involve in generating dependency graph from text. It starts by the initialization of an empty directed graph g the represent the text before performing coreference resolution (A1:2). This is followed by tokenizing the text to sentences and then extending the graph g with the dependency relation extracted from these sentences (A1:4-41). Indeed, for each sentence, the algorithm first employs a dependency parser in order to output grammatical relations in the enhanced universal dependencies representation. This tree-like structure then is convert to a list $dep\_tripples$ whose element is a triple contain the source, the relation and the target keyphrase (A1:6). This follows by a rule-based refinement (A1:2) that calls for algorithm A2.

The rule-based refinement algorithm (A2) first filter out triples whose dependency relation is in our interested list (A2:4-8). As each dependency tends to connect between specific types of part-of-speech tag, this step also lead to part-of-speech filtering. Then, it searches for the existent of three forms of predefined patterns to compose them to a single triple. There are two types of patterns to be searched for:

1. Node patterns: $src\_phr \overset{rel}{\rightarrow} trg\_phr$
2. Edge patterns:
   - $src\_phr \overset{rel_1}{\rightarrow} int\_phr \overset{rel_2}{\rightarrow} trg\_phr$;
   - $src\_phr \overset{rel_1}{\rightarrow} int\_phr \overset{rel_2}{\leftarrow} trg\_phr$;
   - $src\_phr \overset{rel_1}{\leftarrow} int\_phr \overset{rel_2}{\rightarrow} trg\_phr$;
   - $src\_phr \overset{rel_1}{\leftarrow} int\_phr \overset{rel_2}{\leftarrow} trg\_phr$;

where src_phr, trg_phr, int_phr are part-of-speech tag of source phrase, target phrase, intermediate phrase while rel, rel_1, rel_2 denote normal, first relation and second relation respectively.

For example: three dependencies for the sentence *smart dog chases cat* are:

1. $dog[NN] \overset{amod}{\rightarrow} smart[JJ]$

Algorithm 1 *BuildDependencyGraph*(txt)

Input: : text to build dependency graph from txt and options Θ
Output: : constructed graph g

$g \leftarrow CreateEmptyDirectedGraph()$
$coref\_txt \leftarrow CoreferenceResolution(txt)$
$sentences \leftarrow TokenizeSentences(coref\_txt)$
for all $sentence \in sentences$ **do**
  $dep\_tree \leftarrow DependencyParse(sentence)$
  $dep\_triples \leftarrow Tree2Triples(dep_tree)$
  $dep\_triples \leftarrow RuleBasedRefine(dep_tree)$
  $sentiment\_score \leftarrow SentimentAnalysis(sentence)$
  $new\_nodes \leftarrow \varnothing$
  for all $(src\_phrase, relation, trg\_phrase) \in dep\_triples$ **do**
    $new\_nodes \leftarrow \{new\_nodes \cup src\_phrase \cup \{trg\_phrase\}\}$
  end for
  for all $new\_node \in new\_nodes$ **do**
    if $new\_node \in g$ **then** ▷ Node already exists in graph
      if $sentiment\_score > 0$ **then**
        $g.IncreaseNodePositiveCount(new\_node, 1)$
      else if $sentiment\_score < 0$ **then**
        $g.IncreaseNodeNegativeCount(new\_node, 1)$
      else
        $g.IncreaseNodeNeutralCount(new\_node, 1)$
      end if
    else ▷ Add new node to graph
      $g.AddNewNode(new\_node)$
      $g.SetNodeFrequency(new\_node, 1)$
      if $sentiment\_score > 0$ **then**
        $g.SetNodePositiveCount(new\_node, 1)$
      else if $sentiment\_score < 0$ **then**
        $g.SetNodeNegativeCount(new\_node, 1)$
      else$g.SetNodeNeutralCount(new\_node, 1)$
      end if
    end if
  end for
  for all $(src\_phrase, relation, trg\_phrase) \in dep\_triples$ **do**
    if $g.EdgeIsInGraph(src\_phrase, relation, trg\_phrase)$ **then** ▷ Edge existed
      $g.IncreaseEdgeFrequency(src\_phrase, relation, trg\_phrase, 1)$
    else ▷ Add new edge to graph
      $g.AddEdge(src\_phrase, relation, trg\_phrase)$
      $g.SetEdgeFrequency(src\_phrase, relation, trg\_phrase, 1)$
    end if
  end for
end for

Algorithm 2 $RuleBasedRefine(dep\_triples, \Theta)$

Input: : list of dependency triples $dep\_triples$ and dependency refinement options Θ
Output: : list of refined triples $result\_triples$

```
1:  prefered_triples ← ∅
2:  result_triples ← ∅
3:  prefered_dependencies ← Θ.GetPreferedDependencies()
4:  for all (src_phrase, relation, trg_phrase) ∈ dep_triples do
5:      if relation ∈ prefered_dependencies then
6:          prefered_triples ← {prefered_triples ∪ {(src_phrase, relation, trg_phrase)}}
7:      end if
8:  end for
9:  for all first_triple ∈ prefered_triples do
10:     for all second_triple ∈ prefered_triples do
11:         composed_triple = Θ.compose(first_triple, second_triple)
12:         if composed_triple then
13:             result_triples ← {result_triples ∪ {composed_triple}}
14:         else
15:             result_triples ← {result_triples ∪ {first_triple}}
16:             result_triples ← {result_triples ∪ {second_triple}}
17:         end if
18:     end for
19: end for
```

2. $chases[VZ] \overset{nsubj}{\rightarrow} dog[NN]$

3. $chases[VZ] \overset{dobj}{\rightarrow} dog[NN]$

Matching the dependency 1 to the pattern $[NN] \overset{amod}{\rightarrow} [JJ]$ give us a single node $smart_dog[NN]$. This new node replaces the role of all $dog[NN]$ in the extracted dependencies (e.g. dependency 2), thus turn the dependency 2) to:

$$chases[VZ] \overset{nsubj}{\rightarrow} smart\_dog[NN] \tag{3.1}$$

Accompany this rule with triple 3, we match them to the pattern

$$[NN] \overset{nsubj}{\leftarrow} [VZ] \overset{dobj}{\rightarrow} [NN] \tag{3.2}$$

returns one single dependency relation:

$$smart\_dog[NN] \overset{chase}{\rightarrow} cat[NN] \tag{3.3}$$

There are totally 8 node patterns and 42 edge patterns defined in the system. Node patters allow us to establish long phases (e.g. adjective phrase, compound, names, etc.) Edge patterns, most significantly allow the graph capture the potential relation between keyphrases, apart form grammatical dependency relations.

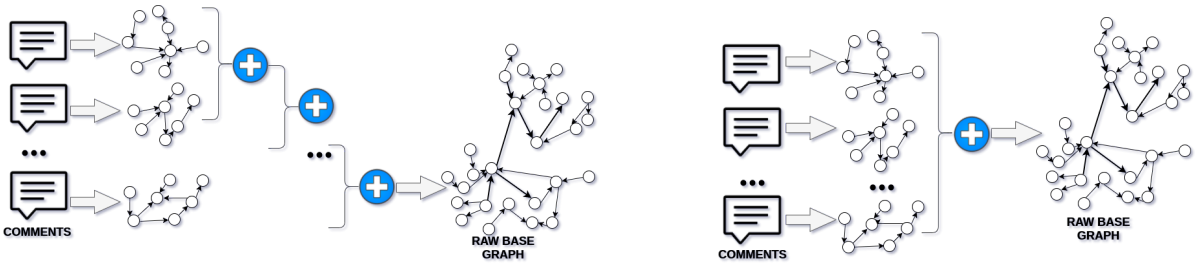
After refinement, source and target keyphrase of the triples are considered as candidate of nodes and stored in the set *new_nodes* (A1:9-12). Meanwhile, we compute the sentiment polarity *sentiment_score* for the sentence and assumption that every phrase and word in this sentence takes this score as its own sentiment score. Then, all candidate keyphrases in *new_nodes* is added to g, accompany with their frequency and sentiment information(A1:13-32). If the candidate has not already existed in the graph, it will be added to g with default frequency set as 1. Each new node is initialized with three sentiment frequency properties: *positive frequency*, *neutral frequency* and *negative frequency*. *Positive frequency* counts its occurrences in a sentence with positive *sentiment_score* while *negative frequency* counts for negative occurrence. *Neutral frequency* addresses its existence in sentences whose *sentiment_score* is exactly zero. In case the node has been added to g by a previous sentence, its frequency will increased by one and sentiment count also increase according to the sentence's *sentiment_score*. In a similar manner, the algorithm adds edges to graph and records their frequencies(A1:33-40).

Once the algorithm 1 has completed, each comment will have a corresponding dependency graph representing its keyphrases and their dependency relation. The next move is to put all these elemental graphs together in *Dependency graphs composition* phase.

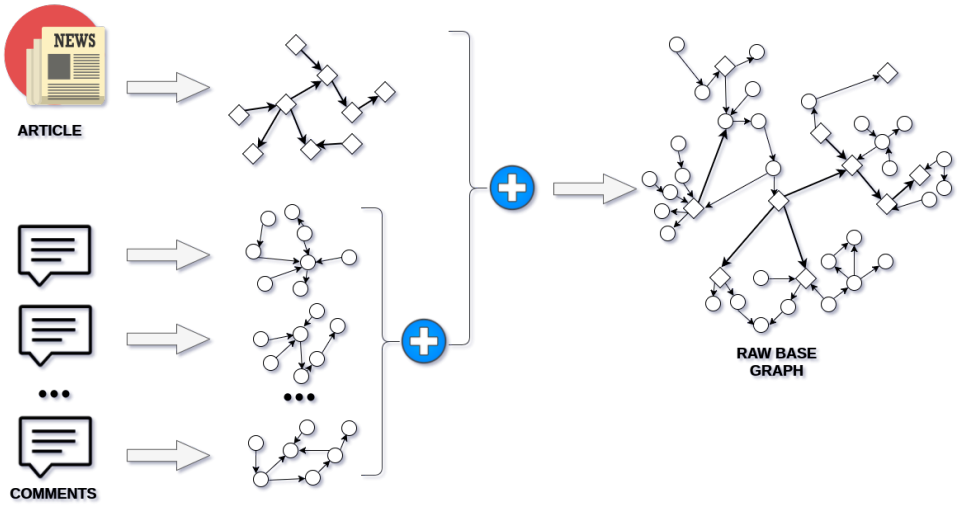
3.2.2 Dependency graphs composition

This phrase aims to compose all dependency graphs representing each comment to form a big graph that represents the whole conversation, namely *raw base graph*. Beside the comments, the article's content contains the conversation's primary topics, thus we designed our system to work both with and without knowledge about the article. Figure 3.4 shows the composition used for composing comments only and comment accompany with article content.

Figure 3.4 *(a)* demonstrates two ways of composition for comment. First, we extract *dependency graph* for each comment, then compose them as a whole. Second, we depart by build up the graph for the first comment, then gradually extend this graph by composing it with the other graph. In theory, both these two methods can build up the same or similar base graph. However, these two methods are different in their use cases. The first strategy is useful when the conversation is relatively short or less complex because it requires resources and computing power because it needs to store the dependency graph of each comment, then process them all together. The advantage of this scheme is that it supports concurrency therefore saves computing time, although this also means that temporal information of comments will not be



(a) 2 composition strategies for comments



(b) Composition comments and article

Figure 3.4 Composition method for elemental graphs

taken into account. In contrast, the second approach is an incremental process for expanding the base graph over time which is suitable for summarizing long conversation in real-time. The ultimate advantage of this approach is its ability to exploit temporal order of comment and requires less computing resources at a moment. However, it may take time compared to the case when order is not considered.

In case we want to take the original article to consideration, we can use the dependency graph extracted from the article to initialize the graph construction as shown as figure 3.4. This approach is especially effective when incorporating with the incremental approach of comment extraction because they help avoid detected aspect from going too far from the main topic of the article. This advantage, however, is also a disadvantage because new aspects rise from the conversation can be ignored. The use of article content is eventually necessary in some cases, especially when the conversation is too short or contains too little information. However, initializing AG with the article's content is an optional step because many conversations come without knowledge about the original topic. Chat is a simple but popular example of this kind of conversation.

Compared to comment what generally contains significantly less number of sentence and information, content of the article is much longer and more informative. They are usually organized in paragraphs with many subtopics. Therefore, before applying the same dependency graph extraction method as we do with comments, the system segmentizes the article in to *subtopic segment* using a famous method named TextTiling[30]. Proposed by Marti A. Hearst, the technique divides texts into multi-paragraph units that represent subtopics bases on patterns of lexical co-occurrence and distribution. This is followed by extracting dependency graph for each segment, then composes them together to form dependency graph for the whole article.

3.2.3 Unification

Raw base graph carries all essential information from the conversation. However, there remains redundancy because similar keywords are treated as individual nodes. Also, the graph is heavily partitioned , and node's connections is rather local (i.e. bias to their occurrence in comments) rather than topical (i.e. indicates their coincidence together in a topic). Therefore, the system carries out a phase called *unification* whose basic idea is to accumulate similar keyphrases to a form a single node that represent them a whole. Two keyphrases are considered as semantically equivalent if they both have the same lexical form or semantically similar.

To determine semantic similarity between the keyphrases represented by two nodes, we compute cosine similarity score between their vectors in vector space representations. This score ranges from -1 (i.e. completely different) to 1.0 (i.e. perfect similarly). Those pair of keyphrases whose semantic similarity are greater than a threshold is considered to be similar. In our case, we set the threshold to 0.9. For deriving vector representation for keyphrases, we employ GloVe a global vectors approach for word representation introduced by Jeffrey et at. [64] which claim better performance than previous vector-based distributed semantic models such as *word2vec* and *skip-gram*. However, GloVe was built to generate vectors representing words, not phrases. Therefore, we derive the vector of a phrase by simple weighted additive composition of the vectors for all words it contains.

There are two common ways to indicate that two node are similar in the graph, as shown in figure 3.5 (semantically similar keyphrases are denoted as black circles):

- **Contraction** identifies the two nodes as a single node incident to any edge that was incident to the original two nodes. The properties of a contracted node is the sum of the original two nodes' properties (i.e. frequency, positive frequency, negative frequency, neutral frequency). The contraction results in a new node which somehow can be considered as a set that contains all semantically similar, which is the essential definition of nod in aspect graph. Beside, the new node takes the keyphrase of its most frequent

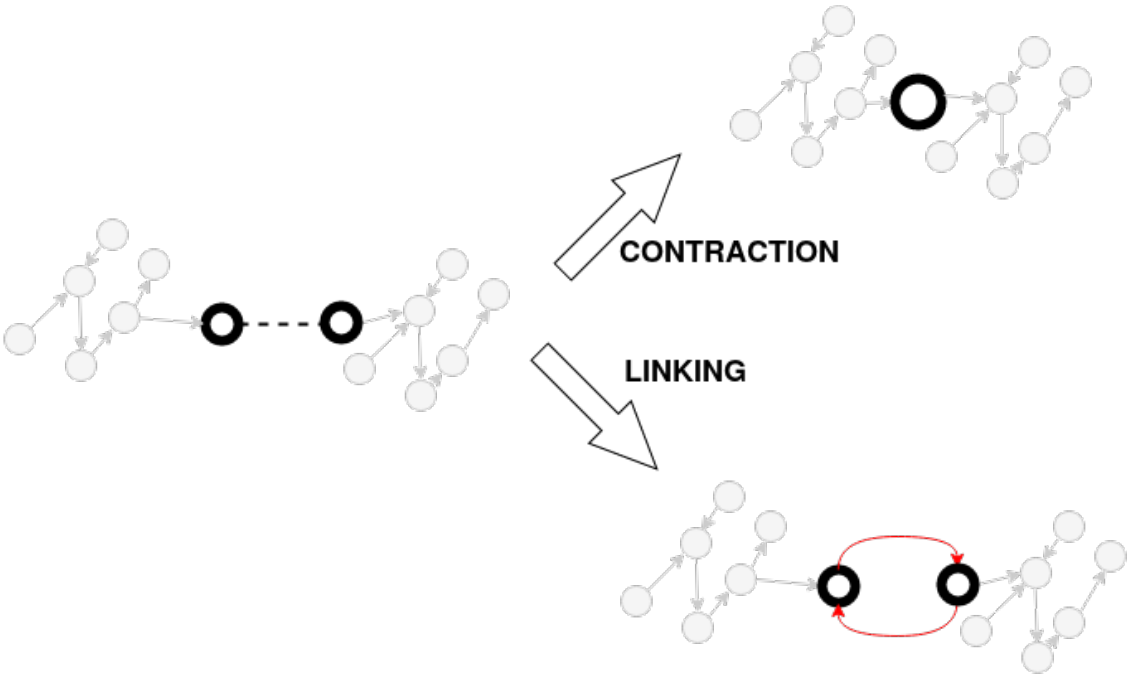


Figure 3.5 Contraction and linking nodes

member as its own label (i.e. the representative keyphrase for the whole set). We then use WordNet lemmatizer to return every word in the keyphrase to its base form;

- **Linking** create a loop consist of two opposite directed edges between two nodes (red arrows in the figure). Their relation names are assigned as *unknown*. Also, their frequency, positive frequency, negative frequency, neutral frequency are set to 1,0,0,1 respectively.

Both contraction and linking create a way connect similar phrases. However, our experiments lead us to the observation that contraction is more efficient because it generally costs less computing resource and reduce the complexity of the graph. Furthermore, contraction results in a more intuitive graph and useful for reducing redundancy. Therefore, we use this approach for all our models, henceforth in this report, we roughly use term *contraction* to imply unification.

So far in the raw base graph, similar keyphrases can be represented by multiple nodes as they may have different surface form or originate from different comments. Therefore, there are two contraction strategies to resolve this issue, as shown in figure 3.6 (red and green nodes indicate nodes originated from two different comment and bold circles denotes nodes that semantically similar):

- **Intra-contraction** contracts similar keyphrases *within* a single comment or text segment. At the same time, it sums up all the properties of these nodes and store these informations in the new node;

- **Inter-contraction** contracts similar keyphrase *within* **and** *between* comments or text segments. Similar to the other approach, the new node take all properties from the nodes to be contracted.

