

A data-driven approach to automatic tweet generation about traffic incidents

by

Khoa Tran

B.Sc., University of Science,
Vietnam National University at Ho Chi Minh City, 2013

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Science

© Khoa Tran 2016
SIMON FRASER UNIVERSITY
Fall 2016

All rights reserved.

However, in accordance with the *Copyright Act of Canada*, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, education, satire, parody, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

Approval

Name: Khoa Tran
Degree: Master of Science (Computing Science)
Title: *A data-driven approach to automatic tweet generation about traffic incidents*
Examining Committee: **Chair:** Leonid Chindelevitch
Assistant Professor

Fred Popowich
Senior Supervisor
Professor

Anoop Sarkar
Supervisor
Professor

Giuseppe Carenini
External Examiner
Associate Professor, UBC

Date Defended: November 7, 2016

Abstract

Traffic congestion continues to be a major problem in large cities around the world and a source of frustration for drivers. Previous studies show that providing drivers with real-time traffic information will help them make better route planning and avoid congestion. In this research, we examine the use of data-driven natural language generation (NLG) techniques to automatically generate tweets from traffic incident data. From the task of automatic tweet generation, we discuss and propose a design of a traffic notification system that can deliver personalized and location-relevant real-time traffic information to drivers. The domain of our NLG work is novel with respect to the previous work in different domains including weather forecasts, educational reports and clinical reports. We evaluate the automatic generated tweets using BLEU-4. Our experimental results show that a well-prepared training corpus is important for better quality output, however, it is currently limited in traffic-related domains.

Keywords: automatic tweet generation, natural language generation, traffic incidents

Acknowledgements

I would like to express my deepest appreciation to Dr. Fred Popowich, my senior supervisor, for his guidance and patience throughout my process of research. His persistent support and dedication have always been the encouragement and inspiration to me during the past years. I would also like to thank Dr. Anoop Sarkar, my supervisor and Dr. Giuseppe Carenini, my thesis examiner for their valuable comments and feedback. Thanks also to Dr. Leonid Chindelevitch for chairing my thesis defense.

A special thank must go to Dr. Ali Tizghadam. Without his assistance in providing access to the necessary data, this thesis would not have been possible.

Last but not least, I would like to acknowledge with gratitude, the support of my family throughout my years of study and through the process of pursuing this degree. And finally, many thanks to Thu, my fiancée, for her unconditioned love and understanding that helped me get through difficult times.

Table of Contents

Approval	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Motivation	1
1.2 Our approach	2
1.3 Thesis outline	3
2 Background	4
2.1 NLG Systems	4
2.1.1 Text-to-text systems	5
2.1.2 Data-to-text systems	6
2.2 Automatic tweet generation	8
2.3 Summary	10
3 Traffic data	11
3.1 The traffic task definition	11
3.2 Traffic-related Datasets	12
3.3 CVST Dataset	13
3.3.1 Road incidents	14
3.3.2 Twitter traffic reports	14
3.3.3 Duo corpus: Matching road incidents with tweets	16
3.3.4 Single corpus: New version of Twitter traffic reports API	16
3.3.5 Data pre-processing	19

3.4	Summary	19
4	Traffic notification system	21
4.1	The alignment model	22
4.2	The generation model	24
4.3	Location-based user model	29
4.4	Content-preference model	31
4.5	Summary	33
5	Experimental results	34
5.1	Implementation and training	34
5.2	Evaluation metric	36
5.3	Experiments	38
5.3.1	Experiment 1	38
5.3.2	Experiment 2	46
5.3.3	Summary	49
6	Conclusion and Future work	51
	Bibliography	53
	Appendix A NLTK Usage	57

List of Tables

Table 2.1	Example input sentence, templates and output questions of Lindberg et al.’s automatic question generation system [26].	7
Table 2.2	Example input and output of Konstas’s system in weather forecast domain [19].	9
Table 3.1	The structure of data \mathbf{d} in the Duo corpus.	20
Table 3.2	The structure of data \mathbf{d} in the Single corpus.	20
Table 4.1	Grammar rules used for generation with their corresponding weights.	25
Table 4.2	The basic decoder deductive system.	26
Table 5.1	Results (BLEU-4 scores) of experiment 1.	39
Table 5.2	Example tweets from different users. Each user has a different style of writing their own tweets.	40
Table 5.3	Top 5-scoring items of the multinomial distributions for some of the grammar rules in the experiment 1.	41
Table 5.4	Top 5-scoring items of the multinomial distributions for some of the grammar rules in the experiment 1 extracted from Table 5.3b.	44
Table 5.5	The number of tweets of some majority users and their percentage in our corpora.	46
Table 5.6	Number of scenarios in training set and test set for each corpus in experiment 2	46
Table 5.7	Results (BLEU-4 scores) of experiment 2.	47

List of Figures

Figure 2.1	Example dialogue with MATCH system [45].	6
Figure 3.1	Example of a scenario in the traffic incident domain. Data entry d consists of 8 records belonging to 6 record types: Main road , Reference road name , Lane , Condition , Reason and Incident Type . Record type Main road is described through 2 fields: Name and Direction while other record types only have one field associated with each of them. The textual description w of d is a tweet mentioning that traffic incident.	12
Figure 3.2	Example record in the Dublin City Council’s road works and maintenance dataset. The Work Description field does not describe essential information such as where and when the resurfacing work was. . . .	13
Figure 3.3	Example record in the Bing Maps’ Traffic Incidents. Street names included in the description cannot be inferred from the data entry. . . .	14
Figure 3.4	Examples of data records from the road incident dataset.	15
Figure 3.5	Examples of data records from the Twitter traffic report dataset.	15
Figure 3.6	Example of matching two records from the traffic incident dataset and Twitter traffic report dataset. The left column shows two individual records before matching. Since both events’ start times are within 16 minutes of each other and the locations are 25 meters away from each other, they are matched into a single record on the right column.	17
Figure 3.7	Example of results returned from the new API.	18
Figure 4.1	Design of the traffic notification system that provides location-relevant traffic information for road users.	22
Figure 4.2	Graphical model representing Liang et al.’s alignment model. Records are chosen and ordered from d . Then, fields are chosen for each record. Finally, words are chosen for each field. The set of records d and the words w are observed, while (r, f, c) are latent variables to be inferred [23].	23

Figure 4.3	Example output of the alignment model for a tweet about traffic incident.	24
Figure 4.4	One of the derivation trees using the grammar in Table 4.1 for the text “Gardiner Express EB at Don Valley Parkway”. In this figure, we use “mr” as a shorthand for “main_road” and “rrn” as a shorthand for “reference_road_name”.	27
Figure 4.5	Konstas’s Viterbi search for the basic decoder	28
Figure 4.6	An example of Krumm et al. system’s prediction results [21]. The start location is in the lower right of the map. The black line shows the complete trip, and the grey circles show the intersections along the partial travelled route. Destinations with higher probabilities are shaded darker.	30
Figure 5.1	Log likelihood of the data in the two corpora during the training iterations using EM algorithm.	35
Figure 5.2	Examples of scenarios where structured data does not have all information mentioned in tweets.	43
Figure 5.3	Examples of tweets generated by our k-best decoder in experiment 1 with $k = 50$	45
Figure 5.4	Example of a generated tweet in experiment 2 with training data extracted from the Single corpus containing tweets from “Ontario Roads” only. We use $k = 50$ in this experiment.	47
Figure 5.5	Examples of generated tweets in experiment 2 with training data extracted from the Single corpus, contains tweets from “680 NEWS Traffic” only. We use $k = 50$ in this experiment.	49

Chapter 1

Introduction

Natural Language Generation (NLG) is a task of Natural Language Processing (NLP) that generates natural language from a machine representation (e.g., structured data, logic form, etc.). The generated texts are intended to be readable and understandable by humans. Their applications include text summarization, automatic generation of reports and help messages, and virtual assistance where a machine can assist and communicate with humans through natural languages.

In this research, we present an NLG system that we first introduced in [43] that can generate tweet messages from structured data about traffic incidents. We examine different traffic related datasets and NLG techniques and explain why data-driven approaches are useful in automatic traffic notification generation applications. We also discuss and propose a traffic notification system that can generate personalized tweets messages for users based on their preferences and locations.

1.1 Motivation

Traffic congestion continues to be a major problem in large cities around the world, and a source of frustration for commuters, commercial drivers, tourists, and even occasional drivers.

Current efforts to reduce congestion and frustration often involve providing road users with real-time traffic information to help estimate travel time accurately, resulting in better route planning and travel decisions [44]. Previous studies show that provision of real-time traffic information affects route-choice behavior of drivers and also helps alleviate congestion [44] [13] [15]. The different channels to deliver traffic information include variable-message signs (VMS), radio, smart navigation devices and social networks. Information from VMS, radio and social networks is delivered as messages which consist primarily of natural language. When delivered on smart navigation devices, information is presented with colour

and icons on interactive maps; for example, congested road segments are usually in red while clear road segments are in green ¹.

Text and audio messages associated with radio and social network channels are mainly human-generated. The information sources for these messages mainly utilize the same data used by smart navigation devices in conjunction with camera images, eye-witness reports and other sources, which collectively require substantial effort and time. Although several social network channels may use computer programs (i.e., “bots”) to generate messages automatically from a data source, these messages are constructed with strict templates which are perceived by users as cold, distant and unnatural.

In this thesis, we present a system that can automatically generate tweets about traffic incidents. Our system is developed based on data-driven NLG techniques where natural language corpora are used to train the machine so that it can generate more natural texts.

1.2 Our approach

We focus on a generation system that can be applied to different types of traffic datasets (road closures, road incidents, traffic flow, etc.). We use an existing alignment model [23] to learn the semantic correspondences between traffic data and its textual description. Konstas’s concept-to-text generation approach [19] is used for the automatic generation of tweets to be ultimately incorporated into a real-time system.

Overall, we choose a data-driven approach since it is location independent. Different cities have different kinds of traffic data and information about road closures, road incidents and road conditions; each with different kinds of data structures. With a data-driven approach, we can handle different datasets without changing the model structure, incorporating it into an end-to-end system with surface realization and content planning in one model. Otherwise, template-based approaches require the construction of templates and rules that are domain dependent and location dependent. They are therefore not flexible enough to apply to different datasets.

We construct our training corpora from data provided by the CVST portal ². We also explore other traffic related datasets that can be used to train our system. However, using such data is restricted as discussed in Section 3.2.

Using the constructed corpora, we apply a semantic alignment model (Section 4.1) to learn the semantic correspondences between data records and their textual descriptions in the tweets. Then, we apply a model for concept-to-text generation (Section 4.2) to generate tweets about traffic incidents from given records. However, our system’s output is not limited to tweet generation. Output can be personalized, for example, as a virtual assistant,

¹Google Maps uses this colour code as described in <https://support.google.com/maps/answer/3092439?hl=en&rd=1>

²<http://portal.cvst.ca>

to generate traffic notifications for users based on their driving routines incorporating daily routes, departure and arrival times, and specific locations. Previous work on capturing users' locations and route prediction (Section 4.3) can be applied to select only the potentially user-interested traffic information and deliver it to the drivers.

The evaluation of automatically generated tweets can be approached from several perspectives. Evaluation in the context of the task outlined above can involve human subjects, looking at metrics such as the usefulness of the tweets (using rating criteria like those in rating the helpfulness of reviews or comments), and the quality of the tweets (involving fluency and readability). Detailed human evaluation of the tweets is beyond the scope of this research. We focus on automated techniques in the evaluation of the automatically generated tweets, given that we have a reference of generated texts. To evaluate our models, we build on BLEU-4, a standard evaluation metric used in the Machine Translation community [19].

We make several contributions in this thesis. We introduce an NLG system that automatically generates personalized and location-relevant traffic notifications to drivers based on structured data about traffic. Our proposed system consists of three main models: the generation model, the location-based user model and the content-preference model. We examine the use of an existing generation model applied to a novel domain: automatic tweet generation about traffic incidents. Different traffic-related datasets are investigated and discussed their opportunities and obstacles to be used as the training corpora for the generation model. We use a data-driven approach that is location dependent and domain dependent, therefore, applicable to different traffic datasets from different cities. Finally, we evaluate the generated tweets using the BLEU-4 score. The experimental results show that a well-prepared training corpus is important for better quality output, however, it is currently limited in traffic-related domains.

1.3 Thesis outline

This thesis is structured into six chapters. In Chapter 2, we provide an overview of NLG systems, examples of those systems and the task of automatic tweet generation. Chapter 3 defines the task of generating text from structured data. In this chapter, we also explore different traffic-related datasets and explain how we collect and construct our corpora. In Chapter 4, we explain the design of our traffic notification system and provide details about each model in the system. The implementation and experimental results of our work are presented in Chapter 5. Finally, Chapter 6 provides a conclusion to our thesis and a discussion about several interesting aspects for future research.

Chapter 2

Background

In this chapter, we provide a definition of NLG systems, looking at the tasks included and how to categorize these systems. We also discuss some examples of different NLG systems in two main categories: **text-to-text** and **data-to-text** systems. Finally, we provide an overview of previous work on automatic tweet generation.

2.1 NLG Systems

According to Stent et al. [41], NLG systems are systems that produce human language artifacts including speech, text, and language-rich multimedia presentations. NLG systems differ in the types of input and output they take and produce, and the degree of interactivity they support. All end-to-end NLG systems include three main tasks:

- Content-selection: determination of “what to say” by making choices and decisions on which content should be selected for generation.
- Surface-realization: determination of “how to say it” including tasks such as assigning content to media, arranging the content or choosing the right vocabularies to use.
- Production: presentation or performance of the generated material which in the case of an embodied conversational agent may include production of body postures, gestures and sign language.

Stent et al. categorize NLG systems based on their input and output. With respect to the types of input, there are two categories: **text-to-text** which accepts text as input and **data-to-text** which accepts data or non-linguistic information as input. There are three categories based on the types of output: systems that produce **language** only (including text and speech), those that produce **multimedia** presentations and those that generate the behavior of **embodied conversational agents**.

2.1.1 Text-to-text systems

Text-to-text NLG systems accept text as input and produce a different text output depending on the applications and the readers. Some applications belonging to this category are text summarization, dialogue systems and automatic question generation.

The MATCH dialogue system [12, 45] is an NLG system that gives users restaurant recommendations. It can understand users' speech commands and based on that input, decides which restaurants should be recommended (content-selection), then generates the recommendation or summarizing text about those restaurants (surface-realization) and presents the results to users through speech or visual presentation. They build a multi-attribute decision model that ranks restaurants based on different attributes such as food quality, cost, decor, service, food type and neighborhood. Each user has a different weight vector for these attributes, depending on the user's preferences. The list of restaurants in the database that matches a user's query is sorted based on the scores returned by the multi-attribute decision model. To present the results to users, they define three strategies: recommending one of a selected set of restaurants, comparing three or more selected restaurants and summarizing a selected set of restaurants. In the first strategy, the restaurant with the highest score ranked by the decision model is selected. Among its attributes, their algorithm only chooses outliers (e.g., the recommended restaurant's attributes that are significantly different from other restaurants from the list) and lexicalizes them. The idea is to present users with only the attributes that make a restaurant outstanding from the list. When comparing different restaurants, they also have similar algorithms to select which restaurants and which attributes to compare. In the summarizing strategy, they first find which attributes have the same values and those that do not. Then, the system simply states the ways in which the restaurants are similar or different. Figure 2.1 shows an example dialogue with MATCH system.