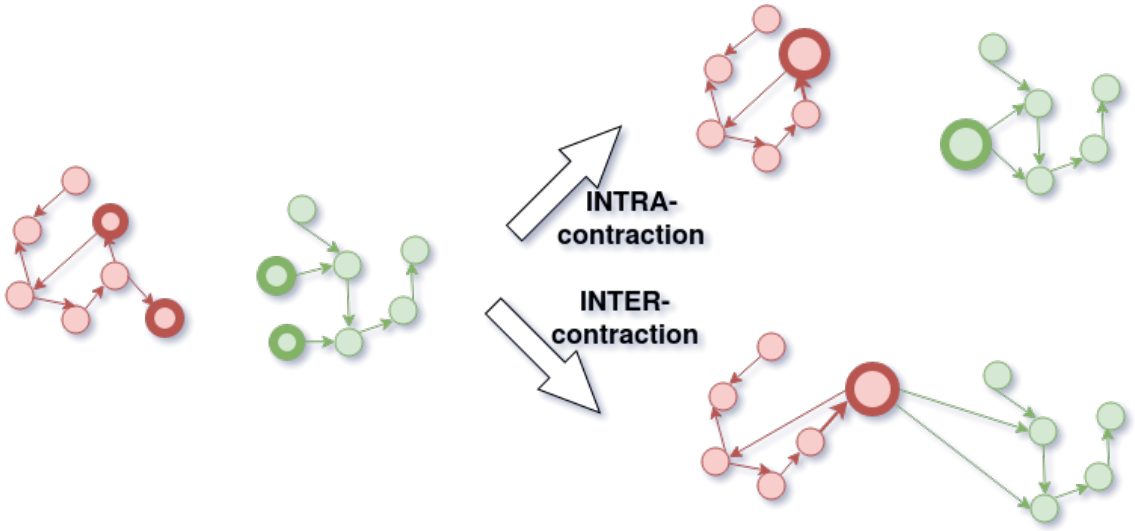


Figure 3.6 Node intra and inter unification

Despite their literal meaning, *inter-contraction* is a global extension of *intra-contraction* which can only locally prevent redundancy within the comment. With intra strategy, nodes are still heavily partitioned an unable to form topics. Inter-contraction, in contrast, garners keyphrase together which can possibly form a topic. However, the drawback for this approach is that it excludes the context (i.e. scoped by the comment) therefore the nodes loose there pragmatic features. The decline of synonyms, homonyms, metaphor, metonymy exemplify the trade-off brought by this approach. For instance, by this step, the verb *play* will be contracted with its metonymy a noun (a story that is written for actors to perform). Our experiment shows that despite its drawbacks, inter-contraction apparently perform better than the other from feasibility and computational perspective.

The unification gives us a tool to reduce the number or semantic similar phrases in graph by grouping them together. At this point, nodes and edges of base graph have already met their definition in AG. Then system now concludes the semantic score of each node according to its positive, negative and neutral frequency. Sentiment score takes value from -1 to $+1$ and reflects the average appeal of participants about all members in the node. The value of -1 indicate absolute negative opinion, while its counter part imply very positive appeal. The

sentiment score of node v_i is computed as follow:

$$sentiment_{v_i} = \frac{posFreq_{v_i} - negFreq_{v_i}}{posFreq_{v_i} + negFreq_{v_i} + neuFreq_{v_i}} \tag{3.4}$$

Where $sentiment_{v_i}$ denotes sentiment score of node v_i, $posFreq_{v_i}$, $negFreq_{v_i}$, $neuFreq_{v_i}$ indicate its positive, negative and neutral frequency, respectively.

Although the base graph is now more clear, it is still somehow noisy due to the existence of infrequent and unimportant phrases. Therefore, in the next step, the system will apply several filters to end up with a complete base graph that facilitates community detection.

3.2.4 Filtering

Filtering aims to eliminate unimportant edges and nodes from the graph. It provides an effective mean to control the size of aspect graph (e.g. number of node, edges). Furthermore, it is useful to avoid noise ensure the graph is well-structured. There are four groups of filters implemented in our system, including general filter, graph-based filter, dependency-based filter and semantic-based filter.

General filters

Comments are noisy, therefore, we need to apply many hard filters. Additionally, the base graph construction process so far applied many modifications to keyphrases, such as merge, lemmatize, join words. To ensure the clearness of base graph, we apply multiple basic filtering step. First, the system check if all keyphrases match a standard regular expression form. Then, we remove short keyphrases who have less than 3 alphabet characters. Abnormal phrases, such as emotions, taunt (e.g. hahaha, lol), etc. will be eliminated from the graph. There is a practical reason why this task has not been carried out in the preprocessing step before build the dependency graph. First and foremost, AG captures sentiment information of conversation therefore, which a proper sentiment analysis method, we can employ emotions, taunt as sentiment indicator.

Graph-based filter

There are a several threshold and options for filtering nodes and edges according to their properties:

- Minimum frequency of the node;
- Minimum frequency of the edge;

- Remove orphan nodes (i.e. node that does not connect to any other nodes).
- Minimum degree of the node. Degree (i.e. valency) of a node is the number of edges incident to it, with loops counted twice. Of course, setting a threshold greater than 0 for minimal degree of nodes entails the removal of orphan nodes.
- Remove rings (i.e. edge that connect a node to itself);

Selecting a good setting for these thresholds is an uneasy task. It mostly depends on size of the conversation and the diversity of keyphrases and aspects in the conversation. Indeed, processing long conversation often require higher value of all these threshold than short one. In our experience, the value of 2 works well for a short conversation with about 100 comments.

Dependency-based filter

This filter focus on the edges between nodes in graph. Previously in algorithm 2, line 4:8, there are a simple step to remove unwanted dependency relation. However, for maintaining the coverage of the construction of dependency graph, this selection step should not be too strict, therefore some unimportant relation remains. Therefore, dependency-based filter strictly removes edges representing unnecessary relations, such as *advcl*(adverbial clause modifier), *parataxis* (discourse-like equivalent of coordination), *det* (determiner is the relation), etc. Dependency relation accommodated from rule-based refinement are excepted for all filters.

Semantic-based filter

In order to ensure that apart from dependency relation, we need to ensure that each edge also connect two nodes that are somehow semantically related. Therefore, we compute cosine similarity for each pair of nodes connected to each other in based graph. Those whose similarity less than a certain threshold will be removed from the graph. In our system, we set this value to 0.15.

Once we have a good base graph, it allows us to detect communities, each corresponds to one aspect of the conversation. Despite a few differences in definition between the term *community* (i.e. a compartment of the graph) and *aspect* (i.e. the insights of a community), for the sake of convenience, we use both of them as interchangeable terms.

3.3 Aspect detection

Our next objective is to point out topics from the base graph constructed in the previous section. With this intention, we employ a method namely community detection. We use a new technique

called Fluid Communities with advantages accuracy, in scalability and diversity. Furthermore, we improve the performance of this algorithm by adding information about importance of nodes to initialization step of the process. In this section, we will first discuss about the original algorithm, following by explanation about our extension for the algorithm. But before coming to description of the method, we will begin with a formulation of community detection problem.

3.3.1 Community detection

A typical graph (i.e. network) consists of nodes and edges that link these nodes together. There are numerous different variations of a graph, but it is found that many of them are inhomogeneous. Instead of consisting of a an undifferentiated mass of nodes, their nodes can be easily grouped into (potentially overlapping) compartments within which nodes connected with sparser connections to other nodes outside. In graph theory and complex networks science, such compartment is called community. Each community, thus, fairly independent therefore can be somehow considered as an autonomous component of the original graph. Scientists have been searching for best way to identify these structures within a graph, especially a complex graph. This task is known as community detection (CD) problem. Indeed, the goal of CD is to detect the modules and, possibly, their organization in hierarchical order, by only exploiting the information encoded in the graph topology[25].

The most basic idea to tackle this problem is so called "cutting the bridges" that deliberately eliminate some edges according to some features until the community structure appears. Other common traditional approaches employ hierarchical clustering and partition clustering in order to join nodes into groups according to their mutual similarity[57]. Modern science has spent many efforts on searching for new methods for solving CD with the assistant of advances in computing technology and Fluid Communities is the most recent method.

If we no turn to what we achieved so far in the pipeline 3.1 which is a large graph consist of keyphrase and their dependency, it is intuitive to realize that our goal (i.e. aspects) is similar to the goal of CD (i.e. communities). Both of them target to look for compartments containing nodes that closely connected to each other and fairly independent in the whole graph. This observation inspires our use of CD to search for aspects in the base graph and we prefer this step as *aspect detection*

3.3.2 Fluid Communities algorithm

Fluid Communities(FluidC) was first proposed by in mid 2017 by Ferran et al.[62]. It is inspired by the iteration between fluids in an environment, along side with their expansion and contraction. The idea of the algorithm is to introduce a number of fluids (each of them

corresponding to a community to be detected) within a non-homogeneous environment (i.e., a non-complete graph). Similar to their natural behaviour, fluids will expand and push each other until reaching stable state. This process depends on the topology of the environment.

The algorithm to find k community in a $G = (V, E)$ can be briefly summarized as follows. First, the algorithm introduces k fluids to k random distinct nodes in the graph. Each community has an associated density bounded in range (0,1]. This follows by the graph expansion consists of many supersteps. At each superstep, it iterates over all nodes in a random order and update the assigned community for each node until convergence or no change was made in two consecutive supersteps. The update of node's community can be imagined as the allocation, rearrangement of fluid as they interact to each other and to the environment. The update is controlled by an update function which basically base on propagation methodology. Despite the simplicity in methodology, it is shown that the algorithm achieves very competitive results compare state-of-the-art. Also, the algorithm is particularly proficient on large and mixing parameters graph. Furthermore, the algorithm is scalable and fast with a linear computational complexity of $\mathcal{O}(E)$. Moreover, it is reported to archive a variable number of communities on a given graph[57].

Despite the fact that FluidC does not clearly outperform the state-of-the-art approaches, as we point our summarization system toward a light-weight scalable method, this algorithm seems to be a good candidate. In addition, putting to the problem of aspect detection gives us more hints to improve the method by taking centrality information to consideration.

3.3.3 Adding centrality information

Although FluidC was demonstrated itself as a prominent candidate, coming to the use case on AG, FluidC exposes one weakness. Practical experiments show that FluidC tends to centralize important communities together to form a big community. In this way, the bias in number of nodes and edges appears toward this big community which obviously leads the aspect detection toward a wrong direction. From the data point of view, the problem can partly address by the fact that there are some dominant aspects in the conversation. Therefore, in the constructed base graph, there are some nodes with high frequency and many connections to its nearby nodes. We can imagine this situation as a surface where a few interconnected big holes occupy the majority of area, while smaller depression scattered in a limited area remains. As we randomly introduce fluid to any location, one fluid will instantly flow to one of the big holes, and since they are interconnected, both two big hole will be filled with the same fluid. The same problem occur when running FluidC in the base graph.

In order to overcome this problem, our solution is to block out the bridge between these big hole to ensure that after initialization step, each hole likely to store a different fluid. This

can be done as following, at the initialization step, we introduce different fluid close to the big holes, so that each fluid can instantly low to its nearest big hole. Once they have arrived at these holes, they push each other and block the bridge between them. In a similar manner, we overcome the problem by anticipating the initialization step of the algorithm. Indeed, instead of randomly introduce k fluids to k random distinct nodes in the graph, we randomly introduce them in top 30% most important node in term of centrality. Our experiments show that this approach allow us to find better aspects with a sufficient set of keyphrase.

To this step, the system results identified communities and help us group there nodes together. This result is equivalent to the result of topic modeling methods, such as Latent Dirichlet Allocation(LDA). However, the difference lies at the representation of aspect(topics). Indeed, LDA represents topics as distributions over all words in the dictionary, and documents as distributions over topics. It comes up with a probabilistic representation of each document as a set of topics. Using *aspect detection*, each aspect is represented as a directed graph over keywords and their relation. In spite of the difference in representation, in theory, it is possible to transform the graph structure to a probabilistic representation for aspects (we will address this issue in section 3.6).

3.4 Aspect labelling

As *aspect detection* draws the borderlines between aspects which consist of a number of nodes, we do not know yet the what each community talks about. Therefore, our next goal is to understand the insight of each detected compartment. With this intention, we assign to each aspect an abstractive label that capture its holistic view according to the nodes and edges that it comprises. This question can be address as an abstraction problem, which take a *graph* of keyphrase as input and result in a single keyphrase the summarical or general idea that all the elements share.

This is an uneasy problem and to the best of our knowledge, there has not been any compelling solution proposed by the research community. Fortunately, a part of this problem was addressed in *cluster labelling* problem. Instead of taking a graph as input, cluster labelling created an abstractive label from a list of keyphrase. For instance, given a list of paraphrases *atmosphere*, *climate*, *climatologist*, *drought*, *ecological disturbance*, *fossil fuel*, *greenhouse effect*, *industrial revolution*, *topography*, *thermal*, a cluster labeller will return keyphrase *global warming*. Of course, it is not easy to perform this task even for human because it requires an external knowledge and subjective opinion. For simplicity, we narrow down our problem (i.e. aspect labelling) to cluster labelling and solve the this problem bases on two approaches: vector-based models and large structured content. But before performing this step, it is notable

that all these cluster labelling methods are very sensitive with the outliers, that is keyphrase that irrelevant to the aspect. Therefore, we apply a semantic filter to those communities that has more than 15 nodes. Indeed, we use one-class Support Vector Machine with parameter ν (upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the total number of training examples) equal to 5.5% with *poly* with 3 dimensions. The remaining labels are now ready for applying two labelling methods.

3.4.1 Labelling with vector-based models

This approach bases on the idea of representing each nodes in the graph as a vector then compose them in to a single vector that allow us to select the best label accordingly. The process consists of 5 steps:

Step 1: Represent the keyphrase (i.e. label) of each nodes in the aspect as a vector in distributed semantic space. We use GloVe[64] to derive a 200-dimension vector for each node. Note that GloVe was originally designed to work at words rather than phrase, therefore we compute this vector by weighted sum of 3 last words of the phrase. The weights for the first words in the sequence (staring from the end of the phrase) are higher than the latter words with the assumption that the head of phrase standing close to the position. For instance, in the keyphrase *beautiful mind*, the word *mind* receives more weight than *beautiful* in composition.

Step 2: Compute single compositional vector using weighted addictive composition over all vectors. Weight is computed as a fraction of the node's frequency over the total frequency of all nodes within the community.

Step 3: Find nearest neighbours of the compositional vector and use their label as the label for aspect.

3.4.2 Labelling with large structured content

For this approach, we use a graph-based topic labelling algorithm introduced by Hulpus et al. [33] base on DBpedia - a manually constructed shallow, cross-domain ontology based on infoboxes in Wikipedia. From a list of keywords: *Atom*, *Energy*, *Electron*, *Quantum*, *Orbit*, *Particle*, the algorithm yield a label such as Theoretical Physics.

The intuition of this method base on concept disambiguation and linking to DBpedia. The starting hypothesis is that the co-occurrence of words in text (in our case, in an aspect) reasonably refers to concepts that belong closely together in the DBpedia graph. The algorithm

employed centrality measures to identify the concepts that best represent the topics of the community. The authors reported that their method gains better results for topic labelling, in terms of both precision and topic coverage, over the standard text-based approaches.

The advantage of this approach over vector-based models is that by nature it support keyphrase, such as *industrial revolution* or even much longer phrases such as *academic ranks in Italy*, and a wide range of proper nouns and named entities, such as *United Nations Development Group* and *United Nations Children's Fund* . However, since it is build from Wikipedia, it mostly contains encyclopediacal concepts rather than general keyphrase, such as *academic career* or *prominent methodology*, which is, unfortunately, frequently used in conversation. This critical drawback leaves many words outside the labelling process, therefore ended up with fairly poor result. In some cases, the method may return an empty result since none of these keyphrases was found in the ontology.

Aspect labelling itself is a sophisticated task and require further consideration. So far, the outcome of both methods are limited at encouraging results. In the following step, we will finalize the process by doing concluding sentiment information for each aspect, and decorate the graph for visualization purpose.

3.5 Finalizing and Visualization

With the complete of aspect labelling, the base graph now own most essential feature of an AG: nodes that hold semantic similar keyphrases, edges indicate potential relation, community structure representing aspects and a label for each community that capture its holistic insights. However, the sentiment layer is still uncompleted due to the fact that the aspect does not have their general sentiment information. In this section, we will explain how sentiment score for each community is computed. Moreover, for the purpose of visualizing aspect graph, we color nodes and communities by their sentiment before displaying in a web-based interactive visualization.

3.5.1 Determine importance of aspects

The importance of an aspect is determined by its frequency and user's opinion over its corresponding community. While the frequency can be computed in a straight-forward way by summing up the frequency of all nodes of the community, this approach does not suit sentiment score because of two reasons. First, sentiment score should bound in range $[-1, 1]$ where -1 indicates very negative opinion and 1 show the opposite appeal. Second, within a community, nodes are different in their importance where influential nodes attribute more to the overall

sentiment than the other ones. Therefore, we compute the sentiment score of a community as the weighted average of the sentiment score of its nodes, where weights are computed upon nodes' frequency. This is a relatively simple, yet, effective approach. However, the trade-off is that it does not take the relation between nodes (edges) to consideration.

3.5.2 Decorating and exporting graph description

As we reach this step, the AG construction is almost finished. The following steps are merely aims to visualization purpose. Indeed, we convert AG to a visual graph in which nodes and edges is corresponding to nodes and edges of the AG. This visual graph also follows exact graph structure as the constructed AG. The only difference is that properties of nodes and clusters (i.e. frequency, sentiment score) is explicitly visualized by its size and color. Frequented node has larger size than rare ones, and color is scale from red (indicate very negative opinion) to green (very positive opinion). We finally export the result a file which will be used in generating online visualization of the graph.

3.5.3 Online interactive visualization

Back to our intention to propose a method for summarizing online conversation, it is also plausible to deploy the whole system or at least display the outcomes on a website. Therefore, we built a web application for visualizing the AG. The website is available at http://cassandra.disi.unitn.it/sm_summarizat
It provides an interactive tool for inspecting the AG accompany with basic modification actions. Details will be described in appendix B.

3.6 Predictive AG model

Whilst an AG reveals topics underlying a conversation, a predictive model derived from AG allows us to predict the topic of a fresh text document among those detected by the AG algorithm. Let $G = (V,E)$ be an aspect graph where V is the set of nodes representing keyphrases and E is a set of directed edges representing potential relation between two nodes. We Let $C = \{c_1, c_2, .., c_n\}$ is a set of communities detected in G where n is the total number of communities. Each community c_i forms a subgraph that has set of member nodes $m_i = \{v_{c_i}^1, v_{c_i}^2, ..., v_{c_i}^m\}$ where m is the total number of members in community i. The prediction matrix A is define as:

$$A = (s_{ij}.cen(v_{c_i}^j)) \in \mathbb{R}^{n \times |V|} \tag{3.5}$$

Where $s_{ij} = 1$ if word $v_j \in m_i$, 0 otherwise and $cen(v^i_{c_j}))$ is the centrality score of node j in community c_i if it is in community m_i. Briefly say, each row of the prediction matrix carries the centrality score of all nodes in the corresponding community if the node is inside the community and 0 otherwise.