Lindberg et al. develop an automatic question generation system that generates questions related to the input documents [26]. The system is useful in on-line and distance learning applications where a self-learner can quickly review his or her knowledge by answering automatically generated questions. It can also help reduce the effort of the instructors in preparing questions and quizzes. Their system utilizes semantic role labeling (SRL) together with a set of pre-defined question templates to generate meaningful and fluent questions. The original document is first pre-processed and split into sentences. Each source sentence is tokenized and annotated with different syntax and semantic information such as POS tags, named entities, lemmata, and its SRL parse. This information is then used to extract predicated frames which are mapped into a pre-constructed template. Finally, role words are filled into variable slots in the template using a set of predicated filters and modifiers. Table 2.1 shows an example of an input sentence (**S**), selected templates (**T**) and output questions (**Q**) of the system.

U1	<i>“show Italian restaurants in the West Village”</i>
S1	[zooms to the West Village, displays Italian restaurants.]
U2	[circles some highlighted restaurants and says <i>“summarize”</i> .]
S2	<i>“The 3 restaurants have the same price range of 28 to 33 dollars. They differ in food quality and decor.”</i>
U3	[circles a selection of restaurants and says <i>“compare”</i>]
S3	<i>“Among the selected restaurants, the following offer exceptional overall value. Uguale’s price is 33 dollars. It has excellent food quality and good decor. Da Andrea’s price is 28 dollars. It has very good food quality and good decor. John’s Pizzeria’s price is 20 dollars. It has very good food quality and mediocre decor.”</i>
U4	[writes word <i>“recommend”</i>]
S4	<i>“Uguale has the best overall value among the selected restaurants. Uguale’s price is 28 dollars.”</i>

Figure 2.1: Example dialogue with MATCH system [45].

In addition to the above applications, automatic text summarization systems incorporating NLG are used in many contexts. Giving text documents as input, these systems can summarize them into one or two sentences that contain the main ideas of the original documents. SweSum is a text summarization system that scores all the sentences in the document and picks the top scoring (most important) sentence for generation [9]. Sankarasubramanian et al. develop a text summarization system using knowledge source from Wikipedia [37]. The system uses Wikipedia as an additional knowledge source to extract and map concepts in input sentences. It then ranks the concepts and generates summary sentences covering the important concepts. Users can also generate longer summaries incrementally if the generated text is not detailed enough. There are also other text summarization products available publicly online such as OpenTextSummarizer ¹, TextSummarization.net ² or AutoSummarizer.com ³.

2.1.2 Data-to-text systems

Data-to-text systems include all NLG applications that start from non-linguistic inputs. These inputs can be images, logic forms, databases or a combination of these sources. The

¹<https://www.splitbrain.org/services/ots>

²<http://textsummarization.net/text-summarizer>

³<http://autosummarizer.com/>

S	As recently as 12,500 years ago, the Earth was in the midst of a glacial age referred to as the Last Ice Age
T	How would you describe [A2 -lp misc]?
Q	How would you describe the Last Ice Age?
T	Summarize the influence of [A1 -lp !comma !nv] on the environment.
Q	Summarize the influence of a glacial age on the environment.
T	What caused [A2 -lp !nv misc]? ## [A0 null]
Q	What caused the Last Ice Age?

Table 2.1: Example input sentence, templates and output questions of Lindberg et al.’s automatic question generation system [26].

output is usually the textual description of the given data such as: an image description, data summarization and a description.

Automatic image description generation is a task involving both Computer Vision and NLG research. Techniques developed in Computer Vision can extract objects and their attributes in the images. Then, the output can be used by NLG systems to generate the textual descriptions. Some examples of these systems are [46] and [5]. Yang et al. [46] utilize previously developed image detection algorithms to extract objects \mathcal{N} , actions \mathcal{V} , scenes \mathcal{S} and prepositions \mathcal{P} from a given test image I . From a set of detected objects $\mathcal{N}_k = \{n_1, n_2, \dots, n_i\}$ in I , they train a language model L_m to estimate $Pr(v | \mathcal{N}_k)$, the verb v given the objects \mathcal{N}_k ; $Pr(s | n, v)$, the predicted scene given the object and verb; and $Pr(p | s)$, the predicted preposition given the scene. The process is repeated for all n, v, s, p in order to find the most likely quadruplet: $\mathcal{T}^* = \{n^*, v^*, s^*, p^*\}$ that makes up the core sentence structure. From the selected structure \mathcal{T} , a sentence is generated based on several pre-defined rules and templates. Similarly, Fang et al. [5] develop a set of word detectors to detect words from given images. Their word detectors are built to detect all the words in the vocabulary that can be used to describe the given image. The next step is to arrange all the detected words to form a complete sentence. A language model is trained to estimate the probability of a word w_l conditioned on the preceding words w_1, w_2, \dots, w_{l-1} , as well as the set of words with high likelihood detections that have not been chosen. They use a beam search decoder for the generation process and then re-rank the candidate sentences.

Other systems generate reports and texts from a database such as [16], [7], ILEX [31] and M-PIRO [11]. Their domains are varied from museum exhibit descriptions [31] [11] to educational reports [3] [35], weather forecasts and weather reports [16] [6] [34] [19], technical documentations [36], sport news and summaries [8] [1] [19], and nursing and clinical reports [7] [10] [33]. Table 2.2 is an example of Konstas’s data-to-text generation system [19] applied to the weather forecast domain.

Although the results from previous work are promising and some proven to be better than human-generated content, they still have limitations. Most current approaches are based on very specific rules or grammars. In the domain of weather forecasts, Ramos-Soto et al. define a set of operators for each weather prediction variables (e.g., sky state, wind, temperature, etc.) [34]. Each of these operators is responsible for generating partial linguistic descriptions for given variables. Then, pre-defined templates or rules are applied to generate the completed sentences. Similarly, GameRecapper — an NLG system that generates soccer game summaries in Portuguese — uses pre-constructed templates to apply to the input data and generate output sentences [1]. To compose these rules and templates, expert knowledge is usually required. Therefore, adapting these systems to a different dataset or domain usually requires re-designing the entire system.

On the other hand, data-driven techniques are applicable to different domains [23, 2, 17, 19]. These approaches define probabilistic models that can be trained to learn the patterns and hidden alignments between data and text, thereby avoiding the construction of rules and grammars that require domain-specific knowledge. Liang et al. [23] present an approach modeling the hidden correspondence alignment between a world state and its textual description. Built from the work of Liang et al., Angeli et al. [2] propose a unified content selection and surface realization generation system. Their generation process is a sequence of local decisions. They first choose which records in the database to talk about, then which fields of those records to mention, and finally which words to use to describe the chosen fields. Even though their generation model is domain-independent, in order to guarantee fluent output, they apply domain-specific constraints in the template extraction steps of the surface realization component. Konstas recasts Liang et al.’s alignment model into a set of context free grammar (CFG) rules that capture the inherent structure of the input [19]. Then, he treats the generation process as a parsing problem using the pre-defined CFG rules. His generation model does not require domain-specific knowledge, therefore, it is applicable to different domains. To ensure the fluency of output, he intersects his model with external linguistically motivated models including a language model and a dependency model.

2.2 Automatic tweet generation

Recent work in automatic tweet generation focuses on the tasks of automatic text summarization and topic classification, thus, it can be categorized as text-to-text NLG.

With the goal of generating tweet messages recapping the contents of government documents, Loli and Krestel [28] present an NLG system utilizing different text processing techniques including content grouping, topic classification and text summarization. Their system first obtains documents from the city’s open-data archives and stores them in an internal database. Then, each document is analyzed, classified, and linked to previously

Input	Temperature				Cloud Sky Cover	
	Time	Min	Mean	Max	Time	Percent (%)
	06-21	32	39	46	06-21	75-100
	Wind Speed				Wind Direction	
Time	Min	Mean	Max	Time	Mode	
06-21	6	7	10	06-21	SE	
Output	Mostly cloudy, with a high near 46. South southeast wind between 6 and 10 mph.					

Table 2.2: Example input and output of Konstas’s system in weather forecast domain [19].

obtained documents. The expected “interestingness” of each document is also predicted based on its type, content, and relation to other documents. Any documents classified as “interesting” are then chosen to be posted on to Twitter. Each tweet is a short summary of the original document with a link to the full document. The hash-tags are also generated automatically using a Support Vector Machine model in order to address the topic of the original document for easily grouping and searching.

Lloret and Palomar [27] apply a similar idea — using text summarization frameworks — in the task of generating Twitter headlines for journal articles. They construct their own corpus containing articles from 10 pre-selected English and Spanish newspaper online websites. They keep only the body of each article and remove other unnecessary information. They also collect the tweets associated with each article in the corpus. Each tweet consists the headline of the article and the link to it. Then, they use six state-of-the-art text summarizers to generate summary sentences for each article. Any sentences that are longer than a tweet’s limit of 140 characters are removed. Finally, the tweets generated by different systems are evaluated in terms of quantity (using ROUGE metrics [25]) and quality (performing a user survey). Their study compares different frameworks and shows that text summarization systems can produce indicative tweets — they can recap the main ideas of original documents, however, these tweets might not be interesting from a human perspective.

Instead of generating full tweet sentences, Krokos and Samet [20] utilize several sentiment analysis and classification methods in their approach to automatically discover and generate hash-tags for tweets that do not have user-generated hash-tags. First, tweets are collected from a pre-selected set of users. Words are extracted from tweets and their associated websites if the tweets contain links to websites. Different features are computed including TF-IDF (Term frequency - Inverse Document Frequency), sentiment, emotion and geo-location (physical location of the tweet). TF-IDF is computed for each extracted word. Details of the computation are described in [20]. The value of TF-IDF indicates the

importance of the word. The idea is that the more frequently the word appear, it is likely more important. However, the score will be penalized if that word appears in different documents since it could be a general word (e.g., “the”). Using extracted features, they train different classifiers in order to assign appropriate hash-tags to tweets.

In contrast to previous works based on text summarization techniques, Sidhaye and Cheung conclude that applying extractive summarization methods to generate tweets could be a limitation [39]. They first collect tweets that have links to the original articles. Then, text and information from both tweets and their articles are extracted. They calculate different metrics and statistics between the contents of the tweets and their articles such as exact match, percentage match and longest common subsequence match. The results of their experiments show that only a portion of the tweets can be recovered from the articles they link to.

2.3 Summary

In the previous sections, we provided background on NLG systems including some examples of previously developed systems in two main categories: text-to-text and data-to-text. Then, we discussed previous studies related to our work – automatic tweet generation. Current work in the domain focuses on generating tweets summarizing a document (e.g., government documents or journal articles); thus, can be categorized as a text-to-text system. Our work focuses on another aspect where tweets are constructed from structured data, and in our case the data can come from a real-time web application. As discussed, our generation task belongs to the category of data-to-text, also known as concept-to-text or as linguistic description of data generation.

Chapter 3

Traffic data

In this chapter, we introduce the task definition. With the task definition in hand, we explore and discuss how different available traffic-related datasets can or cannot be used to train the system.

3.1 The traffic task definition

A data entry \mathbf{d} consists of a set of records $\mathbf{r} = \{r_1, r_2, \dots, r_n\}$. Each record r_i is described with a record type $r_i.t$, $1 \leq i \leq |\mathbf{r}|$, and a set of fields \mathbf{f} . Each field $f_j \in \mathbf{f}$, $1 \leq j \leq |\mathbf{f}|$, has a field type $f_j.t$ and a value $f_j.v$.

A scenario in the training corpus is a pair of (\mathbf{d}, \mathbf{w}) where \mathbf{w} is the text describing the data entry \mathbf{d} . Figure 3.1 is an example of a scenario in the traffic incident domain. In this scenario, \mathbf{d} is a data entry about a traffic incident with 8 records having 6 possible record types: **Main road**, **Reference road name**, **Lane**, **Condition**, **Reason** and **Incident Type**. There are 2 records having record type **Lane** and 2 records with record type **Incident Type**. Record type **Main road** is described through 2 fields: **Name** and **Direction** with values “401” and “Eastbound” respectively. Other record types only have one field associated with each of them. The textual description \mathbf{w} of \mathbf{d} is a tweet mentioning that traffic incident.

Our goal is to train a model that represents the hidden alignments between data entry \mathbf{d} and the observed text \mathbf{w} in the training corpus. Then, the trained model that captures the alignment is used to generate text \mathbf{g} from a new entry \mathbf{d} which is not contained in the training corpus.

Data d	Main road		Reference road name	Lane	
	Name	Direction	Value	Value	
	401	EB	James Snow Pkwy	Centre lanes	

Lane	Condition	Reason	Incident Type	Incident Type
Value	Value	Value	Value	Value
Collectors	Bad traffic	Collision	Debris	Accident

Text **w** COLLISION: #hwy401 eb b/w James Snow & scales #milton: 2 vehicles blocking centre lane. Debris on roadway

Figure 3.1: Example of a scenario in the traffic incident domain. Data entry **d** consists of 8 records belonging to 6 record types: **Main road**, **Reference road name**, **Lane**, **Condition**, **Reason** and **Incident Type**. Record type **Main road** is described through 2 fields: **Name** and **Direction** while other record types only have one field associated with each of them. The textual description **w** of **d** is a tweet mentioning that traffic incident.

3.2 Traffic-related Datasets

There are various types of traffic-related data including traffic flow, traffic incidents, road construction and road closures. Such data is usually available through different map and road navigation APIs. Tom Tom Traffic¹, Google Maps² and Bing Maps³ are a few examples of map APIs that provide traffic flow data showing vehicles' speeds on road segments. These data are often used in navigation devices to help users avoid traffic congestion. Besides the popular and widely used map APIs, there are also other parties which collect and develop their own data; however, these data are usually limited within a city or a region. For example, Translink's Real-time Traffic Map⁴ provides real-time traffic status for the Metro Vancouver Area, in British Columbia, Canada or CBC's Toronto Live Traffic Map⁵ provides similar information for the Metro Toronto Area, in Ontario, Canada. Road construction, road closures and traffic incidents data are published through governments' open datasets. However, they are usually historical data and not available for real-time applications.

Despite the wide availability of traffic-related data, most of the data are only useful for visualization purposes since they lack the corresponding textual descriptions. A few of the data sources have a short description associated with each data entry such as the Dublin

¹<http://developer.tomtom.com/>

²<https://developers.google.com/maps/>

³<https://www.bingmapsportal.com/>

⁴<http://www.translink.ca/en/Getting-Around/Driving/Real-Time-Traffic-Map.aspx>

⁵<http://www.cbc.ca/toronto/features/traffic/>

City Council’s road works and maintenance dataset ⁶ and Bing Maps’ Traffic Incidents ⁷. However, the textual description is either more detailed than the data entry or not detailed enough to cover all essential information from the data entry. For instance, Figure 3.2 is an example entry in the Dublin City Councils’ road works and maintenance dataset. The textual description (Work Description) of the data entry is too short and does not describe other essential information in the data entry such as location and time of the resurfacing work. On the other hand, Figure 3.3 shows a record in Bing Maps’ Traffic Incidents where the description contains information that cannot be inferred from the data entry such as the street names.