We will examine 4 methods for measuring centrality of a node:

- Degree(cen_{deg}): number of edges that incident to a node.
- PageRank(cen_{pr}): importance of a node based on number of important neighbors it has [61].
- Eigenvector centrality(cen_{eig}): centrality of a node based on the centrality of its neighbors[16]. Eigenvector centrality of a node i is:

$$\mathbb{A}x = \lambda x \tag{3.6}$$

  Where $\mathbb{A}$ is the adjacency matrix of the graph G with eigenvalue λx. There can be many eigenvector generally, however Perron–Frobenius theorem implies that the the desired centrality measure can only be the result of the greatest eigenvalue[56].

- Closeness(cen_{cl}): reciprocal of average shortest path distance to the node over all $n-1$ reachable nodes[26], where n is the number of nodes that can reach the node:

$$cen_{cl} = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)} \tag{3.7}$$

  Where $d(v,u)$ is the shortest-path distance between node v and u.

The model aims to predict the probability that an unseen document d. Let $\overrightarrow{\delta}$ be its bag-of-word representation using the set of nodes V as vocabulary. The probability that d can be labeled as the topic represented by cluster k is:

$$P(d_{label} = k|\delta, A) = \sigma(v^T a_k) \tag{3.8}$$

Where a_k is row k of prediction matrix and σ is normalization function. There are two normalization methods are examined:

- Sum function:

$$\sigma_{sum}(z)_j = \frac{z_j}{\sum_{h=1}^{H} z_h} \tag{3.9}$$

  for $j = 1, ..., H$

- Softmax function:

$$\sigma_{softmax}(z)_j = \frac{e^{z_j}}{\sum_{h=1}^{H} e^{z_h}} \tag{3.10}$$

  for $j = 1, ..., H$

Where z is a a H-dimensional vector (in our case, $H = |V|$).

Finally, the label of the unseen document is computed as follow:

$$d_{label} = \underset{c_i \in C}{\operatorname{argmax}} P(d_{label} = c_i | \delta, A) \tag{3.11}$$

Chapter 4

Experiments and results

In the previous chapter, we discussed about two experiments used for evaluating our definition and implementation for constructing AG from online conversation. However, this construction method is also applicable to single or multiple text document as well with the assistance of text segmentation methods. Evaluating summarization system is a challenging task because even human annotators may derive different summaries from the same document. Especially in this work, we gave the definition of our method produces summarical representation of the conversation in stead of an actual summary. A good AG needs to satisfy two requirements: being able to capture most informative keyphrases, and correctly group detected keyphrases into aspects (i.e. sub-topics) can be clearly recognized. Analogically, this suggests that the method can be evaluated from two perspectives: keyphrase extraction and topic modeling method.

4.1 Keyphrase extraction

Similar to human behavior of focusing on informative words and phrase while reading text documents [44], the goal of keyphrase extraction is to automatically choose important and topical phrases from the body of a document [82]. It is a crucial component for most of text summarization systems today which concerns on providing a abridged version of a text by distilling its most important information for a particular task and user [49]. According to Santosh *et al.* in [13], there are 3 common approach for keywords extraction as shown in figure 4.1. We consider keyphrase extraction as a subtask of aspect detection, which is intuitively understandable from the topic modelling perspective. In the illustrated classification, AG is a hybrid approach by employing simple statistic, linguistic and graph theory.

Although keyphrase extraction is a generic step that involves to range of tasks in natural language processing, most of the researches was built for rich text document, such as paper

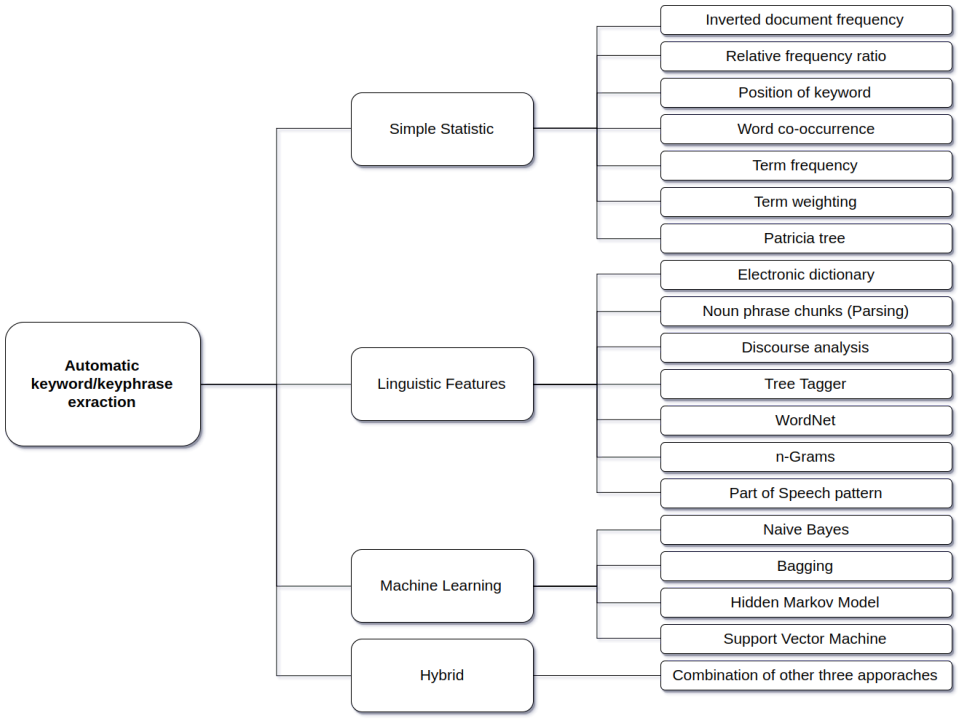


Figure 4.1 Classification of common keyword extraction approaches.

abstracts [34], scientific paper [58, 50], news articles [35, 79, 45]... There are a few research on conversation-like corpus such as meeting transcripts [28] and live chats[40], however neither the datasets are public nor keyphrase extraction method was explained.

We benchmark our method with three commonly state-of-the-art methods for keyphrase extraction:

- TextRank [51] with the implementation of Nathan [54]. This is a famous graph-based unsupervised methods for extracting keyword and sentence. The extraction process begins with tokenization, part-of-speech tagging, n-grams filtering. To this step, single and multi-word keywords serve as candidates for the graph and edges are added by their co-occur within a window of N words. Then, they perform an algorithm on the given graph to determine the importance of each node, then select most important keyword according to this score. This method is different from our method in selecting candidates and edges. Also, we use centrality score for ranking nodes, instead of merely based on assigned score as TextRank.

- TopicRank [18] implemented in pke - an open source python-based keyphrase extraction toolkit [17]. The method was introduced in 2013 by Adrien et al. Unlike TextRank, it targets extracting keyphrases, instead of keyword although both of them used graph-based

approach. First, the algorithm enlists candidates by performing sentence segmentation, word tokenization and part-of-speech tagging. These candidates are stemmed and group into topics with Hierarchical Agglomerative Clustering (HAC) algorithm. Then a graph is constructed where nodes are candidates and edges are the semantic relation defined as they appear close to each other in the document. Then, they employ the ranking method from TextRank to rank nodes and similarly end up with a raking list of keyphrase by its importance. TopicRank is often considered as a "keyphrase-based" extension of TextRank. It's also important to know that TopicRank output single words and phrases after reducing the inflected word forms into root forms by a stemmer.

- Rapid Automatic Keyword Extraction (RAKE) [68] with the implementation named RAKE-NLTK by Fabian von Feilitzsch [84]. RAKE is a keyword extraction algorithm that maximize the efficiency on not only single but also collection of documents. The technique in RAKE is relatively simple. First, it establishes a list of candidates as tokens and phrases detected from the document. Then, a co-occurrence graph is built, and nodes are ranked by to the ratio between frequency and degree. Finally, the algorithm adjoins keywords that appear at least twice in one document, which turns the outcome to keyphrases. By its name, the method is extremely fast while operating in large amount of text in various type.

All these methods accompany with their implementation have been widely use by the research community. The common theme of all these approach ins that they are graph-based approaches merely constructed bases on word co-occurrence. Notwithstanding, there has been no report about their performance in short document and conversational text. The dataset they used for evaluation, unfortunately, have not been publicly release. Therefore, we benchmark these methods with ours in conversation text. Indeed, as mentioned in *Construct Aspect Graph*, by the end of graph pruning phase (before community detection), we have a big graph that represent the conversation whose nodes are keyphrases and edges are dependency relations. All the phrase before this acts similar to candidate selection step of TextRank, TopicRank and RAKE. With the given graph, we define important keyphrases as those who connected to many others nodes. In another word, a keyphrase is important as many other keyphrases depend on it (in term of dependency relation). In term of graph theory, this fits to the function of centrality indices. There are several centrality indices, and in this experiment, we examined eigenvector, degree, closeness and Pagerank score. We rank keyphrases by centrality score and select 20 highest score keyphrases from comparison with other methods. The results are shown in table 4.1, 4.2 and 4.3.

4.1.1 Scenarios

To understand how AG works on news article (i.e. the content of the article) and comment, we consider 3 scenarios:

- Article only: perform AG on the text of the article, which is mostly plain and well-written text, following journalistic Writing. Standard news article usually follow as theme called Inverted Pyramid structure (i.e. summary news lead style, bottom line up front)[65]. It can be thought as a horizontally flipped pyramid indicating the priority of information in each story. The widest part at the top represents the most newsworthy information (about who, what, when, where, why how), the middle level provide other important information, while the tapering lower portion mention other material, general of back ground information order of diminishing importance. The article itself contains a remarkable amount of information which is in most cases organized in paragraphs. Therefore, instead of processing its content in a whole, we first segmentize it with texttiling algorithm [30], then perform AG construction to each segment.

- Comment only: with this scenario, we aim to target the real conversation. The idea is that we first extract keyphrase graph for each comment, then compose these extracted graph to form a single graph that represents the whole conversation. Keyphrases (represented as nodes) are picked from this big graph and rank by its importance, indicated by centrality score.

- Combination: this scenario takes into consideration the role of article in triggering the conversation. This can be done by first extract a base AG for article content using the techniques mentioned above, then it comes to the turn of AG for comments then we compose both of two graphs to form a full representation of the conversation and article. Similar to comment only case, we determine and rank keyphrases according to nodes' centrality score.

Among these three scenarios, only the second one address a typical conversation summarization problem where we don't have information about the primary topic beforehand.

4.1.2 Dataset

Most experiments in this thesis was carried out with SENSEI annotated corpus[10]. The corpus consists content and reader comments of 18 online news articles from The Guardian online newspaper[1]. They are numbered 0-5, 9-12, 15-17, 19, 21-23 and 25. Among them, 15 articles

[1] https://www.theguardian.com

were double annotated and 3 articles were triple annotated. Every article comprise the title (i.e. headline) and comments. In this experiment, we use article 23 and their comments as an example. This article content and title consist of 687 words while there are 100 comments from 48 commenters with totally 3,485 words.

4.1.3 Results

Table 4.1, 4.2 and 4.3 show top 20 keyphrases extracted by RAKE, TextRank, TopicRank and our method, respectively.

- **Phrase completeness** refer to the identical semantic meaning of the candidate. RAKE appeared to establish phrases with more words than the others. Unfortunately, these phrases are relatively noisy with redundancy words. This problem is even much worse when coming to the case of comments, therefore are many unexpected characters in the phrases, such as . . . , *!!!!!*. TextRank, TopicRank and AG, fortunately do not have this problem as their lists are very clean in all three table. However, AG id also made error with some unnecessary words, such as *th century* (original word was *20th centuries*), *half*

- **Redundancy errors** occur when the method identifies a candidate correctly as a keyphrase, but at the same time it also recognizes semantically equivalent candidates (e.g., its inflected form, alias) as a keyphrase. In another word, the method can not determine that two candidates are semantically equivalent. Redundancy happened to TextRank when it was unable to spot word inflection, thus, keyphrase *australian* and *australians*, *issue* and *issues* both appeared in its list. TopicRank and AS overcame this problem by using stemmer and lemmatizer.

- **Infrequency errors** are the most critical error of keyphrase extraction method when it is unable to identify a candidate as keyphrase due to its infrequent occurrence in the document[29]. As the topic is *Researchers tackle link between climate change and public health*, it is understandable that *climate change* is one of the most important keyphrase. However, it is absent in the keyphrase list of RAKE and TextRank. TopicRank successfully recognized this word, but it didn't recognize *public health* as a keyphrase as people mostly mentioned *health* instead. Fortunately, all AG-based methods leveraged the problem correctly. *Language related error* occurs due to the special characteristic of conversation language. How this error affect can be seen by comparing the performance of each method between table 4.1 and 4.2. Besides phrase completeness error previously mentioned, some frequent candidates in spoken language such as *likely*, *long time*

appeared in the keyphrase list, despite their limited importance. This *frequent error* strongly affected RAKE and TextRank, with less effect to TopicRank and AG.

The result, as shown in table 4.1, 4.2, 4.3, indicate that AG outperform other methods in the pilot experiment. AG was able to deal with phrase completeness, diminish redundancy, infrequent and language-related error. However, it is also important to realize that every method tried to optimize their system for a specific purpose. Although in our experiment, RAKE appeared to be the lest effective method, it was developed with emphasis on speed and capability to operate on large amount of text. Similarly, TextRank was designed as a simple method for pre-processing method. TopicRank and AG output a very competitive result with many overlap keyphrase because they are both designed with the idea of capturing topics with detected keyphrase. Additionally, it is worth mentioning that in the result, we only display the representative keyphrase for each node, which in fact is a set of semantically equivalent keyphrases.

Table 4.1 Top 20 keyphrases extracted from **the article** by Aspect graph with 5 different node centrality metrics against RAKE, TextRank and TopicRank

#	RAKE	TextRank	TopicRank	Aspect Graph				
				Eigenvector	**Degree**	**Closeness**	**Betweenness**	**Pagerank**
1	year world bank president jim yong kim warned	public health	climat chang	food	health impact	public health	food	food
2	said monash university climate scientist dr ailie gallant	professor	health	public health	food	food	health impact	certain disease
3	food security climate change may threaten australians livelihoods	researchers	extrem weather event	certain disease	certain disease	certain disease	certain disease	public health
4	recommendations included improving early warning systems	research	public health respons	health impact	climate change	extreme event	climate change	health impact
5	group spent two days analysing	science	health impact	water supply	average temperature	water supply	average temperature	water supply
6	tackled issues including agricultural productivity	said	water suppli	extreme event	public health	health impact	school	extreme event
7	groups examining five key areas	warned	research	climate change	group	average temperature	monash university	climate change
8	science brought together 60 early	warning	food	average temperature	school	australian	environmental science	average temperature
9	armstrong said climate change	complex	public health australian academi	australian	monash university	climate change	population increase	public health researcher
10	said professor sharon friel	social	earli recommend	public health researcher	extreme event	environmental science	australian	monash university
11	climate change social inequalities	food	infecti diseas	monash university	environmental science	public health researcher	group	australian
12	emeritus professor bruce armstrong	issue	temperatur	environmental science	academy	monash university	public health researcher	school
13	relatively small scientific contribution	issues	small increas	school	australian	school	academy	environmental science
14	science brings experts together	australian	univers	population increase	public health researcher	population increase	extreme event	population increase
15	better engaging disadvantaged communities	australians	govern	academy	population increase	group	bruce armstrong	academy
16	influence public health responses	publicly	scienc bring expert	bruce armstrong	water supply	academy	water supply	group
17	monash universitys school	climate	disadvantag	group	extreme weather	bruce armstrong	public health	bruce armstrong
18	making early recommendations	scientifically	social instabl	extreme weather	bruce armstrong	extreme weather	extreme weather	extreme weather
19	public health australian academy	temperatures	conflict	challenge	challenge	challenge	challenge	challenge
20	physical science issues	temperature	group					

Table 4.2 Top 20 keyphrases list extracted from **the comments** by Aspect graph with 5 different node centrality metrics against RAKE, TextRank and TopicRank

#	RAKE	TextRank	TopicRank	Aspect Graph				
				Eigenvector	**Degree**	**Closeness**	**Betweenness**	**Pagerank**
1	zero scientific credibility ... australian news dated july 12th 2014 oh look	climate	cours climat chang	hot climate	climate change	hot climate	hot climate	hot climate
2	warming denialist culture well done interesting ...	climates	year	climate change	warm	ocean current	climate change	climate change
3	begin digging roman style cisterns instead	warming	temperatur	cool	hot climate	climate change	warm	cool
4	reviewed research authors rejects global warming	warm	poor scienc	wind system	cool	wind system	cool	metabolic heat
5	000 peer reviewed scientific papers published	warms	last ice	ignorant unread climate denier	imprison people	sea ice	warm period	warm
6	liesbig liesstatisticsfossil fuel climate science could	blockquote	global warm	ocean current	warm period	southern	radiation phase	ignorant unread climate denier
7	one big fat lie ... well	like	planet	sea ice	radiation phase	long time	positive side	wind system
8	new obesity miracle cure !!!!!	likely	lie	long time	ocean current	cool	ocean current	sea ice
9	less vehicles means less emissions	changes	real scientist	percentage point	peak	percentage point	peak	ocean current
10	hypocritically nauseous neoliberal ideology really	change	20th centuri averag	major impact	degrees mean	excess	excess	st century
11	fossil fuel companies keep lying	ice	heat	warm period	continent	ignorant unread climate denier	warming trend	radiation phase
12	soviet like science came forth	years	antarct sea ice	interglacial period end	half	metabolic heat	major impact	easy way
13	power failures would become life	year	peopl	radiation phase	st century	major impact	interglacial period end	southern
14	seems science deniers always seek	global	australia	warm	deal	warm	imprison people	excess
15	principal scientific research organisations	temperature	noth	solar cycle	major impact	interglacial period end	billions home	model
16	spinning worldwide crop failures	temperatures	time	southern	excess	country	continent	long time
17	recommend reading greg hunt	heat	last	country	sea ice	warm period	poor science	drought proof
18	represents narrow corpratz power	period	period	excess	solar cycle	solar cycle	solar cycle	solar cycle
19	despite tony robots attack	periods	seriou problem	peak	ignorant unread climate denier	peak	ignorant unread climate denier	ice age
20	yet another excellent article	periodical	denialist	warming trend	long time	warming trend	long time	percentage point

Table 4.3 Top 20 keyphrases list extracted from **both article and comments** by Aspect graph with 5 different node centrality metrics against RAKE, TextRank and TopicRank