X Co-ordinate:	310,089.810
Y Co-ordinate:	233,321.270
__id:	40
Asset Type:	Roadway
Final Actual Area:	
Completion Date:	2011-03-12T00:00:00
Title:	Kylemore Rd/Walkinston Ave
Status of Works:	Complete
Contractors Name:	Siac Bituminous Products Ltd
Item Type:	Item
Road Classification:	Traffic Impact 4
Path:	roadsandtraffic/WorksProg/Lists/Works Programme Listing KYLEMORE ROAD BALLYFERMOT DUBLIN 10; #2211;
Roads Listing:	#WALKINSTOWN AVENUE WALKINSTOWN DUBLIN 12; #3950
Electoral Area:	South East
Re-Surfacing Materials:	Bituminous;#2
Surface Type:	Bituminous;#2
Work Description:	Resurfacing

Figure 3.2: Example record in the Dublin City Council’s road works and maintenance dataset. The Work Description field does not describe essential information such as where and when the resurfacing work was.

3.3 CVST Dataset

The CVST project is an open platform that supports smart transportation applications [42] by providing Application Programming Interfaces (APIs) for different traffic-related datasets of the greater Toronto area including traffic cameras and sensors, road closures and incidents, public transportation and tweets. We use two datasets collected from the CVST APIs, road incidents and Twitter traffic reports, to construct our first corpus. The other corpus is constructed from the newly developed version of the Twitter traffic report

⁶<https://data.dublincity.ie/dataset/roads-maintenance-annual-works-programme>

⁷<https://msdn.microsoft.com/en-us/library/hh441726.aspx>

```

{
  "__type": "TrafficIncident:http://schemas.microsoft.com/search/local/ws/rest/v1",
  "point": {
    "type": "Point",
    "coordinates": [38.85135, -94.34033]
  },
  "congestion": "",
  "description": "M0-150 is closed between 5th Ave S and Court Dr - construction",
  "detour": "",
  "end": "\\Date(1310396400000)\\",
  "incidentId": 210546697,
  "lane": "",
  "lastModified": "\\Date(1309391096593)\\",
  "roadClosed": true,
  "severity": 3,
  "start": "\\Date(1307365200000)\\",
  "type": 9,
  "verified": true
}

```

Figure 3.3: Example record in the Bing Maps’ Traffic Incidents. Street names included in the description cannot be inferred from the data entry.

API. Details about how we collect and construct the corpora are provided in the following sections.


3.3.1 Road incidents

The road incidents dataset has details about traffic incidents such as time, location, type and reason. Each road incident has a short textual description describing the incident. Figure 3.4 shows two examples of data records from the road incidents dataset. The bold texts at the top of each example are the textual description of the structured data below.


Even though there are textual descriptions associated with every data record, this dataset by itself is not ideal for training the models. In the first record (Figure 3.4a) the information in the textual description can be easily inferred from the structured data. On the other hand, in the second record (Figure 3.4b), the textual description contains no information included in the structured data and vice versa. In addition, the textual descriptions in the road incidents data seem to follow a pre-defined template. Therefore, additional information in the structured data, **Lanes Affected** in the first record, will never appear in the textual description.

3.3.2 Twitter traffic reports

The Twitter traffic report dataset contains basic information about the incident and its related tweets. The tweets are collected from different Twitter users and analyzed by a credit-based evolutionary algorithm to identify trustable data sources [42]. Figure 3.5

 **401 Westbound Express [HWY 404 -] - Disabled Vehicle**

Event Id: 377428
Highway: 401
Direction: Westbound
Stream: Express
From/At: HWY 404
To:
Lanes Affected: LEFT SHOULDER AND 1 LEFT LANE(S)
Comment :
Traffic Impact :
Type/Reason Unplanned - Disabled Vehicle
Event Start: 09/22/16 01:03:00
Event End: 09/22/16 01:09:00
Last Change: 09/22/16 01:09:00

 **All lanes closed due to police investigation.**

Event Id: 43097
District: Scarborough
Road Type: Collector
Main Road: SCUNTHORPE RD
From Road: MILNER AVE
At Road: INVERGORDON AVE
Comment : '
Urgent Emergency
Event Start: 09/21/16 18:58:00
Event End: 09/21/16 21:58:00
Last Change: 09/21/16 21:58:00

(a) In this example, information in the textual description **can** be inferred from the structured data.

(b) In this example, information in the textual description **cannot** be inferred from the structured data.

Figure 3.4: Examples of data records from the road incident dataset.

shows two examples of data records from the Twitter traffic report dataset. As shown, the information about road incidents included in the records is limited with only basic fields such as type of incident, traffic condition and affected lanes. Therefore, using the Twitter traffic report dataset in conjunction with the road incident dataset described above could establish a reasonable corpus for training purposes. In the next section, we explain how we match records from the two datasets to construct a corpus of road incidents and tweets.

start: 16/09/22 16:20
Last updated: 16/09/22 16:20
Incident: accident
Condition:
Lanes: right_lanes

Toronto Traffic Guy 16:20: OPP reporting a crash 404 SB just past Finch in the right lane. I'm seeing traffic bunching up there. #TorontoTraffic(ID=779052662796853248)
Mike Philp 16:20: OPP reporting a crash 404 SB just past Finch in the right lane. I'm seeing traffic bunching up there. #TorontoTraffic(ID=779052662746521600)

(a)

start: 16/09/22 14:45
Last updated: 16/09/22 15:40
Incident: disabled_vehicle
Condition:
Lanes: left_lanes

Ari Rabinovitch 14:45: 401 WB before Mississauga rd there's a stalled vehicle in the LL(ID=779028757885575168)
Ari Rabinovitch 15:40: 401 WB at Mississauga road CLEAR(ID=779042729753911296)

(b)

Figure 3.5: Examples of data records from the Twitter traffic report dataset.

3.3.3 Duo corpus: Matching road incidents with tweets

The Twitter traffic report dataset lacks essential information about incidents such as street names and directions. Therefore, we need to collect this missing information from the road incident dataset. By matching times and locations of records in the two datasets, we construct a parallel corpus of traffic incidents with their related tweets. However, the times and locations from the two datasets are not always exactly matched. Therefore, we allow variations when matching these values. We consider two incidents from two datasets to be matched if:

- the events' locations are within 100 meters from each other,
- and the events' start times are within 90 minutes of each other

Note that we can choose different values for the above criteria. However, relaxing or tightening the conditions could potentially match the records incorrectly. In fact, we compute how long each incident lasts using the event start and event end field in the data and find out that 95% of the incidents in the dataset last less than 90 minutes.

Figure 3.6 shows an example of a record in our corpus constructed from the two datasets. On the left column, we have two records received from the two APIs: traffic incident (on the top) and Twitter traffic report. Even though both events happen in the same location (Highway 401 and Morningside Avenue), the values of longitude and latitude in the two records are not matched. Using those values, we are able to compute the approximate distance between the two locations. In this example, the two locations are approximately 25 meters away from each other. Similarly, using the “eventStart” field from the traffic incident record and the “starttime” field from the Twitter traffic report record, we know that the two records start within 16 minutes of each other. Since the criteria are satisfied, the two records are matched into a single record as shown on the right column.

The data is collected from January 2015 to May 2016. There are 27,795 records in the road incident dataset and 13,134 records with 13,667 tweets in the Twitter traffic report dataset (some records have more than one associated tweet). After matching the two datasets using the described rules, we have a corpus of 1,388 incidents and 2,829 tweets. The tweets are crawled from Twitter and generated by both humans and machines. For the rest of the thesis, we use the term “Duo corpus” to refer to this corpus.