#	RAKE	TextRank	TopicRank	Aspect Graph				
				Eigenvector	**Degree**	**Closeness**	**Betweenness**	**Pagerank**
1	zero scientific credibility ... australian news dated july 12th 2014 oh look	climates	climat chang	climate warming	climate change	public health	climate change	climate change
2	year world bank president jim yong kim warned	warm	year	public health	poor science	certain disease	climate warming	climate warming
3	said monash university climate scientist dr ailie gallant	warms	temperatur	certain disease	climate warming	cataclysmic event	health impact	health impact
4	food security climate change may threaten australians livelihoods	climate change	scienc	climate change	warm	health impact	food	food
5	begin digging roman style cisterns instead	effects global warming	last ice	food	health impact	climate change	cool	certain disease
6	recommendations included improving early warning systems	changing	global warm	health impact	global temperature	global temperature	certain disease	global temperature
7	warming denialist culture well done interesting ...	changes	planet	cool	high school	food	warm period	move
8	reviewed research authors rejects global warming	changed	lie	cataclysmic event	public health	ocean current	warm	public health
9	liesbig liesstatisticsfossil fuel climate science could	year	20th centuri averag	ignorant unread climate denier	certain disease	move	global temperature	cataclysmic event
10	000 peer reviewed scientific papers published	years	real scientist	ocean current	cataclysmic event	long time	cataclysmic event	cool
11	one big fat lie ... well	researchers	guardian australia	long time	food	climate warming	excess	poor science
12	new obesity miracle cure !!!!!	research	extrem weather event	global temperature	warm period	natural environment	ocean current	ignorant unread climate denier
13	tackled issues including agricultural productivity	blockquote	peopl	natural environment	cool	cool	plan	metabolic heat
14	group spent two days analysing	likely	antarct sea ice	warm period	deal	ignorant unread climate denier	imprison people	long time
15	less vehicles means less emissions	like	heat	move	imprison people	warm period	population increase	ocean current
16	hypocritically nauseous neoliberal ideology really	ice	complex problem	warm	move	excess	evidence	warm
17	groups examining five key areas	scientifically	govern	solar cycle	academy	metabolic heat	poor science	excess
18	fossil fuel companies keep lying	science brings	public health respons	radiation phase	planet	warm	radiation phase	drought proof
19	science brought together 60 early	said	last	excess	long time	radiation phase	warming trend	th century
20	soviet like science came forth	scientists	health	warming trend	daytime heat extreme	poor science	high school	warm period

4.2 Topic modeling

Aspect discovery in some senses is similar to topic modeling task, because they both try to discover topics from the text. Therefore, we can evaluate AG as a topic modeling method who seeks for topics, that is a cluster of words (or keyphrase) frequently co-occurred [6]. In our case, the existence of these keyphrases and their clusters can intuitively be seen as nodes and communities, respectively. Similar to the case of keyphrase extraction, there are not many topic modeling methods particularly built conversation yet. Most of the notable researches paid attention on online conversations, such as Twitter conversation[67, 90, 7]. A relevant research conducted by Enamul Hoque [31] on modeling topic for comment threads following news article.

4.2.1 Procedure

In this research, we evaluate the performance of AG in comment clustering and comment classification task. Indeed, we previously proposed a predictive model base on AG which labels an unseen document according to the aspect (i.e. sub-topics) detected while building the aspect graph. This model allows us to performs two sub-experiments:

- Comment topic clustering: we perform hard clustering on the comments by their topic.
- Comment classification: this can also be called comment to topic linking part.

The general idea is to construct topic model for the first 70% expressions in the conversation, then use the model to cluster fresh data. At the same time, this model can be used for classifying fresh comment to the modelled topics. The both two sub-experiments are carried out as follow:

- Step 1: Divide the dataset (comments of each article) into training set (70%) and testing set. In multiple document experiment, training data is drawn randomly from the dataset, however in the experiment with conversation dataset, we select first 70% comments from the conversation. This is due to the fact that conversation's expressions appear in a timeline therefore it is a time-ordered list of text.
- Step 2: train the model, as shown in 4.5. We first build the aspect graph from textual content of the conversation without awareness about the golden label or the identity of the text. This is a typical topic modeling task, followed by the construction of predictive AG model from the AG (details will be described in the following section). Then we use the model to predict the label for each document from the training dataset. By this point, the test set is clustered by topics, each of them comprises documents assigned the same

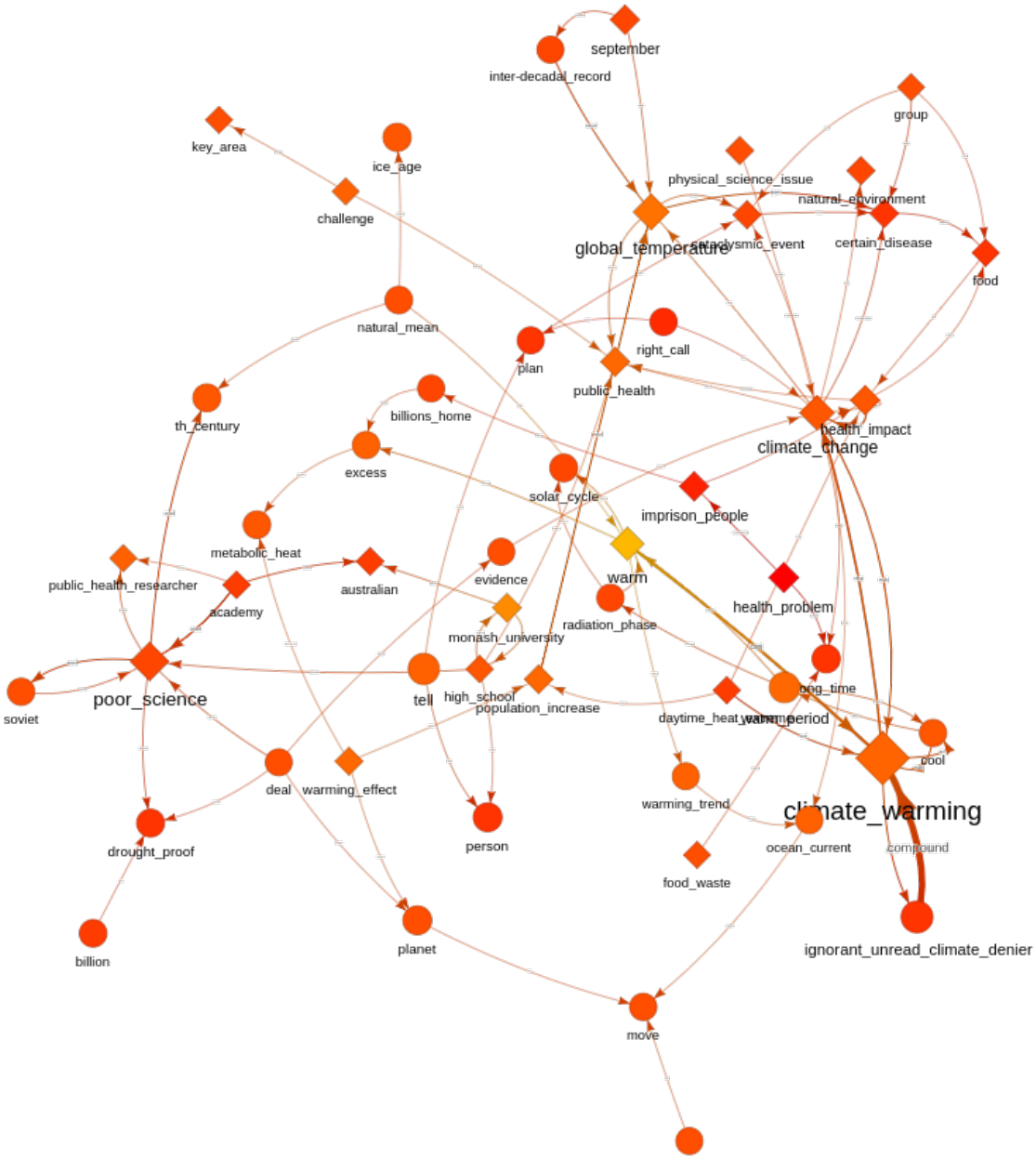
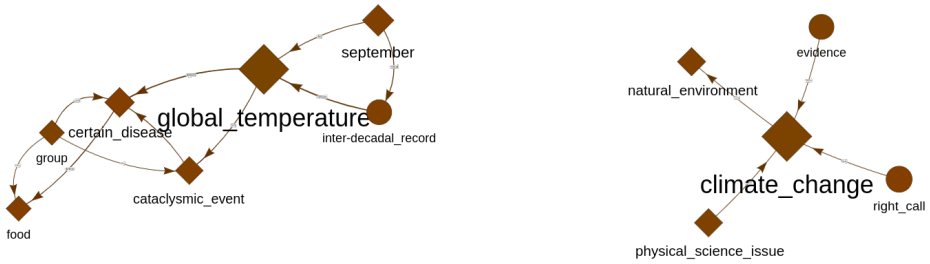
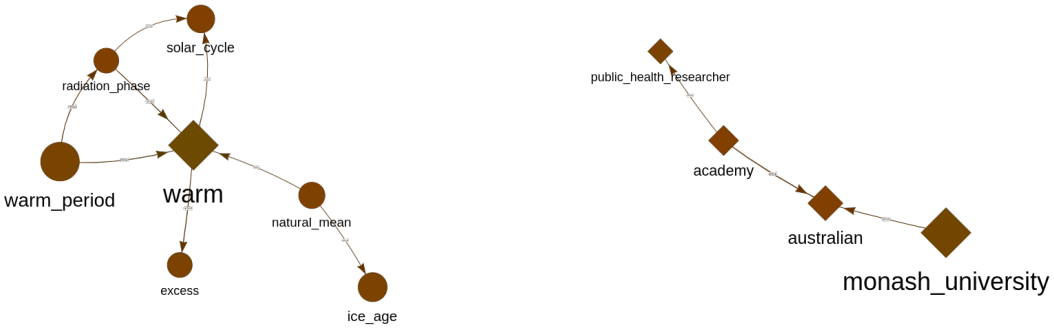


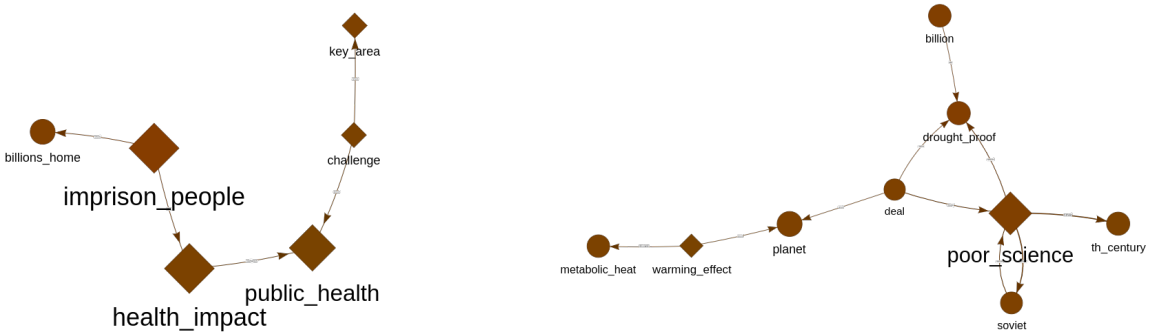
Figure 4.2 A minimized AG constructed from article 23 in combined scenario



(a) Cluster 1, 2



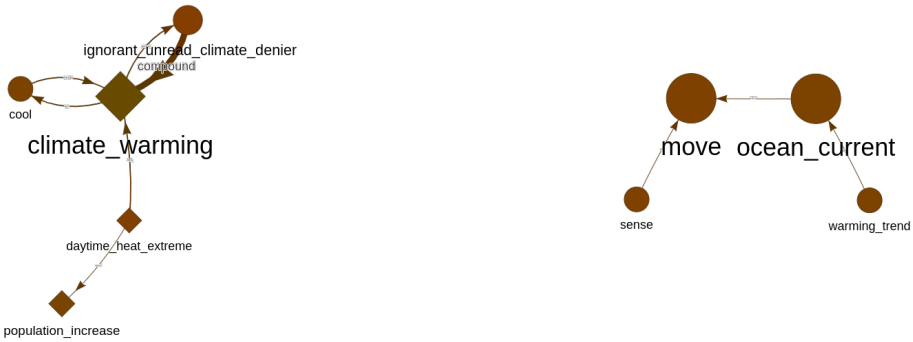
(b) Cluster 3, 4



(c) Cluster 5, 6

Figure 4.3 Visualization of clusters 1 - 6 constructed from article 23 in combined scenario

(a) Cluster 7, 8



(b) Cluster 9, 10

Figure 4.4 Visualization of clusters 7 - 10 constructed from article 23 in combined scenario

label. Similarly, the golden labels can be thought to cluster documents by their topic. We should now compare the predicted clusters and golden clusters. However, all the predicted labels are the result of community labels abstraction, therefore we need to map them to the ground truth labels. Intuitively, the best match for a ground-truth label among the predicted labels is the one that has highest number of documents in common.[71, 43]. In general, the best map between a list of predicted labels to the list of golden labels is the one that maximizes the total overlap between pairs of predicted and golden clusters, that is argmax $\sum_{c_i \in L_g, c_j \in Lp} \left| c_i \cap c_j \right|$ where L_g is the set of clusters derived from golden labels and L_p is the set of clusters derived from predicted labels. Once the predicted labels are mapped to their corresponding golden labels, the algorithm will update the model's labels to be golden labels.

- Step 3: Evaluation: predict the test set and compute score, as shown in figure 4.6. By this step, the system assign a label to each comment indicating its topic. We can now evaluate the result from clustering and classification point of views with various metrics

4.2.2 Evaluation metrics

Comment topic clustering

We compute the result of topic clustering with LDA, LDA with Gibbs sampling and LSI on 4 standard clustering metrics:

- Adjusted rand score (bounded range from -1 to 1) is the corrected-for-chance version of the Rand index[83]. The Rand index is a measure of the similarity between two data clusterings. Negative values indicate bad clustering outcome (independent labelings), similar clusterings have a positive adjusted rand index, 1.0 is the perfect match score.

- Silhouette score (bounded range from -1 to 1) is a method of interpretation and validation of consistency within clusters, showing how well each comment lies within its cluster[69]. Value -1 means incorrect clustering, 1 indicates highly dense clustering and zero indicates clusters have many overlap.

- Completeness score (bounded range from 0 to 1) approaches 1 when most of the comment that are members of a given cluster are elements of the same cluster. In another word, it indicate how well all comments of a given topic is assigned to the same cluster. Zero value means all clusters are incomplete.

- Adjusted mutual info score (bounded range from 0 to 1). The closer this value to 0 shows that the label assignments are largely independent from the ground truth. In contrast,

values close to one show significant agreement. Further, when all predicted and golden clusters purely independent, this score get value 0, and it get value 1 when two label assignments method are equal (with or without permutation).

- Homogeneity score (bounded range from 0 to 1). It indicates how well each cluster contains only members of a single topic. Value 0 indicates bad clustering result, when 1 is the perfect outcome.
- V measure score (bounded range from 0 to 1). V-Measure is defined as the harmonic mean of homogeneity and completeness of the clustering. The closer the score to 1 means better clustering result
- Fowlkes mallows score (bound range from 0 to 1). It is an external evaluation method that is used to determine the similarity between two clusterings,based on error analysis with *True Positive*, *True Negative*, *False Positive*, *False Negative* count (see more at Comment classification metrics). A higher value for this core indicates a greater similarity.

For all these scores, the higher value means better clustering result.

Comment classification

For each cluster (i.e. aspect/subtopic) c_l contains documents assigned label l, we use four scores to measure the effectiveness of the model while making prediction: accuracy, precision, recall and F1-score [59] in both micro and macro computation methods:

$$Mic.avgPrecision = \frac{\sum_{t \in T} TP_i}{\sum_t TP_t + \sum_t FP_t} \tag{4.1}$$

$$Mac.avgPrecision = \frac{\sum_{t \in T} P_i}{|T|} \tag{4.2}$$

$$Mic.avgRecall = \frac{\sum_{t \in T} TP_i}{\sum_t TP_t + \sum_t FN_t} \tag{4.3}$$

$$Mac.avgRecall = \frac{\sum_{t \in T} R_i}{|T|} \tag{4.4}$$

$$Mic.avgF1 = \frac{2 \times Mic.avgPrecision \times Mic.avgRecall}{Mic.avgPrecision + Mic.avgRecall} \tag{4.5}$$

$$Mac.avgF1 = \frac{2 \times Mac.avgPrecision \times Mac.avgRecall}{Mac.avgPrecision + Mac.avgRecall} \tag{4.6}$$

$$Avg.Accuracy = \frac{\sum_{t \in T} TP_i + TN_i}{\sum_{t \in T} TP_i + TN_i + FP_i + FN_i} \tag{4.7}$$

Table 4.4 Dataset for conversation topic modelling

Article No.	Title	No. words in article	No. of comments	Annotations
1	Police access to medical records will not help the vulnerable	633	101	Annotator 1: 15 aspects Annotator 2: 10 aspects
11	WHO declares Ebola outbreak an international public health emergency	921	106	Annotator 3: 15 aspects Annotator 9: 13 aspects
23	Researchers tackle link between climate change and public health	687	100	Annotator 8: 13 aspects Annotator 9: 10 aspects Annotator 11: 6 aspects

Where T is set of all topics), TP (true positive) is number documents whose labels were correctly predicted as l, TN (true negative) is number documents whose labels were correctly predicted as **not** l, FP (false positive) is number documents whose labels were **incorrectly** predicted as l (i.e. false alarm, or type I error), FN (false positive) is number documents whose labels were **incorrectly** predicted as **not** l (i.e. missed, or type II error). P_i is the simple precisions of prediction on topic i, $P_i = \frac{TP_i}{TP_i+FP_i}$. R_i is the simple recall of predictions on topic i, $P_i = \frac{TP_i}{TP_i+FN_i}$. In addition, besides micro and macro, we also compute weighted average of precision, recall and F1 between all topics. The average value is weighted by support (i.e. the number of true instances for each topic).

4.2.3 Dataset

We use annotated conversations from SENSEI annotated corpus[10]. Among 18 articles, we select article 23, 11 and 1 because the conversations have identical topics which were captured correctly by the annotators. These articles are selected based on the quality of annotation. Indeed, SENSEI annotating process consist of three stages. First of all, participants are asked to write a *label* for each comment. These labels are short, free text annotation, capturing its essential content. Second, annotator grouped together labels (from Stage 1) which were similar or related, then provided a label for each group. The label is expected to describe the common theme of the group in terms of. Finally, they write a summary for all these groups. Since the whole annotation process is subjective, not all annotators perform the task correctly as instructed. Some annotator marks all comments as positive, negative and miscellaneous. Others over-categorized the comments into too small topics, consequently, some topics a few

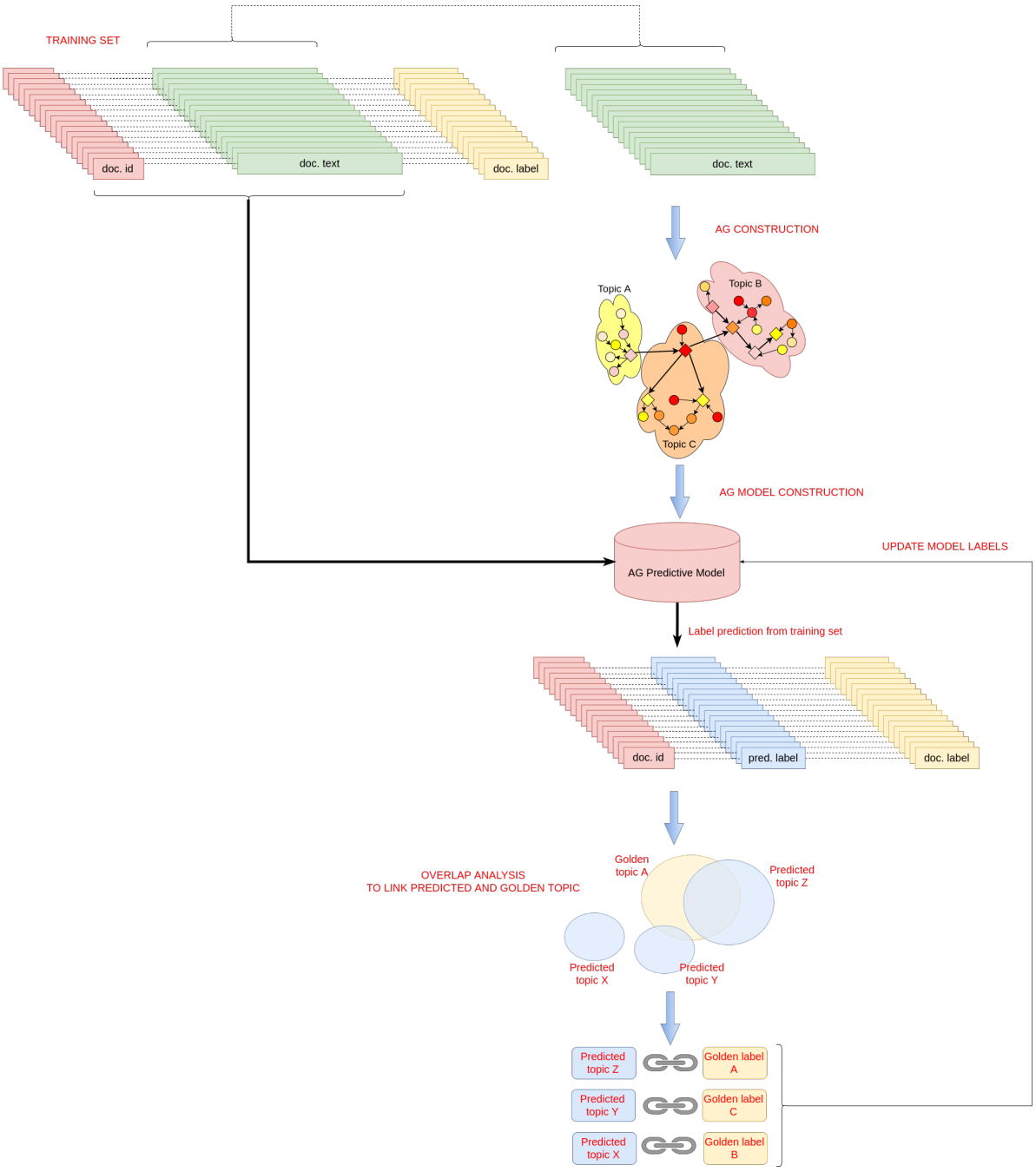


Figure 4.5 Train a AG predictive model.

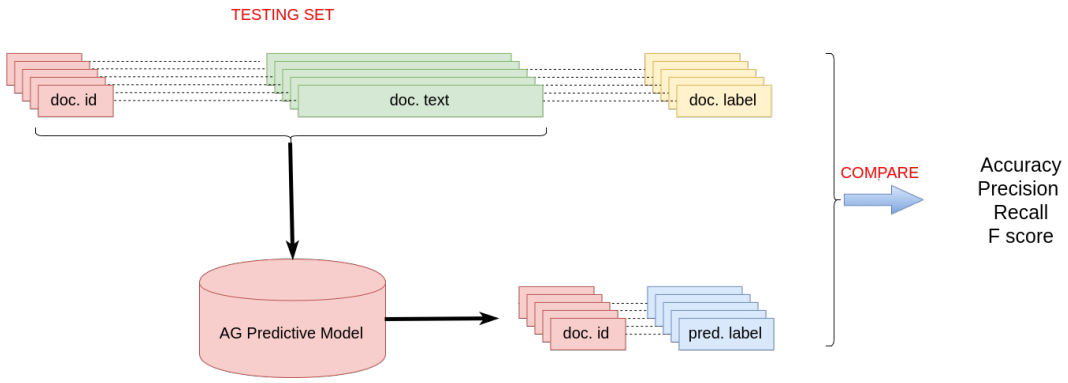


Figure 4.6 Test AG predictive model.

comments, while the other has too many comments. Technically speaking, this problem cause a situation called imbalanced classes who is a problematic issue for machine learning. Of course, We don't want our system to learn this bias, therefore we conduct our experiments on well-annotated articles only. We use group labels assigned in the second stage as ground-truth for our classification and clustering.

Additionally, since each conversation was annotated by at least 2 annotators, we conduct in total 6 runs, whose number of topics are shown in table 4.4. Each run corresponds to one annotator over one conversation, which is split to training set (70% first comments) and testing set. The topic for a comment is the closest group or subgroup that it belongs to according to the annotator. Besides, due to the fact that annotator subjectively decides the number of topics, we force the algorithm according to the annotator's choice in order to generate comparable keyphrase and document clusters. Table 4.4 show information about three article, their number of comment, number of word in the content and number of golden standard provided by annotators.

4.2.4 Results

Comment clustering results

Table A.3 compares the clustering performance of our method against LDA, LDA with Gibbs sampling and LSI. The first 8 rows are the results from 8 derivatives of AG predictive model, that use four different centrality measurement and two normalization method. It can be either dividing by sum (denoted as +) or applying softmax function (denoted as ^ symbol). As show in the table, AG models in general provide a better clustering result compare to other candidate. The average value of Adjusted rand, Silhouette, Completeness, Adjusted mutual info and Fowlkes mallows score of clusters generated by all AG models are 0.0505, 0.0507, 0.3852, 0.0419 and 0.0507 which is much higher than other candidates. Only the case of Homogeneity

Table 4.5 Average clustering scores comment clustering on all 3 articles (7 annotators)

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score
*Pagerank^*	0.0615	0.0628	0.4090	0.0601	0.4259	0.4141	0.0628
Pagerank+	0.0504	0.0504	0.3939	0.0445	0.4048	0.3976	0.0504
*Eigenvector^*	0.0513	0.0513	0.4010	0.0441	0.4169	0.4044	0.0513
Eigenvector+	0.0203	0.0203	0.3413	0.0057	0.3066	0.3181	0.0203
*Degree^*	0.0630	0.0630	0.3984	0.0575	0.3858	0.3904	0.0630
Degree+	0.0485	0.0485	0.3850	0.0375	0.3981	0.3891	0.0485
*Closeness^*	**0.0768**	**0.0768**	**0.4193**	**0.0710**	0.4293	0.4166	**0.0768**
Closeness+	0.0323	0.0323	0.3334	0.0150	0.3141	0.3176	0.0323
LDA	0.0315	0.0315	0.3975	0.0516	**0.4506**	**0.4193**	0.0315
LDA w/ Gibbs	0.0049	0.0049	0.3691	0.0177	0.4030	0.3819	0.0049
LSI	0.0192	0.0192	0.3961	0.0256	0.3186	0.3415	0.0192

score and V measure score where LDA perform better than ours, which indicates that LDA was ale to produce clusters contains only members of a single topic which can be interpreted as high precision. Therefore, V measure score, which is the harmonic measurement of completeness and homogeneity of LDA is higher than ours, although our homogeneity score is slightly better.

It is apparent from the table that compare to other candidate, AG models have advantages by generating clusters that similar to ground-truth (indicated by high Adjusted rand and Fowlkes Mallows score) and dense cluster indicated by Silhouette score) and a high recall (indicated by completeness score). However, the boundary of AG models generated cluster is too big that it may produce result with low precision. Another explanation for this result lies on the fact that in AG, every topic is a distinctive set of keyphrase (i.e. there are no intersection between keyphrases set of any two topics). LDA, on the contrary, learn to assign a weight to every keyword in the dictionary that may associated to the topic. Therefore, the number of indicative keywords for each topic in AG is much less than LDA, which mean that AG models have a higher chance to end up with probability 0 for all topics while considering a fresh sample. By default, AG models assign this sample the label "unknown", while LDA is less lightly to face to this problem. Obviously, the larger number of samples that belong to "unknown" cluster means the lower precision AG will be. Nevertheless, despite this weak point of AG, its archived value of V score in the model using closeness centrality and softmax normalization is only 0.64% less than LDA.

Among all 8 models of AG, the use of node closeness as centrality metric and softmax as normalization method consistently outperforms other set-ups. The interesting point here is that closeness, as computed as the reciprocal of the sum of the shortest path distances from a node to all other nodes, took edges, which are the frequency of the potential relations to consideration. Other metrics that didn't use this information, such as degree or PageRank did not perform well in clustering task. This raise the further observation that frequency relations between keyphrases seems to as useful as the keyphrase itself, at least in clustering circumstances.

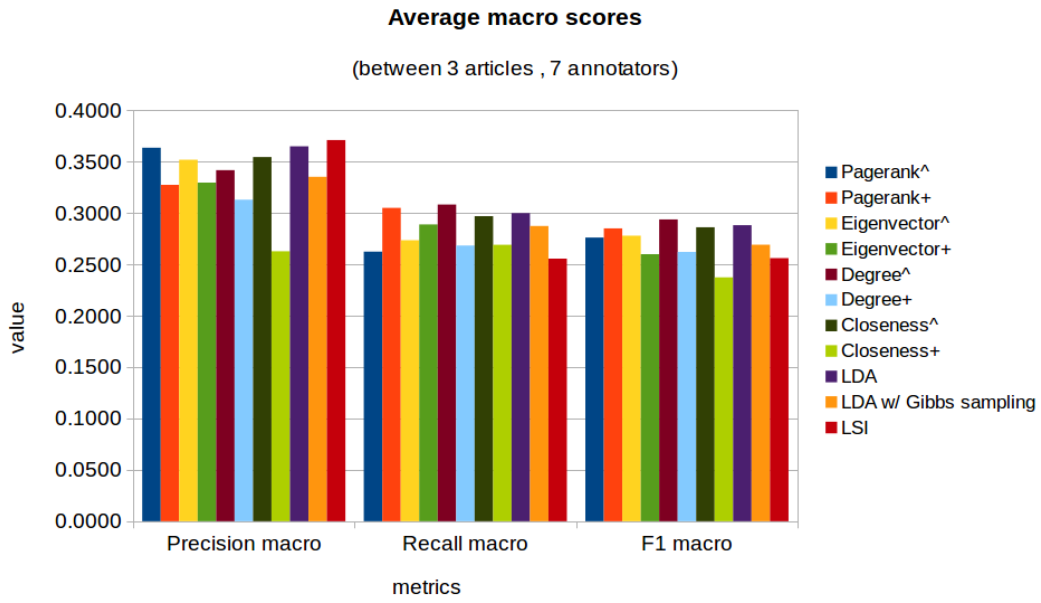


Figure 4.7 Average macro scores for 6 articles (7 annotators)

Besides, we can also infer from the table that using softmax for normalization produced a better clustering result than merely normalize centrality by sum.

Comment classification results

Figure 4.7, 4.8 and 4.9 report the micro, macro and weighted score for AG models, versus LDA, LDA with Gibbs sampling and LSI. In general, AG has a very competitive result compared to is opponents in macro score, whilst outperforming them in micro and weighted score. Figure 4.7 gives results of the performance of AG in classification task under average macro score. LSI archived the highest average precision micro score of 37.09% while the highest result of AG reported at Pagerank+ with 36.33%. However, these two method do not result in high macro recall average, there fore their recall score is relatively small. The harmonic metric of F1 shows the best score for degree^at 29.36%, only 0.55% better than the runner-up LDA. Apparently, thess straight-forward metrics didn't clearly unveil the difference in outcome of AG and other candidate.

Coming to the case of average micro score demonstrated in 4.8, AG clearly revealed its superiority over the three opponents. Indeed, for all scores precision, recall and F1, closeness^and degree^archived the value of 41.20% and 41.11%. F1 score for LDA, LSI and LDA with Gibbs sampling is at lowest among all benchmarked methods, which is 36.65%, 35.66%

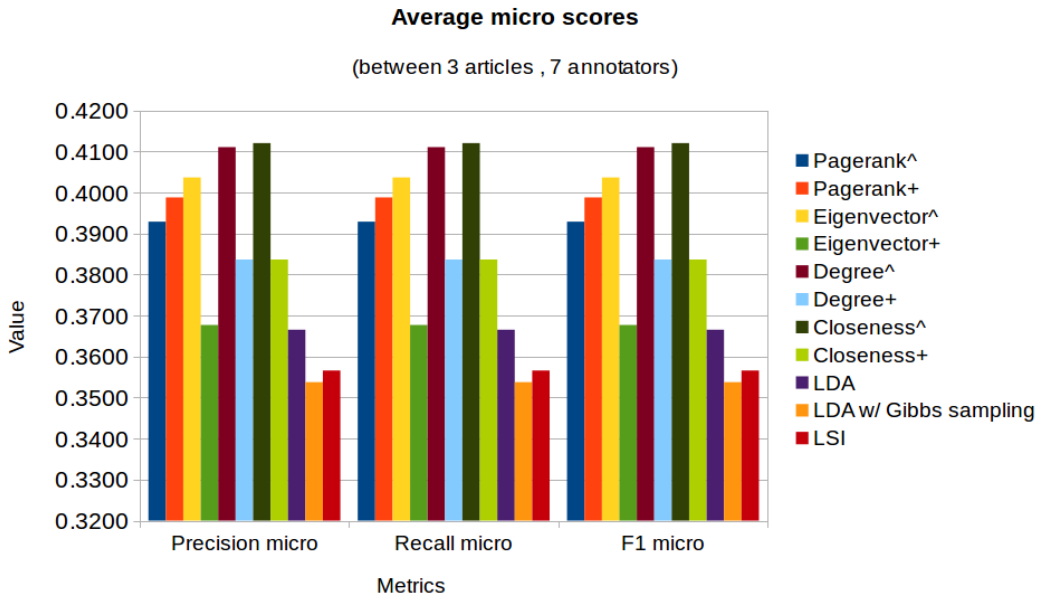


Figure 4.8 Average micro scores for 6 articles (7 annotators)

and 35.37%. Similar to the result from comment clustering, closeness+ gained the highest performance among all AG models and in this case, it is also better than LDA.

Figure 4.9 confirm the successful of AG in general and Closeness+ in specific over other candidates. Under this average computation method, all methods resulted in lower recall than precision. Despite the advantage of LDA while approaching 57.10% recall, its runner-up in this measure archived 41.20% recall, which is 4.55% better than LDA recall. As a result, F1 score of LDA, the highest among three candidates, only reaches 39.18% wheras this number of closeness^is 42.50%.

The similar conclusion can also be seen in average accuracy score demonstrated in figure 4.10. Best accuracies are reported for closeness^(41.20%) and degree^(41.11%) while LDA, LDA with Gibbs sampling and LSI only come up with 36.65%, 35.37% and 35.66%, respectively.

The results also show the difference in performance between AG models. Closeness^constantly show as the best model in both comment clustering and comment classification evaluation. At the second position, surprisingly, degree^also give positive results. Between tow normalization methods, softmax function highlighted it is prominence over scaling by sum method. Beside, eigenvector reported to be the weakest centrality measurement metric. This can be explained by its concept, that is connections to an important nodes contribute more to the score of the node in question than equal connections to less important nodes. This method is useful for graph traversing, however from the syntactic point of view, it allows rare keyphrase gained

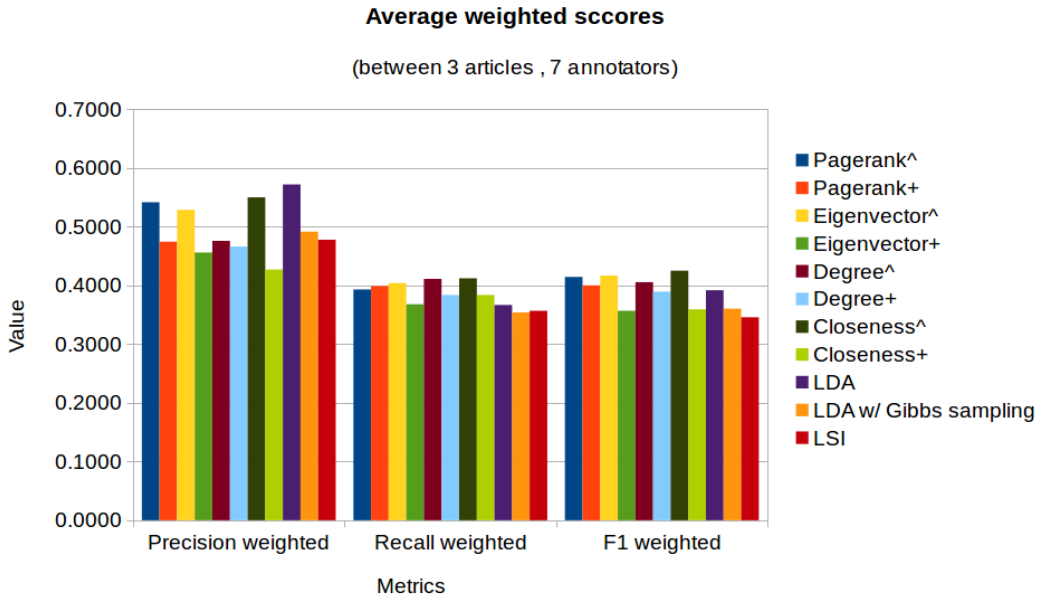


Figure 4.9 Average weighted scores for 6 articles (7 annotators)

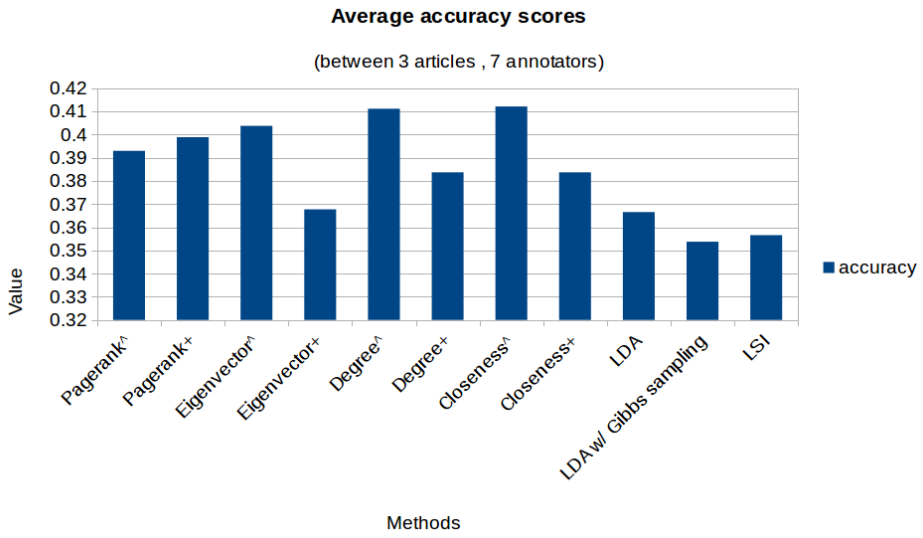


Figure 4.10 Average accuracy scores for 6 article (7 annotators)

high important score, as long as it is connected to some high-score node and result in low topic modeling result. Eigenvector^appeared to result in the most modest result, as it is the combination of two less effective method in measuring centrality and normalization.