3.3.4 Single corpus: New version of Twitter traffic reports API

In addition to the above corpus, we collect another dataset from a newer version of the CVST's Twitter traffic report API. The new API is not yet published through the CVST portal. However, with the assistance of the author, we are able to access the new API and construct another corpus. We call this corpus the “Single corpus” for all remaining sections in the thesis.

```

Traffic incident record

{
  "type": "unplanned",
  "longitude": -79.196896,
  "latitude": 43.796284,
  "highway": "401",
  "direction": "Eastbound",
  "referenceRoad": "MORNINGSIDE AVE",
  "lanes": "LEFT SHOULDER AND 1 LEFT LANE(S)",
  "stream": "Collector(s)",
  "reason": "collision",
  "eventStart": "2015-02-02 20:33:00",
  "eventEnd": null,
  "lastUpdated": null
}

```

```

Twitter traffic report record

{
  "longitude": -79.1972079,
  "latitude": 43.7962881,
  "incident": ["accident"],
  "lanes": ["collectors"],
  "condition": ["tow_truck"],
  "starttime": "2015-02-02 20:49:00",
  "endtime": "2015-02-02 20:59:00",
  "tweets": [
    "COLLISION: #Hwy401 EB collectors ramp to Morningside Ave: Single vehicle into the ditch",
    "COLLISION: #Hwy401 EB collectors at Morningside Ave: Single vehicle into the right ditch, tow truck at scene."
  ]
}

```

```

Matched record

{
  "Incident": {
    "Conditions": ["tow_truck"],
    "Reason": "collision",
    "Type": ["accident"]
  },
  "Location": {
    "Direction": "Eastbound",
    "Lanes": ["collectors"],
    "Main road": "401",
    "Reference road": "MORNINGSIDE AVE"
  },
  "Twitter messages": [
    "COLLISION: #Hwy401 EB collectors ramp to Morningside Ave: Single vehicle into the ditch.",
    "COLLISION: #Hwy401 EB collectors at Morningside Ave: Single vehicle into the right ditch, tow truck at scene."
  ]
}

```

Figure 3.6: Example of matching two records from the traffic incident dataset and Twitter traffic report dataset. The left column shows two individual records before matching. Since both events' start times are within 16 minutes of each other and the locations are 25 meters away from each other, they are matched into a single record on the right column.

```

{
  "truncated": false,
  "text": "QEW CLEAR ---- EB - past CAWTHRA - stalled vehicle off to the RSH
          (police & tows on scene w/ flashing lights)",
  "analysis": {
    "confidence": "precise",
    "lane_effects": {
      "right_shoulder": ["RSH"]
    },
    "locations": {
      "confidence": "precise",
      "main": {
        "name": "qew",
        "dir": ["EB"]
      },
      "geo": {
        "lat": "43.5827969",
        "lon": "-79.5838117"
      },
      "cross": {
        "name": "Cawthra Road"
      }
    },
    "incidents": {
      "incident_end": ["CLEAR"],
      "disabled_vehicle": ["DISABLE"]
    },
    "conditions": {
      "EMS": ["POLICE"],
      "tow_truck": ["TOW"]
    },
    "rpt_type": "incident"
  },
  "tweet_id": "649533333139947520",
  "user.name": "Trafficnet.ca",
  "timestamp": "Thu, 01 Oct 2015 10:36:29 -0000",
  "user.screen_name": "TRAFFIC_Toronto",
  "user.label": "media"
}

```

Figure 3.7: Example of results returned from the new API.

The data source of the new corpus is not different from the two datasets above. However, it provides easier access to the data and the analysis results of the credit-based algorithm [42]. As shown in Figure 3.7, each data entry returned from the new API contains a pair of traffic incident information and its related tweet. Therefore, it is not necessary to apply the matching method described in the previous section. In addition, each entry is included with the analysis results of the credit-based algorithm that tells whether the accuracy of information is “precise”, “medium” or “low” (in the “confidence” field of the result). Moreover, the information about the traffic incident in the new API is more detailed with some additional fields and values that are not supported in the older API such as “EMS” and its values including “POLICE”, “EMS” and “FIRE_CREWS”.

In order to prevent duplicated tweets (tweets that are retweeted from main sources), we only collect tweets from major users such as “OPP GTATraffic”, “OntarioRoads”, “680NEWSTraffic”, “TRAFFIC Toronto”, “TO DVP”, “TRAFFIC Toronto” and “TO Gardiner LS”. We also keep only records with high confidence (more trustable) from the results analyzed by the credit-based algorithm. Overall, we collected data from September 2015 to June 2016 consisting of around 11,000 records.

3.3.5 Data pre-processing

Before using the data provided from the CVST APIs to construct our corpora, we first scan through the data to remove duplicated tweets. We also remove extra words and tokens in the tweets that cannot be mapped to any records or fields in the data entries. The idea is to reduce the alignment noise which could be generated by the additional information in the tweet that does not appear in the structured data. We remove hyperlinks and tagged tweet usernames (tokens begin with the “@” character).

We also re-structure the data \mathbf{d} returned from the APIs into set of records and fields. The data structures of the Duo corpus and the Single corpus are shown in Table 3.1 and Table 3.2 respectively.

3.4 Summary

In this chapter, we formalized the task of tweet generation from structured data about traffic incidents and provide notations used throughout the rest of the thesis. Then, we investigated different types of traffic-related datasets currently available. We explained the reason why most of these datasets are not suitable to use for our task. Finally, we provided information about the CVST APIs and how we constructed our corpora from these APIs. We have two corpora: the Duo corpus constructed by matching records from the traffic incident API and Twitter traffic report API and the Single corpus using data collected from the new version of the Twitter traffic report API.

Record types	Fields	Example values
Main road	Name	“401”, “400”, “QEW”, ...
	Direction	“NB”, “SB”, “WB”, “EB”
Reference road	Name	“Keele Street”, “Markham Road”, “Dufferin Street”, ...
Lanes	Value	“right_lanes”, “left_lanes”, “centre_lane”, ...
Stream	Value	“collector(s)”, “collector off-ramp”, “express”,...
Condition	Value	“bad_traffic”, “road_open”, “road_closure”...
Reason	Value	“collision”, “disabled_vehicle”, “emergency_roadwork”...
Type	Value	“problem”, “debris”, “accident”...

Table 3.1: The structure of data **d** in the Duo corpus.

Record types	Fields	Example values
Main road	Name	“401”, “400”, “QEW”, ...
	Direction	“NB”, “SB”, “WB”, “EB”
Reference road name	Value	“Morningside Avenue”, “Markham Road”, “Salem Road”, ...
Lanes	Name	“right_lanes”, “left_lanes”, “centre_lane”, ...
	Count	1, 2, 3
Condition	Value	“bad_traffic”, “road_open”, “tow_truck”...
Incident	Value	“accident_crash”, “disabled_vehicle”, “debris”,...

Table 3.2: The structure of data **d** in the Single corpus.

Chapter 4

Traffic notification system

We design a traffic notification system having a location-based user model to predict a user's routes and deliver real-time notifications if traffic incidents occur, and a content-preference model to deliver only information that the user prefers. Figure 4.1 shows the design of our proposed system, parts of which we have implemented and evaluated in Chapter 5. The GPS location of a user is processed through the location-based user model. It predicts a ranked list of routes and destinations the user could take. Concurrently, live stream of traffic incidents is collected and forwarded to the location-relevant information filter. This component applies a location filter on the traffic incident data based on the predicted user's routes and destinations. The output is the data scenarios of traffic incidents that happen on or nearby the routes the user may take. Simultaneously, based on the user's preferences, the content-preference model forwards the corresponding parameters to the generation model. Next, using the parameters given by the content-preference model and the traffic incident information, the generation model composes short messages describing nearby traffic incidents. Finally, these messages are sent to users as textual or speech notifications using a text-to-speech system.