Chapter 5

Conclusion and discussion

5.1 Conclusion

Originating from the need for an unsupervised abstractive summarization method for online conversation, we have present Aspect Graph as an effective intermediate representation that capture summarical information of the conversation. This structure supports later efforts in generation of an abstractive summary which is outside the scope of this study. We further proposed a method for constructing AG was benchmarked the performance with state-of-the-art methodologies in solving two problems: keyphrase extraction and topic modeling task on online comments. The keyword extraction results show that our method works better than state-of-the-art systems in term of phrase completeness, redundancy and infrequency error avoidance. In topic modeling task, AG does not clearly outperform the current state-of-the-art in terms of cluster quality. However, its classification result for linking comment to topics surpass state-of-the-art in precision, recall an accuracy. Furthermore, we implemented an online web visualization that allow the user to display an interact with graph in convenience.

Our research highlighted the importance of sentiment analysis in conversation summarization. Additionally, to the best of our knowledge, our work is the first study on incorporating the advantage of graph theory and sentiment analysis in solving text summarization problem. It also provides a blueprint for new abstractive conversation summarization method. Moreover, it is possible to extend current approaches in summarization by substituting only the conventional intermediate representation by AG without breaking down the whole system. This proved the wide range of application brought by the proposed method.

However, given the small sample size, caution must be taken on the scalability and the capability to tackle diversity of aspects. Also, the involvement of many natural language processing many refrains efforts in reproducing the method.

5.2 Discussion

In general, the reported result have so far consistently revealed that AG modes are better than AG outperformed other candidates in conversation data. This result is consistent to the finding of research [5, 46] that LDA does not work well for online conversation. This is a surprising result since LDA has been well known as a state-of-the-art for topic modeling fore more than a decade. If we now turn to explain this observation, the first possible reason is the special characteristic of online conversation as explained in *Introduction*. Secondly, the LDA was built for standard documents that was structured in the paragraph. Online conversation, on the contrary, consist of short expression which is usually closed to spoken language. Another distinctive characteristic about the structure of conversation is that it follows a timeline, on which new idea can spark out and drive a part of the conversation.

Besides linguistic and structural distinction of online conversation that refrains the performance of LDA, our method comprised many techniques that successfully enhanced the result. First, while LDA used bag-of-word model, we employed syntactic dependency to make the relationship between words more meaningful, rather than merely capture co-occurrence. This technique clearly helpful in reducing unnecessary words, which is a well-known problem in language of conversation. Furthermore, beside capturing a list of keywords and their commutation into topics, AG also address the potential relation between them, which is useful for text generation and constructing a larger semantic graph. Secondly, we incorporated semantic information to ensure clusters are semantically consistent. Third and most import, in AG, we consider keyphrase not only as a single unique of meaning, but also the whole history since the beginning of the conversation. In this manner, every node in the graph can be considered as a set of keyphrases that convey the same information throughout the conversation. For instance, keyphrases like *Fiat*, *Ford*, *Toyota* appeared at the beginning of the conversation, but then, people mention about *car companies*. AG was designed to group these words to be a single set, with *car companies* as representative and *Fiat*, *Ford*, *Toyota* are history of keyphrase *car companies*. This organization is an summarical approach for capture summarical information from the text while avoiding redundancy and infrequent words error.

There is a doubt in the importance of the topic (given by the article) while setting up conservation summarization system. Many research, such as [48] take the article as an essential input for the system. However, their results revealed that topics in the conversation went far beyond the boundary of article's topics. Henceforth, there is a question whether the prior understanding about the main topic is useful for starting the summarization or, on the contrary restrict the system's ability to detect fresh topics. Although so far there are no clear answer for this question although many recent researches [46, 5] are turning to to pure conversation summarization and disregard the role of an apriori knowledge about the original topic. In our

work, we considered both two scenarios, although it is easy to realize that not all conversation staring with a clear main topic. An example of this can be taken from chat conversation where topics change as the conversation go on. An alternative cases are conversations that triggered from video, music or image where main topic can't be directly taken from these content.

5.3 Future works

Further works will concentrate on investigating a better use of relations on aspect detection and aspect labelling. Indeed, current implementation of FluidC with refinement in initialization step does not take direction of edge into account. In addition, aspect labelling phase narrowed down the problem to cluster labelling which is only a part of problem. Obviously, this approach wastes the information about relation between nodes, therefore the it is importance to overcome this shortfall in future work.

Also, much attention was paid toward the importance of nodes in the graph. On the other hand, far too little use of edges has been discussed. However, edges and the relation they carry are markedly importance from various perspective. In the view of summarization purpose, some summarization process focus on summary the relation, action between the participants or aspects rather than just point out these aspects of keyphrases. For example, from a conversation about climate, the following sentence may appeared in the summary: *the majority blamed burning fossil fuel for causing global warming* or *most comments are criticism*. Here in this example, these sentence summarized about relations rather than keyphrases. More particularly, the keyphrase *criticism* may not necessaries originated from the conversation, but being the *general word* for a set of relations found in the graph, such as *criticize*, *disagree*, etc. Besides, some relations can be considered as more important than the other, especially the relations that learned from the conversation. All these question require further studies to be undertaken.

Finally, as our ultimate goal is a complete unsupervised summarization system for online conversation, we intend to develop a text generating component. Indeed, the component will take aspect graph as input and result in a complete summary.

Bibliography

[fb2] 100 social media statistics for 2012. http://thesocialskinny.com/100-social-media-statistics-for-2012/. Accessed: 2017-11-18.

[2] Data never sleeps 5.0. https://www.domo.com/blog/data-never-sleeps-5/. Accessed: 2017-11-18.

[3] Things that happens on internet every 60 seconds. https://www.go-globe.com/blog/things-that-happen-every-60-seconds/. Accessed: 2017-11-18.

[4] Abuobieda, A., Salim, N., Eltayeb, R., bin Wahlan, M. S., Suanmali, L., and Hamza, A. (2011). Pseudo genetic and probablisitic-based feature selection method for extractive single document summarization. *Journal of Theoretical and Applied Information Technology*.

[5] Aker, A., Kurtic, E., Balamurali, A., Paramita, M., Barker, E., Hepple, M., and Gaizauskas, R. (2016). A graph-based approach to topic clustering for online comments to news. In *European Conference on Information Retrieval*, pages 15–29. Springer.

[6] Alghamdi, R. and Alfalqi, K. (2015). A survey of topic modeling in text mining. *I. J. ACSA*, 6(1):147–153.

[7] Alvarez-Melis, D. and Saveski, M. (2016). Topic modeling in twitter: Aggregating tweets by conversations. *ICWSM*, 2016:519–522.

[8] Angraini, Y. (2017). Sentiment analysis of sarcasm in spoken language. *ANGLO-SAXON: Jurnal Ilmiah Program Studi Pendidikan Bahasa Inggris*, 8(1):103–114.

[9] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[10] Barker, E., Paramita, M. L., Aker, A., Kurtic, E., Hepple, M., and Gaizauskas, R. (2016). The sensei annotated corpus: Human summaries of reader comment conversations in on-line news. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pages 42–52. Association for Computational Linguistics.

[11] Baron, N. S. (2010). *Always on: Language in an online and mobile world*. Oxford University Press.

[12] Berg-Kirkpatrick, T., Gillick, D., and Klein, D. (2011). Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

[13] Bharti, S. K. and Babu, K. S. (2017). Automatic keyword extraction for text summarization: A survey. *arXiv preprint arXiv:1704.03242*.

[14] Binwahlan, M. S., Salim, N., and Suanmali, L. (2009). Swarm based features selection for text summarization. *IJCSNS International Journal of Computer Science and Network Security*, 9(1):175–179.

[15] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[16] Bonacich, P. (1987). Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182.

[17] Boudin, F. (2016). pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan. The COLING 2016 Organizing Committee.

[18] Bougouin, A., Boudin, F., and Daille, B. (2013). Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 543–551.

[19] Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

[20] Cukier, K. and Mayer-Schoenberger, V. (2013). The rise of big data: How it's changing the way we think about the world. *Foreign Aff.*, 92:28.

[21] Demchenko, Y., De Laat, C., and Membrey, P. (2014). Defining architecture components of the big data ecosystem. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, pages 104–112. IEEE.

[22] Donath, J. (2002). A semantic approach to visualizing online conversations. *Communications of the ACM*, 45(4):45–49.

[23] Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

[24] Fattah, M. A. and Ren, F. (2009). Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Computer Speech & Language*, 23(1):126–144.

[25] Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3):75–174.

[26] Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

[27] Gambhir, M. and Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

[28] Gillick, D., Favre, B., Hakkani-Tür, D., Bohnet, B., Liu, Y., and Xie, S. (2009). The icsi/utd summarization system at tac 2009. In *TAC*.

[29] Hasan, K. S. and Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *ACL (1)*, pages 1262–1273.

[30] Hearst, M. A. (1997). Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

[31] Hoque, E. and Carenini, G. (2016). Interactive topic modeling for exploring asynchronous online conversations: Design and evaluation of convisit. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(1):7.

[32] Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

[33] Hulpus, I., Hayes, C., Karnstedt, M., and Greene, D. (2013). Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 465–474. ACM.

[34] Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics.

[35] Hulth, A. and Megyesi, B. B. (2006). A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 537–544. Association for Computational Linguistics.

[36] Ji, S., Vishwanathan, S., Satish, N., Anderson, M. J., and Dubey, P. (2015). Blackout: Speeding up recurrent neural network language models with very large vocabularies. *arXiv preprint arXiv:1511.06909*.

[37] Jin, X., Wah, B. W., Cheng, X., and Wang, Y. (2015). Significance and challenges of big data research. *Big Data Research*, 2(2):59–64.

[Jobson and Gutiérrez] Jobson, E. and Gutiérrez, A. Abstractive text summarization using attentive sequence-to-sequence rnns.

[39] Joshi, M. and Rosé, C. P. (2007). Using transactivity in conversation for summarization of educational dialogue. In *Workshop on Speech and Language Technology in Education*.

[40] Kim, S. N. and Baldwin, T. (2012). Extracting keywords from multi-party live chats. In *PACLIC*, pages 199–208.

[41] Kroeger, P. R. (2005). *Analyzing grammar: An introduction*.

[42] Lee, S., Belkasim, S., and Zhang, Y. (2013). Multi-document text summarization using topic model and fuzzy logic. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 159–168. Springer.

[43] Li, Z., Wang, B., Li, M., and Ma, W.-Y. (2005). A probabilistic model for retrospective news event detection. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 106–113. ACM.

[44] Liu, Z. (2005). Reading behavior in the digital environment: Changes in reading behavior over the past ten years. *Journal of documentation*, 61(6):700–712.

[45] Liu, Z., Huang, W., Zheng, Y., and Sun, M. (2010). Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 366–376. Association for Computational Linguistics.

[46] Llewellyn, C., Grover, C., and Oberlander, J. (2014). Summarizing newspaper comments. In *ICWSM*.

[47] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[48] Ma, Z., Sun, A., Yuan, Q., and Cong, G. (2012). Topic-driven reader comments summarization. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 265–274. ACM.

[49] Mani, I. and Maybury, M. T. (1999). *Advances in automatic text summarization*, volume 293. MIT Press.

[50] Medelyan, O., Frank, E., and Witten, I. H. (2009). Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1318–1327. Association for Computational Linguistics.

[51] Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

[52] Murray, G. and Carenini, G. (2008). Summarizing spoken and written conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 773–782. Association for Computational Linguistics.

[53] Nagwani, N. (2015). Summarizing large text collection using topic modeling and clustering based on mapreduce framework. *Journal of Big Data*, 2(1):6.

[54] Nathan, P. (2016). Pytextrank, a python implementation of textrank for text document nlp parsing and summarization. https://github.com/ceteri/pytextrank/.

[55] Nenkova, A. (2005). Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *AAAI*, volume 5, pages 1436–1441.

[56] Newman, M. (2010). *Networks: an introduction*. Oxford university press.

[57] Newman, M. E. (2004). Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330.

[58] Nguyen, T. and Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 317–326.

[59] Olson, D. L. and Delen, D. (2008). *Advanced data mining techniques*. Springer Science & Business Media.

[60] Ouyang, Y., Li, W., Li, S., and Lu, Q. (2011). Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47(2):227–237.

[61] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[62] Parés, F., Gasulla, D. G., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U., and Suzumura, T. (2017). Fluid communities: A competitive, scalable and diverse community detection algorithm. In *International Workshop on Complex Networks and their Applications*, pages 229–240. Springer.

[63] Pasunuru, R., Guo, H., and Bansal, M. (2017). Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 27–32.

[64] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[65] Po¨ ttker, H. (2003). News and its communicative quality: the inverted pyramid—when and why did it appear? *Journalism Studies*, 4(4):501–511.

[66] Radev, D. R., Hovy, E., and McKeown, K. (2002). Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408.

[67] Ritter, A., Cherry, C., and Dolan, B. (2010). Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics.

[68] Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

[69] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

[70] Sarkar, K. (2010). Syntactic trimming of extracted sentences for improving extractive multi-document summarization. *J. Comput*, 2:177–184.

[71] Sayyadi, H. and Raschid, L. (2013). A graph analytical approach for topic detection. *ACM Transactions on Internet Technology (TOIT)*, 13(2):4.

[72] Schuster, S. and Manning, C. D. (2016). Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*.

[73] Song, W., Choi, L. C., Park, S. C., and Ding, X. F. (2011). Fuzzy evolutionary optimization modeling and its applications to unsupervised categorization and extractive summarization. *Expert Systems with Applications*, 38(8):9112–9121.

[74] Sood, A., Mohamed, T. P., and Varma, V. (2013). Topic-focused summarization of chat conversations. In *European Conference on Information Retrieval*, pages 800–803. Springer.

[75] Stepanov, E., Favre, B., Alam, F., Chowdhury, S., Singla, K., Trione, J., Béchet, F., and Riccardi, G. (2015). Automatic summarization of call-center conversations. In *In Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU 2015)*.

[76] Suzuki, J. and Nagata, M. (2017). Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 291–297.

[77] Tan, J., Wan, X., and Xiao, J. (2017). From neural sentence summarization to headline generation: A coarse-to-fine approach.

[78] Tao, Y., Zhou, S., Lam, W., and Guan, J. (2008). Towards more effective text summarization based on textual association networks. In *Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on*, pages 235–240. IEEE.

[79] Thomas, J. R., Bharti, S. K., and Babu, K. S. (2016). Automatic keyword extraction for text summarization in e-newspapers. In *Proceedings of the International Conference on Informatics and Analytics*, page 86. ACM.

[80] Torres-Moreno, J.-M. (2014). *Automatic text summarization*. John Wiley & Sons.

[81] Tur, G. and De Mori, R. (2011). *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

[82] Turney, P. D. (2000). Learning algorithms for keyphrase extraction. *Information retrieval*, 2(4):303–336.

[83] Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854.

[84] von Feilitzsch, F. (2016). python-rake, a python module implementation of the rapid automatic keyword extraction (rake). https://github.com/fabianvf/python-rake.

[85] Walter, E. (2008). *Cambridge Advanced Learner's Dictionary Hardback with CD-ROM for Windows and Mac Klett Edition*. Ernst Klett Sprachen.

[86] Wamba, S. F., Akter, S., Edwards, A., Chopin, G., and Gnanzou, D. (2015). How 'big data'can make big impact: Findings from a systematic review and a longitudinal case study. *International Journal of Production Economics*, 165:234–246.

[87] Wan, X. (2008). Using only cross-document relationships for both generic and topic-focused multi-document summarizations. *Information Retrieval*, 11(1):25–49.

[88] Wang, D., Zhu, S., Li, T., Chi, Y., and Gong, Y. (2011). Integrating document clustering and multidocument summarization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(3):14.

[89] Wang, D., Zhu, S., Li, T., and Gong, Y. (2009). Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300. Association for Computational Linguistics.

[90] Xu, Z., Ru, L., Xiang, L., and Yang, Q. (2011). Discovering user interest on twitter with a modified author-topic model. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 422–429. IEEE Computer Society.

[91] Zhu, X. and Penn, G. (2006). Summarization of spontaneous conversations. In *Ninth International Conference on Spoken Language Processing*.