In the following sections, we describe how the generation model is constructed using a data-driven approach. In the scope of this thesis, we only implemented the generation model in order to automatically generating tweets about traffic incidents. Note that the other two models have not yet been implemented. However, we provide an overview discussion of previous work on location-based user models and how they can be applied to our traffic notification system. Finally, we consider different issues with the content-preference model and explain why it should be a separate component in our system.

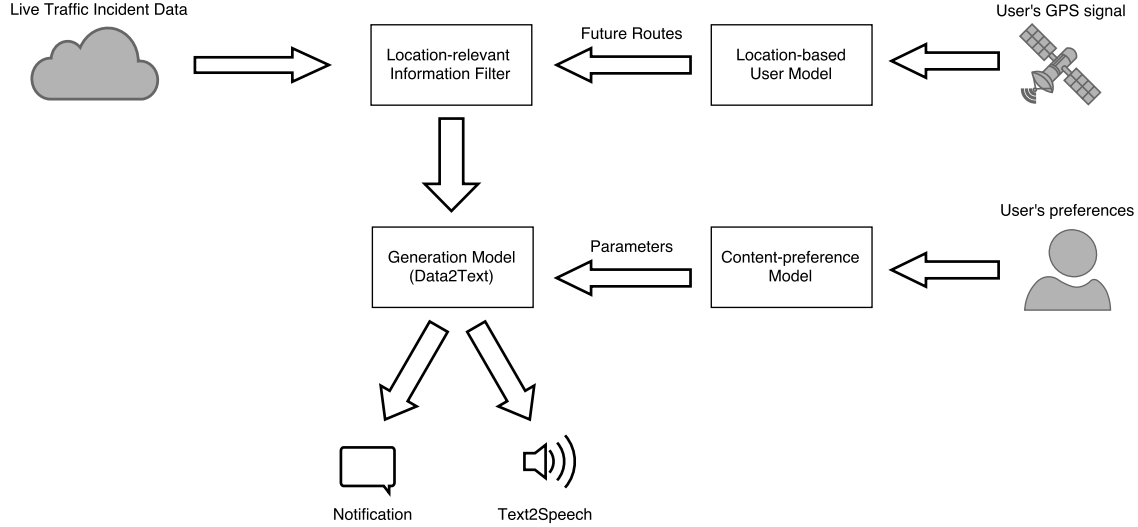


Figure 4.1: Design of the traffic notification system that provides location-relevant traffic information for road users.

4.1 The alignment model

Liang et al. [23] introduce a hierarchical semi-Markov model to learn the correspondences between a world state and an unsegmented stream of text. Their approach is a generative process with three main components:

- Record choice: choose a sequence of records $\mathbf{r} = (r_1, \dots, r_{|\mathbf{r}|})$ where each $r_i \in \mathbf{d}$ and has a record type $r_i.t$. The choice of consecutive records depends on their types.
- Field choice: for each chosen record r_i , select a sequence of fields $\mathbf{f}_i = (f_{i1}, \dots, f_{i|\mathbf{f}_i|})$ belonging to that record.
- Word choice: for each chosen field f_{ij} , choose a number $c_{ij} > 0$ and generate a sequence of c_{ij} words.

The words emitted by chosen fields and records are concatenated to generate the textual description of the world state. In our application, a world state is structured data describing a traffic incident. Each traffic incident has one or more associated textual descriptions (tweets).

The processes of choosing which records, fields and words described above is modeled as a hierarchy of Markov chains. The graphical model presentation is shown in Figure 4.2. The first layer of the model is the record choice, a Markov chain of records conditioned on record types: given a record type, the next record is chosen uniformly in the set of records with that type. The intention is to capture the two types of regularities in the discourse

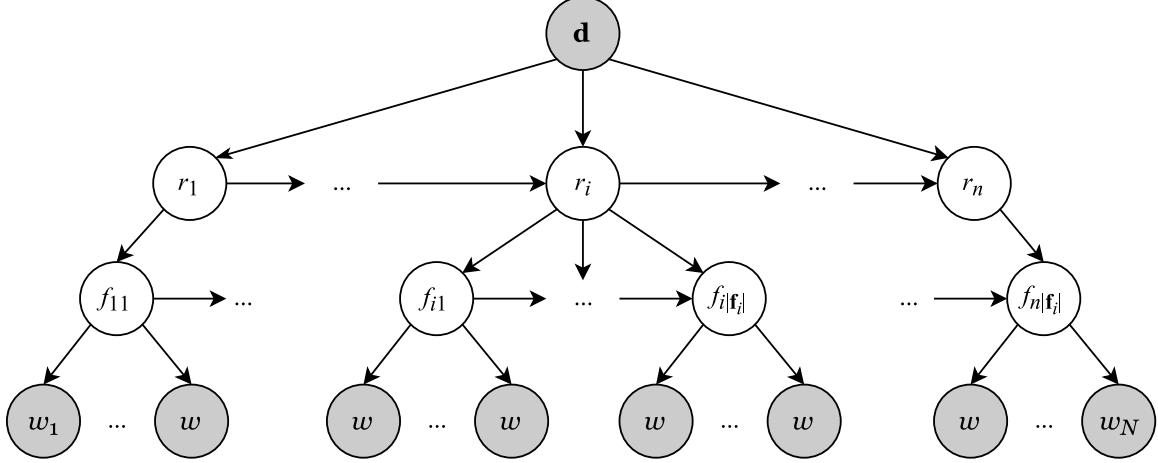


Figure 4.2: Graphical model representing Liang et al.’s alignment model. Records are chosen and ordered from \mathbf{d} . Then, fields are chosen for each record. Finally, words are chosen for each field. The set of records \mathbf{d} and the words \mathbf{w} are observed, while $(\mathbf{r}, \mathbf{f}, \mathbf{c})$ are latent variables to be inferred [23].

structure of language, salience and coherence [23]. Formally:

$$p(\mathbf{r} \mid \mathbf{d}) = \prod_{i=1}^{|\mathbf{r}|} p(r_{i.t} \mid r_{i-1.t}) \frac{1}{|\mathbf{s}(r_{i.t})|} \quad (4.1)$$

where $\mathbf{s}(r_{i.t})$ is the set of records in \mathbf{d} that has record type $r_{i.t}$ and $r_{0.t}$ is the START record type. The model also includes a special NULL record type responsible for generating text that does not belong to any record types in the structured data \mathbf{d} . Analogously, the field choice model is a Markov chain of fields conditioned on the choice of records:

$$p(\mathbf{f} \mid \mathbf{r}) = \prod_{i=1}^{|\mathbf{r}|} \prod_{j=1}^{|\mathbf{f}_i|} p(f_{ij} \mid f_{i(j-1)}) \quad (4.2)$$

Two special fields — START and STOP — are also implemented to capture the transitions at the boundaries of the phrases. In addition, each record type has a NULL field aligned to words that refer to that record type in general. The final step of the process is the word choice model where words are generated from the choice of records and fields. Specifically, for each field f_{ij} , we generate a number of words c_{ij} , chosen uniformly. Then the words \mathbf{w} are generated conditioned on the field \mathbf{f} .

$$p(\mathbf{w} \mid \mathbf{r}, \mathbf{f}, \mathbf{c}, \mathbf{d}) = \prod_{k=1}^{|\mathbf{w}|} p_w(w_k \mid r^{(k)}.t_{f^{(k)}}, r^{(k)}.v_{f^{(k)}}) \quad (4.3)$$

where $r^{(k)}$ and $f^{(k)}$ are the record and field responsible for generating word w_k at position k , determined by the segmentation \mathbf{c} : $\sum_{i=1}^{|\mathbf{r}|} \sum_{j=1}^{|\mathbf{f}_i|} c_{ij} = |\mathbf{w}|$; and $p_w(w_k \mid t, v)$ is the

distribution of words given a field type t and field value v . The model supports three different field types. Depending on the field types, Liang et al. define different methods for generating words:

- Integer type: generate the exact value, rounding up, rounding down and adding or subtracting unexplained noise ϵ_+ or ϵ_-
- String type: generate a word chosen uniformly from those in the field value
- Categorical type: maintain a separate multinomial distribution over words for each field value in the category.