Appendix A

Topic modelling error and cluster analysis results

Table A.1 Average scores for topic modelling clustering and analysis on Article 23 (3 annotators)

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	accuracy
*Pagerank^*	0.0453	0.0453	0.4144	0.0421	0.3387	0.3710	0.0453	0.3463	0.4000	0.4297	0.3235	0.4000	0.4000	0.3099	0.4000	0.3762	0.4000
Pagerank+	0.0462	0.0462	0.4192	0.0383	0.3908	0.4029	0.0462	**0.4081**	**0.4333**	0.5413	**0.3632**	**0.4333**	**0.4333**	0.3459	0.4333	**0.4349**	**0.4333**
*Eigenvector^*	**0.0540**	**0.0540**	**0.4241**	0.0532	0.3433	0.3759	**0.0540**	0.3878	**0.4333**	0.5028	0.3201	**0.4333**	**0.4333**	0.3213	0.4333	0.4240	**0.4333**
Eigenvector+	0.0353	0.0353	0.3454	0.0112	0.2331	0.2741	0.0353	0.2655	0.3889	0.3702	0.3178	0.3889	0.3889	0.2640	0.3889	0.3532	0.3889
*Degree^*	0.0449	0.0449	0.4103	0.0470	0.3562	0.3800	0.0449	0.3555	0.4222	0.4458	0.3478	0.4222	0.4222	0.3257	0.4222	0.4003	0.4222
Degree+	0.0287	0.0287	0.3874	0.0172	0.3313	0.3557	0.0287	0.2621	0.3889	0.3904	0.2904	0.3889	0.3889	0.2551	0.3889	0.3624	0.3889
*Closeness^*	0.0186	0.0186	0.3951	0.0120	0.2775	0.3206	0.0186	0.3148	0.3778	0.4482	0.3271	0.3778	0.3778	0.2685	0.3778	0.3436	0.3778
Closeness+	0.0265	0.0265	0.3492	0.0156	0.2544	0.2931	0.0265	0.2419	0.3889	0.3249	0.3025	0.3889	0.3889	0.2504	0.3889	0.3284	0.3889
LDA	0.0393	0.0393	0.4129	**0.0646**	**0.4233**	0.4153	0.0393	0.3639	0.4000	**0.5481**	0.3628	0.4000	0.4000	0.3238	0.4000	0.4204	0.4000
LDA w/ Gibbs	0.0442	0.0442	0.4176	0.0525	0.4199	**0.4172**	0.0442	0.2824	0.3778	0.4147	0.2745	0.3778	0.3778	0.2619	0.3778	0.3749	0.3778
LSI	0.0379	0.0379	0.4162	0.0553	0.3729	0.3904	0.0379	0.3300	0.3556	0.3961	0.3076	0.3556	0.3556	0.2871	0.3556	0.3453	0.3556

Table A.2 Average scores for topic modelling clustering and analysis on Article 11 (2 annotators)

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.0144	0.0144	0.3739	0.0034	0.4320	0.4008	0.0144	0.3808	0.3594	0.5852	0.1985	0.3594	0.3594	0.2402	0.3594	**0.4085**	0.3594
Pagerank+	-0.0040	-0.0040	0.3224	-0.0544	0.3749	0.3463	-0.0040	0.2687	0.3438	0.4260	0.2256	0.3438	0.3438	0.2301	0.3438	0.3657	0.3438
*Eigenvector^*	0.0044	0.0044	0.3531	-0.0114	0.3886	0.3699	0.0044	0.3764	**0.3906**	0.5595	0.2408	**0.3906**	**0.3906**	0.2645	**0.3906**	0.4231	**0.3906**
Eigenvector+	-0.0076	-0.0076	0.3014	-0.0406	0.3115	0.3058	-0.0076	**0.4064**	0.3594	0.5047	**0.3021**	0.3594	0.3594	**0.2901**	0.3594	0.3815	0.3594
*Degree^*	0.0228	0.0228	0.3626	-0.0045	0.3802	0.3705	0.0228	0.3135	0.3594	0.4753	0.2125	0.3594	0.3594	0.2284	0.3594	0.3748	0.3594
Degree+	0.0265	0.0265	0.3469	-0.0168	0.4148	0.3775	0.0265	0.3439	0.3750	0.5020	0.2668	0.3750	0.3750	0.2820	0.3750	0.4101	0.3750
*Closeness^*	**0.0421**	**0.0421**	0.3878	**0.0606**	0.4130	0.4000	0.0421	0.3701	**0.3906**	0.5035	0.2676	**0.3906**	**0.3906**	0.2870	**0.3906**	0.4132	**0.3906**
Closeness+	0.0083	0.0083	0.2467	-0.0244	0.2156	0.2241	0.0083	0.2228	0.3750	0.3558	0.2327	0.3750	0.3750	0.2168	0.3750	0.3541	0.3750
LDA	0.0369	0.0369	**0.4047**	**0.0606**	**0.5373**	**0.4614**	**0.0369**	0.2902	0.3125	0.5464	0.1966	0.3125	0.3125	0.2117	0.3125	0.3674	0.3125
LDA w/ Gibbs	-0.0252	-0.0252	0.3417	-0.0332	0.4635	0.3930	-0.0252	0.3242	0.3125	**0.5701**	0.2553	0.3125	0.3125	0.2460	0.3125	0.3496	0.3125
LSI	0.0203	0.0203	0.3645	0.0205	0.3950	0.3772	0.0203	0.3618	0.3594	0.5117	0.2389	0.3594	0.3594	0.2675	0.3594	0.3953	0.3594

Table A.3 Average scores for topic modelling clustering and analysis on Article 1 (2 annotators)

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.1288	0.1288	0.4387	0.1347	0.5071	0.4704	0.1288	0.3629	0.4194	0.6094	0.2649	0.4194	0.4194	0.2778	0.4194	0.4581	0.4194
Pagerank+	0.1089	0.1089	0.4402	**0.1497**	0.4489	0.4435	0.1089	0.3052	0.4194	0.4556	0.3259	0.4194	0.4194	0.2788	0.4194	0.3993	0.4194
*Eigenvector^*	0.0955	0.0955	0.4258	0.0906	0.5186	0.4675	0.0955	0.2909	0.3871	0.5231	0.2592	0.3871	0.3871	0.2478	0.3871	0.4024	0.3871
Eigenvector+	0.0331	0.0331	0.3769	0.0463	0.3751	0.3744	0.0331	0.3165	0.3548	0.4930	0.2464	0.3548	0.3548	0.2253	0.3548	0.3351	0.3548
*Degree^*	0.1214	0.1214	0.4223	0.1300	0.4209	0.4208	0.1214	0.3557	0.4516	0.5060	**0.3641**	0.4516	0.4516	0.3266	0.4516	0.4413	0.4516
Degree+	0.0903	0.0903	0.4205	0.1122	0.4483	0.4339	0.0903	0.3327	0.3871	0.5058	0.2476	0.3871	0.3871	0.2488	0.3871	0.3955	0.3871
*Closeness^*	**0.1698**	**0.1698**	**0.4751**	0.1405	**0.5975**	**0.5291**	**0.1698**	0.3782	**0.4677**	**0.6978**	0.2958	**0.4677**	**0.4677**	0.3026	**0.4677**	**0.5181**	**0.4677**
Closeness+	0.0621	0.0621	0.4042	0.0537	0.4723	0.4356	0.0621	0.3232	0.3871	0.5996	0.2717	0.3871	0.3871	0.2448	0.3871	0.3951	0.3871
LDA	0.0183	0.0183	0.3748	0.0297	0.3914	0.3814	0.0183	**0.4402**	0.3871	0.6208	0.3401	0.3871	0.3871	**0.3287**	0.3871	0.3875	0.3871
LDA w/ Gibbs	-0.0042	-0.0042	0.3479	0.0337	0.3256	0.3356	-0.0042	0.3986	0.3710	0.4897	0.3320	0.3710	0.3710	0.2992	0.3710	0.3563	0.3710
LSI	-0.0005	-0.0005	0.4075	0.0010	0.1877	0.2568	-0.0005	0.4209	0.3548	0.5250	0.2200	0.3548	0.3548	0.2134	0.3548	0.2961	0.3548

Table A.4 Scores for topic modelling clustering and analysis on Article 23, annotator 8 with 13 topics

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.0722	0.0722	0.6188	0.0561	0.4354	0.5112	0.0722	0.2390	0.3667	0.2881	0.2628	0.3667	0.3667	0.2423	0.3667	0.3067	0.3667
Pagerank+	0.1065	0.1065	0.6491	0.0847	0.5325	0.5851	0.1065	**0.5115**	**0.4667**	**0.5811**	**0.4423**	**0.4667**	**0.4667**	**0.4154**	**0.4667**	**0.4404**	**0.4667**
*Eigenvector^*	0.0609	0.0609	0.6207	0.0555	0.4069	0.4916	0.0609	0.3641	0.4333	0.4533	0.3141	0.4333	0.4333	0.3045	0.4333	0.3828	0.4333
Eigenvector+	0.0517	0.0517	0.5597	0.0390	0.2912	0.3831	0.0517	0.1603	0.3333	0.2556	0.1795	0.3333	0.3333	0.1444	0.3333	0.2516	0.3333
*Degree^*	**0.1426**	**0.1426**	**0.6821**	**0.1472**	0.5335	**0.5987**	**0.1426**	0.3141	**0.4667**	0.4222	0.4038	**0.4667**	0.4667	0.3145	**0.4667**	0.3900	**0.4667**
Degree+	0.0994	0.0994	0.6202	0.0675	0.4699	0.5347	0.0994	0.1923	0.3667	0.2861	0.2628	0.3667	0.3667	0.2032	0.3667	0.3025	0.3667
*Closeness^*	0.1095	0.1095	0.6730	0.1001	0.3777	0.4839	0.1095	0.2630	0.4333	0.3406	0.2885	0.4333	0.4333	0.2589	0.4333	0.3481	0.4333
Closeness+	0.1111	0.1111	0.6385	0.0927	0.4738	0.5439	0.1111	0.3256	0.4333	0.3867	0.3526	0.4333	0.4333	0.3141	0.4333	0.3759	0.4333
LDA	0.0462	0.0462	0.6250	0.0814	0.5385	0.5785	0.0462	0.3040	0.4000	0.3548	0.4103	0.4000	0.4000	0.3297	0.4000	0.3529	0.4000
LDA w/ Gibbs	0.0278	0.0278	0.6062	0.0304	**0.5463**	0.5747	0.0278	0.2622	0.3667	0.3456	0.3013	0.3667	0.3667	0.2682	0.3667	0.3414	0.3667
LSI	0.0388	0.0388	0.6208	0.0619	0.4728	0.5368	0.0388	0.2872	0.3333	0.2700	0.3141	0.3333	0.3333	0.2970	0.3333	0.2926	0.3333

Table A.5 Scores for topic modelling clustering and analysis on Article 23, annotator 9 with 10 topics

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.0598	0.0598	**0.4352**	0.1012	0.4094	0.4219	0.0598	**0.5524**	0.4333	0.6094	**0.4449**	0.4333	0.4333	**0.4422**	0.4333	0.4426	0.4333
Pagerank+	0.0639	0.0639	**0.4384**	0.1069	0.4558	0.4469	0.0639	0.4593	**0.4667**	**0.6300**	0.4061	**0.4667**	**0.4667**	0.3831	**0.4667**	0.4868	**0.4667**
*Eigenvector^*	0.0812	0.0812	0.4081	0.0942	0.3699	0.3881	0.0812	0.4639	0.4333	0.5149	0.3733	0.4333	0.4333	0.3748	0.4333	0.4340	0.4333
Eigenvector+	0.0156	0.0156	0.3141	0.0179	0.2553	0.2817	0.0156	0.3167	0.4000	0.3856	0.4150	0.4000	0.4000	0.3317	0.4000	0.3695	0.4000
*Degree^*	0.0300	0.0300	0.4046	0.0746	0.4080	0.4063	0.0300	0.5000	0.4000	0.5444	0.3787	0.4000	0.4000	0.4103	0.4000	0.4316	0.4000
Degree+	0.0246	0.0246	0.3979	0.0648	0.3968	0.3973	0.0246	0.3417	0.4000	0.5144	0.3475	0.4000	0.4000	0.3097	0.4000	0.4052	0.4000
*Closeness^*	-0.0122	-0.0122	0.3390	0.0077	0.2623	0.2957	-0.0122	0.3619	0.3333	0.5122	0.3835	0.3333	0.3333	0.2639	0.3333	0.3022	0.3333
Closeness+	-0.0108	-0.0108	0.2062	-0.0210	0.1163	0.1487	-0.0108	0.1344	0.3333	0.1897	0.2423	0.3333	0.3333	0.1727	0.3333	0.2417	0.3333
LDA	0.0645	0.0645	0.3782	0.0941	0.4544	0.4128	0.0645	0.4479	0.4333	0.7333	0.3452	0.4333	0.4333	0.3479	0.4333	**0.5061**	0.4333
LDA w/ Gibbs	**0.1429**	**0.1429**	0.4231	**0.1547**	**0.4980**	**0.4575**	**0.1429**	0.3281	0.4000	0.4625	0.3199	0.4000	0.4000	0.3066	0.4000	0.4125	0.4000
LSI	-0.0369	-0.0369	0.2730	-0.0538	0.3200	0.2947	-0.0369	0.2604	0.3000	0.4333	0.2440	0.3000	0.3000	0.2229	0.3000	0.3322	0.3000

Table A.6 Scores for topic modelling clustering and analysis on Article 23 , annotator 11 with 6 topics

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.0040	0.0040	0.1893	-0.0310	0.1712	0.1798	0.0040	0.2476	0.4000	0.3917	0.2627	0.4000	0.4000	0.2453	0.4000	0.3793	0.4000
Pagerank+	-0.0318	-0.0318	0.1702	-0.0767	0.1841	0.1769	-0.0318	0.2536	0.3667	0.4129	0.2412	0.3667	0.3667	0.2391	0.3667	0.3776	0.3667
*Eigenvector^*	0.0199	0.0199	0.2433	0.0099	0.2531	0.2481	0.0199	0.3353	**0.4333**	0.5401	0.2730	**0.4333**	**0.4333**	0.2845	**0.4333**	**0.4554**	**0.4333**
Eigenvector+	0.0387	0.0387	0.1626	-0.0232	0.1528	0.1575	0.0387	0.3195	**0.4333**	0.4694	0.3591	**0.4333**	**0.4333**	0.3160	**0.4333**	0.4385	**0.4333**
*Degree^*	-0.0380	-0.0380	0.1442	-0.0809	0.1271	0.1351	-0.0380	0.2524	0.4000	0.3706	0.2609	0.4000	0.4000	0.2524	0.4000	0.3794	0.4000
Degree+	-0.0380	-0.0380	0.1442	-0.0809	0.1271	0.1351	-0.0380	0.2524	0.4000	0.3706	0.2609	0.4000	0.4000	0.2524	0.4000	0.3794	0.4000
*Closeness^*	-0.0416	-0.0416	0.1733	-0.0717	0.1924	0.1823	-0.0416	0.3194	0.3667	0.4917	0.3093	0.3667	0.3667	0.2826	0.3667	0.3807	0.3667
Closeness+	-0.0208	-0.0208	0.2028	-0.0250	0.1731	0.1868	-0.0208	0.2657	0.4000	0.3984	0.3127	0.4000	0.4000	0.2645	0.4000	0.3676	0.4000
LDA	0.0071	0.0071	0.2354	0.0181	0.2769	0.2545	0.0071	0.3398	0.3667	**0.5563**	0.3328	0.3667	0.3667	0.2939	0.3667	0.4021	0.3667
LDA w/ Gibbs	-0.0380	-0.0380	0.2236	-0.0277	0.2154	0.2194	-0.0380	0.2569	0.3667	0.4361	0.2023	0.3667	0.3667	0.2109	0.3667	0.3708	0.3667
LSI	**0.1116**	**0.1116**	**0.3549**	**0.1578**	**0.3260**	**0.3398**	**0.1116**	**0.4424**	**0.4333**	0.4849	**0.3645**	**0.4333**	**0.4333**	**0.3413**	**0.4333**	0.4111	**0.4333**

Table A.7 Scores for topic modelling clustering and analysis on Article 11, annotator 3 with 15 topics

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.0081	0.0081	0.3701	0.0118	0.4384	0.4014	0.0081	**0.4394**	0.3750	**0.6615**	0.2091	0.3750	0.3750	0.2590	0.3750	**0.4398**	0.3750
Pagerank+	-0.0284	-0.0284	0.3114	-0.0711	0.3867	0.3450	-0.0284	0.2652	0.3438	0.4427	0.2030	0.3438	0.3438	0.2207	0.3438	0.3792	0.3438
*Eigenvector^*	0.0081	0.0081	0.3688	0.0241	0.4204	0.3929	0.0081	0.3900	0.4063	0.5500	0.2550	**0.4063**	**0.4063**	0.2894	0.4063	0.4423	0.4063
Eigenvector+	-0.0094	-0.0094	0.3239	-0.0184	0.3616	0.3417	-0.0094	0.4206	0.3750	0.5424	**0.2852**	0.3750	0.3750	**0.3049**	0.3750	0.4104	0.3750
*Degree^*	-0.0176	-0.0176	0.3092	-0.0685	0.3581	0.3318	-0.0176	0.2500	0.3750	0.4219	0.2300	0.3750	0.3750	0.2335	0.3750	0.3912	0.3750
Degree+	0.0020	0.0020	0.3516	-0.0120	0.4439	0.3924	0.0020	0.3864	0.3750	0.5703	0.2485	0.3750	0.3750	0.2846	0.3750	0.4310	0.3750
*Closeness^*	0.0529	0.0529	0.4121	0.1106	0.4455	0.4282	0.0529	0.3333	**0.4063**	0.4609	0.2833	**0.4063**	**0.4063**	0.2891	**0.4063**	0.4197	**0.4063**
Closeness+	-0.0146	-0.0146	0.1573	-0.0375	0.0764	0.1029	-0.0146	0.0915	0.3438	0.2429	0.1571	0.3438	0.3438	0.1113	0.3438	0.2794	0.3438
LDA	**0.0557**	**0.0557**	**0.4358**	**0.1182**	**0.6063**	**0.5071**	**0.0557**	0.2417	0.3125	0.5135	0.1750	0.3125	0.3125	0.1910	0.3125	0.3633	0.3125
LDA w/ Gibbs	-0.0166	-0.0166	0.3540	-0.0005	0.5093	0.4177	-0.0166	0.3318	0.3438	0.6021	0.3000	0.3438	0.3438	0.2853	0.3438	0.3737	0.3438
LSI	-0.0047	-0.0047	0.3277	-0.0225	0.4113	0.3648	-0.0047	0.3972	0.3125	0.5781	0.2167	0.3125	0.3125	0.2517	0.3125	0.3797	0.3125

Table A.8 Scores for topic modelling clustering and analysis on Article 11, annotator 9 with 13 topics

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	0.0207	0.0207	0.3778	-0.0051	0.4255	0.4002	0.0207	0.3223	0.3438	0.5090	0.1879	0.3438	0.3438	0.2214	0.3438	0.3772	0.3438
Pagerank+	0.0204	0.0204	0.3334	-0.0377	0.3630	0.3476	0.0204	0.2722	0.3438	0.4094	0.2481	0.3438	0.3438	0.2394	0.3438	0.3523	0.3438
*Eigenvector^*	0.0006	0.0006	0.3374	-0.0469	0.3568	0.3468	0.0006	0.3628	0.3750	0.5689	0.2267	0.3750	0.3750	0.2397	0.3750	0.4040	0.3750
Eigenvector+	-0.0058	-0.0058	0.2789	-0.0629	0.2614	0.2699	-0.0058	0.3922	0.3438	0.4670	**0.3190**	0.3438	0.3438	0.2752	0.3438	0.3526	0.3438
*Degree^*	**0.0633**	**0.0633**	**0.4160**	**0.0596**	0.4024	0.4091	**0.0633**	0.3771	0.3438	0.5286	0.1950	0.3438	0.3438	0.2233	0.3438	0.3583	0.3438
Degree+	0.0509	0.0509	0.3423	-0.0216	0.3857	0.3627	0.0509	0.3013	0.3750	0.4337	0.2852	0.3750	0.3750	0.2795	0.3750	0.3893	0.3750
*Closeness^*	0.0314	0.0314	0.3634	0.0105	0.3805	0.3718	0.0314	**0.4068**	0.3750	0.5460	0.2519	0.3750	0.3750	0.2848	0.3750	0.4066	0.3750
Closeness+	0.0313	0.0313	0.3361	-0.0112	0.3548	0.3452	0.0313	0.3542	**0.4063**	0.4688	0.3083	**0.4063**	**0.4063**	**0.3222**	**0.4063**	**0.4288**	0.4063
LDA	0.0181	0.0181	0.3737	0.0031	**0.4683**	**0.4157**	0.0181	0.3387	0.3125	**0.5792**	0.2182	0.3125	0.3125	0.2323	0.3125	0.3715	0.3125
LDA w/ Gibbs	-0.0339	-0.0339	0.3294	-0.0659	0.4177	0.3684	-0.0339	0.3167	0.2813	0.5380	0.2106	0.2813	0.2813	0.2067	0.2813	0.3255	0.2813
LSI	0.0453	0.0453	0.4013	0.0634	0.3787	0.3897	0.0453	0.3264	**0.4063**	0.4453	0.2611	**0.4063**	**0.4063**	0.2833	**0.4063**	0.4109	**0.4063**

Table A.9 Scores for topic modelling clustering and analysis on Article 1, annotator 1 with 13 topics

Technique	Adjusted rand score	Silhouette score	Completeness score	Adjusted mutual info score	Homogeneity score	V measure score	Fowlkes mallows score	Precision macro	Precision micro	Precision weighted	Recall macro	Recall micro	Recall weighted	F1 macro	F1 micro	F1 weighted	Accuracy
*Pagerank^*	**0.1233**	**0.1233**	**0.5109**	**0.1796**	**0.5940**	**0.5493**	**0.1233**	0.4341	0.4194	**0.7323**	0.2634	0.4194	0.4194	0.2879	0.4194	**0.4731**	0.4194
Pagerank+	0.1024	0.1024	0.4849	0.1843	0.4514	0.4675	0.1024	0.3733	0.4516	0.4713	0.4309	0.4516	0.4516	0.3441	0.4516	0.4023	0.4516
*Eigenvector^*	0.0446	0.0446	0.4653	0.1027	0.5459	0.5024	0.0446	0.2902	0.3548	0.5059	0.2696	0.3548	0.3548	0.2381	0.3548	0.3505	0.3548
Eigenvector+	0.0168	0.0168	0.4331	0.0660	0.4793	0.4550	0.0168	0.2208	0.3226	0.3925	0.2598	0.3226	0.3226	0.2130	0.3226	0.2928	0.3226
*Degree^*	0.1192	0.1192	0.4743	0.1653	0.4368	0.4548	0.1192	0.4219	**0.4839**	0.5331	**0.4485**	**0.4839**	**0.4839**	0.3848	**0.4839**	0.4596	**0.4839**
Degree+	0.0541	0.0541	0.4814	0.1415	0.5192	0.4996	0.0541	0.3375	0.3226	0.4677	0.2556	0.3226	0.3226	0.2433	0.3226	0.3247	0.3226
*Closeness^*	0.0228	0.0228	0.4325	0.0199	0.5155	0.4704	0.0228	0.2917	0.3871	0.5032	0.2838	0.3871	0.3871	0.2618	0.3871	0.3976	0.3871
Closeness+	0.0258	0.0258	0.4403	0.0595	0.5054	0.4706	0.0258	0.3079	0.3548	0.5849	0.3196	0.3548	0.3548	0.2412	0.3548	0.3312	0.3548
LDA	0.0164	0.0164	0.4288	0.0573	0.4964	0.4602	0.0164	**0.4729**	0.4194	0.6876	0.4165	0.4194	0.4194	**0.3864**	0.4194	0.4251	0.4194
LDA w/ Gibbs	0.0242	0.0242	0.4333	0.0764	0.4348	0.4340	0.0242	0.4648	0.4194	0.6016	0.4239	0.4194	0.4194	0.3581	0.4194	0.4031	0.4194
LSI	0.0273	0.0273	0.4655	0.0285	0.2268	0.3050	0.0273	0.2969	0.3871	0.3911	0.2210	0.3871	0.3871	0.2101	0.3871	0.3115	0.3871

Table A.10 Scores for topic modelling clustering and analysis on Article 1, annotator 2 with 10 topics

Technique	**Adjusted rand score**	**Silhouette score**	**Completeness score**	**Adjusted mutual info score**	**Homogeneity score**	**V measure score**	**Fowlkes mallows score**	**Precision macro**	**Precision micro**	**Precision weighted**	**Recall macro**	**Recall micro**	**Recall weighted**	**F1 macro**	**F1 micro**	**F1 weighted**	**Accuracy**
Technique	**Adjusted rand score**	**Silhouette score**	**Completeness score**	**Adjusted mutua info score**	**Homogeneity score**	**V measure score**	**Fowlkes mallows score**	**Precision macro**	**Precision micro**	**Precision weighted**	**Recall macro**	**Recall micro**	**Recall weighted**	**F1 macro**	**F1 micro**	**F1 weighted**	**accuracy**
*Pagerank^*	0.1343	0.1343	0.3665	0.0899	0.4203	0.3916	0.1343	0.2917	0.4194	0.4866	0.2664	0.4194	0.4194	0.2677	0.4194	0.4431	0.4194
Pagerank+	0.1154	0.1154	0.3955	0.1151	0.4464	0.4194	0.1154	0.2370	0.3871	0.4398	0.2209	0.3871	0.3871	0.2136	0.3871	0.3964	0.3871
*Eigenvector^*	0.1463	0.1463	0.3863	0.0785	0.4914	0.4326	0.1463	0.2917	0.4194	0.5403	0.2488	0.4194	0.4194	0.2575	0.4194	0.4543	0.4194
Eigenvector+	0.0495	0.0495	0.3207	0.0267	0.2709	0.2937	0.0495	0.4122	0.3871	0.5935	0.2330	0.3871	0.3871	0.2375	0.3871	0.3774	0.3871
*Degree^*	0.1236	0.1236	0.3702	0.0947	0.4051	0.3869	0.1236	0.2896	0.4194	0.4790	0.2798	0.4194	0.4194	0.2684	0.4194	0.4229	0.4194
Degree+	0.1265	0.1265	0.3596	0.0829	0.3774	0.3683	0.1265	0.3279	0.4516	0.5439	0.2396	0.4516	0.4516	0.2543	0.4516	0.4662	0.4516
*Closeness^*	**0.3168**	**0.3168**	**0.5178**	**0.2610**	**0.6795**	**0.5877**	**0.3168**	0.4646	**0.5484**	**0.8925**	**0.3077**	**0.5484**	**0.5484**	**0.3434**	**0.5484**	**0.6387**	**0.5484**
Closeness+	0.0985	0.0985	0.3682	0.0479	0.4392	0.4006	0.0985	0.3386	0.4194	0.6143	0.2238	0.4194	0.4194	0.2484	0.4194	0.4590	0.4194
LDA	0.0203	0.0203	0.3208	0.0021	0.2863	0.3026	0.0203	0.4076	0.3548	0.5541	0.2636	0.3548	0.3548	0.2710	0.3548	0.3499	0.3548
LDA w/ Gibbs	-0.0327	-0.0327	0.2626	-0.0089	0.2163	0.2372	-0.0327	0.3323	0.3226	0.3777	0.2401	0.3226	0.3226	0.2403	0.3226	0.3096	0.3226
LSI	-0.0283	-0.0283	0.3495	-0.0265	0.1486	0.2086	-0.0283	**0.5449**	0.3226	0.6588	0.2189	0.3226	0.3226	0.2167	0.3226	0.2806	0.3226

Appendix B

Online interactive visualization

For visualizing AG, we built an web application, available online[1].

In the user interface, as demonstrated in figure B.1, support 4 primary features:

- **Display aspects**. Aspects are denotes by rounded rectangles, whose size and color is corresponding to its sentiment score in scale from red (very negative) to green (very positive). Hovering mouse over an aspect show additional information, consist of its nodes' label, total frequency and sentiment score, as shown in figure B.2. It also display all comments that linked to this aspect at the right side of the interface.
- **Display comments**. Comments pop out as user hover its corresponding icon, nodes or aspect that it connected to.
- **Aspect inspection**. Inspection gives a closer look at the aspect's corresponding community, as shown in figure B.3.
- **Edit graph**. After expanding AG to its original form, several modifications can be made, including add, edit of remove nodes and edges.

To begin, user need to open the graph description file generated by AG construction program. This file is in JSON(JavaScript Object Notation) format, containing metadata, nodes, edges and comments. This can be done either drag and drop the file to the visualization area or select *Open and Draw*. For viewing the original AG without being grouped by aspects, press *Expand all clusters*. In order to refresh the graph, press *Reraw*.

[1] cassandra.disi.unitn.it/sm_summarization/

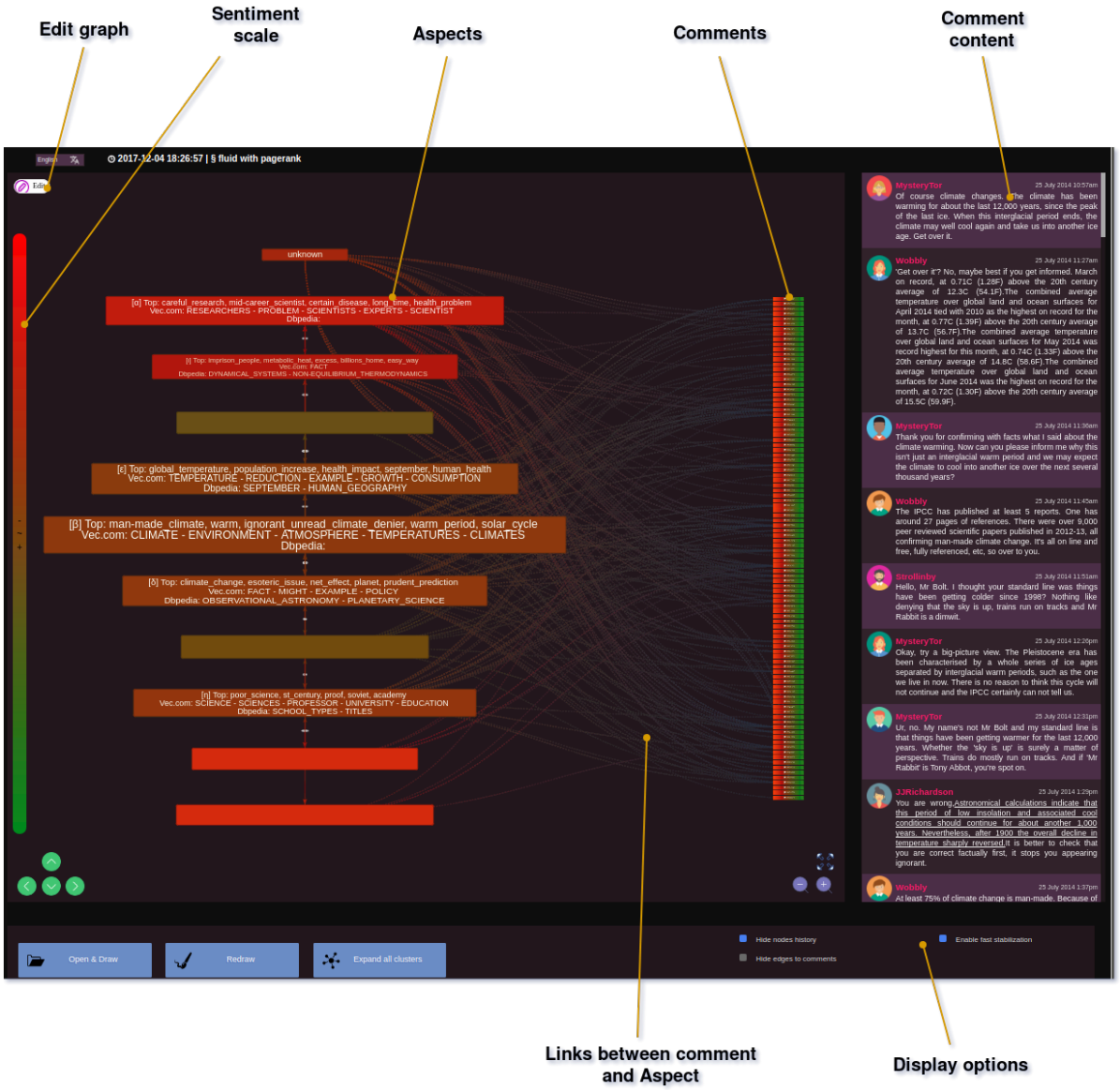


Figure B.1 Interface of online visualization application.

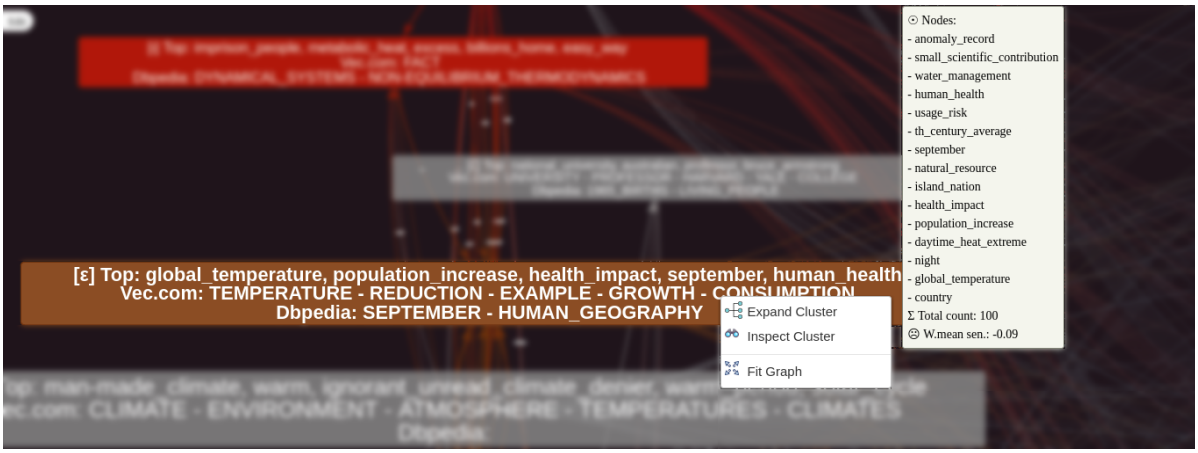


Figure B.2 View keyphrases and options for an aspect

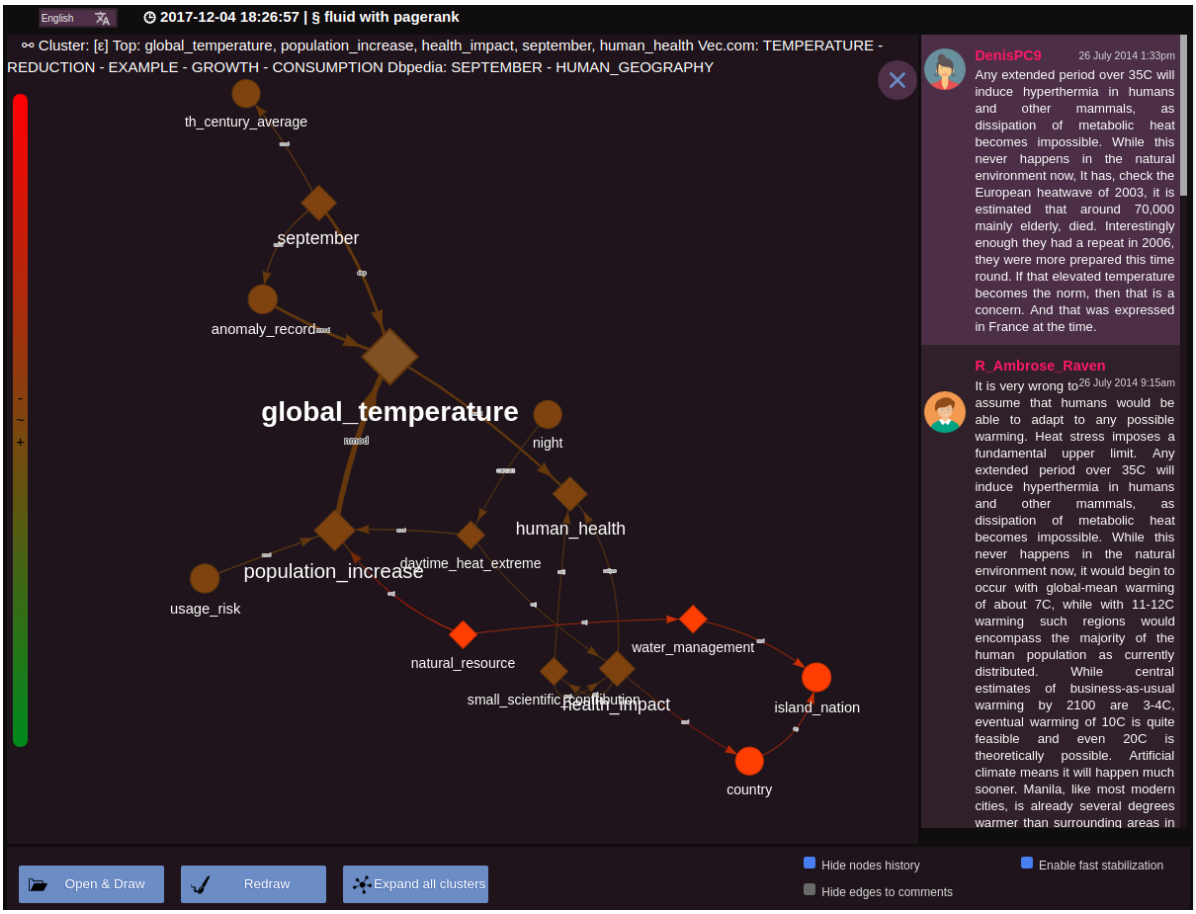


Figure B.3 Inspect an aspect.

Index