An example output of the model for the text “Gardiner Expressway EB at Don Valley Parkway: left lane blocked due to collision. Police on scene.” is shown in Figure 4.3. The top row contains the records in the structured data \mathbf{d} selected by the model. The subscripts next to each record type name indicate the record tokens belong to that record type. In this example, there are two records belonging to record type **condition**: **condition₁** and **condition₂**. The second row contains selected fields for each record and their associated values. The special field NULL is mapped to words that are not directly referred to the values of fields and records in the data, such as “at”, “:”, “due to”, “.” and “on scene.”. Finally, the last row contains the segmentation and alignment of text produced by the model.

Records:	main_road₁			ref_road_name₁	
Fields:	name=gardiner	direction=eb		value=dvp	
Text:	Gardiner Expressway	EB	at	Don Valley Parkway	:
Records:	lane₁	condition₁	incident₁		
Fields:	value=left_lanes	value=bad_traffic_blocked		value=accident_collision	
Text:	left lane	blocked	due to	collision	.
Records:	condition₂				
Fields:	value=ems_police				
Text:	Police	on scene.			

Figure 4.3: Example output of the alignment model for a tweet about traffic incident.

Liang et al.’s alignment model [23] is able to find the alignment between the text and the structured data. However, it is not designed for actual text generation itself. The generation model that we are going to present in the next section applies the idea of the discussed alignment model to generate texts given structured data.

4.2 The generation model

Konstas [19] recasts an earlier model [23] into a set of context-free grammar (CFG) rules. To capture word-to-word dependencies during the generation process, he added more rules to

G_{CS}	1. $S \rightarrow R(start)$	$[Pr = 1]$
	2. $R(r_i.t) \rightarrow FS(r_j, start) R(r_j.t)$	$\left[P(r_j.t \mid r_i.t) \frac{1}{ S(r_j.t) } \right]$
	3. $R(r_i.t) \rightarrow FS(r_j, start)$	$\left[P(r_j.t \mid r_i.t) \frac{1}{ S(r_j.t) } \right]$
	4. $FS(r, r.f_i) \rightarrow F(r, r.f_j)FS(r, r.f_j)$	$[P(f_j \mid f_i)]$
	5. $FS(r, r.f_i) \rightarrow F(r, r.f_j)$	$[P(f_j \mid f_i)]$
	6. $F(r, r.f) \rightarrow W(r, r.f)F(r, r.f)$	$[P(w \mid w_{-1}, r, r.f)]$
	7. $F(r, r.f) \rightarrow W(r, r.f)$	$[P(w \mid w_{-1}, r, r.f)]$
G_{SURF}	8. $W(r, r.f) \rightarrow \alpha$	$[P(\alpha \mid r, r.f, f.t, f.v, f.t = cat, null)]$
	9. $W(r, r.f) \rightarrow gen(f.v)$	$[P(gen(f.v).mode \mid r, r.f, f.t = int) \times P(f.v \mid gen(f.v).mode)]$
	10. $W(r, r.f) \rightarrow gen_str(f.v, i)$	$[Pr = 1]$

Table 4.1: Grammar rules used for generation with their corresponding weights.

emit a chain of words, rather than words in isolation. Table 4.1 shows his defined grammar rules with their corresponding weights. There are two groups of rules, G_{CS} and G_{SURF} . The first group is the set of rules responsible for content selection while the latter one is the set of surface realization rules.

The first rule, rule (1), in the grammar expands from a start symbol S to a special START record $R(start)$. Then, the chain of two consecutive records, r_i and r_j is defined through rule (2) and (3). Their weight is the probability of emitting record type $r_j.t$ given record $r_i.t$ and corresponds to the record choice model of Liang et al. [23]. Equivalently, rule (4) and (5) define the chain of two consecutive fields, f_i followed by f_j , and their weight corresponds to the field choice model. Rule (6) and (7) are added to specify the expansion of the field to a sequence of words. Their weight is the bigram probability of the current word given its previous word, the current record and field. Finally, rules (8)-(10) are responsible for generating words. If the field type is categorical (denoted as *cat*) or NULL (denoted as *null*), rule (8) is applied to generate a single word α in the vocabulary of the training set. Its weight is the probability of seeing α , given the current record, field and *cat* or *null* field type. Rule (9) is applied if the field type is integer (denoted as *int*). $gen(f.v)$ is a function that accepts the field value (an integer) as its input and return an integer using one of the six methods described by Liang et al. [23]. $P(gen(f.v).mode \mid r, r.f, f.t = int)$ is the multinomial distribution over the six integer generation function choices given the record field f . $P(f.v \mid gen(f.v).mode)$ is the geometric distribution of noise ϵ_+ and ϵ_- if adding or subtracting noise generation functions are chosen. It equals 1 otherwise. The weight of rule (9) is the product of $P(gen(f.v).mode \mid r, r.f, f.t = int)$ and $P(f.v \mid gen(f.v).mode)$. In words, we first choose which method to generate the integer value, then generate that value according to the chosen method. Rule (10) is responsible for generating a word for

Items:	$[A, i, j]$ $R(A \rightarrow B)$ $R(A \rightarrow BC)$
Axioms:	$[W, i, i + 1] : s \quad W \rightarrow g_{i+1}, g_{i+1} \in \{\alpha, \text{gen}(), \text{gen_str}()\}$
Inference rules:	$(1) \quad \frac{R(A \rightarrow B) : s[B, i, j] : s_1}{[A, i, j] : s \cdot s_1}$ $(2) \quad \frac{R(A \rightarrow BC) : s[B, i, k] : s_1[C, k, j] : s_2}{[A, i, j] : s \cdot s_1 \cdot s_2}$
Goal:	$[S, 0, N]$

Table 4.2: The basic decoder deductive system.

string-type field. $\text{gen_str}(f.v, i)$ is a function that simply returns the i^{th} word of the string in the field value $f.v$.

Figure 4.4 is an example of applying the defined grammar rules for the scenario in Figure 4.3. We only show the partial tree for the trimmed text in that scenario — “Gardiner Expressway EB at Don Valley Parkway :”. Note that there are other derivation trees for the given text, the tree shown in Figure 4.4 is just one of them.

After defining the grammar rules, Konstas [19] treats the generation problem as a parsing problem using the CFG rules. He uses a modified version of the CYK algorithm [14, 48] to find the best text \mathbf{w} given a structured data entry \mathbf{d} . His basic decoder is presented as a deductive proof system [38] in Table 4.2. In his basic decoder, items take two forms: $[A, i, j]$ — indicating a non-terminal A span from position i to j in the generated text — and $R(A \rightarrow B)$ or $R(A \rightarrow BC)$ — corresponding to any rules in content selection production rules G_{CS} that have one or two non-terminals on the right hand side. Axioms are generated words using surface realization rules G_{SURF} . The inference rules follow two forms: one for grammar rules with one non-terminal on the right hand side and one for grammar rules with two non-terminals on the right hand side. For example, in Table 4.2, the first inference rule (1) combines two items, production rule $R(A \rightarrow B)$ with weight s and a generated span $[B, i, j]$ — non-terminal B spans from i to j — with weight s_1 to create a new span $[A, i, j]$ — non-terminal A spans from i to j — with weight $s \cdot s_1$. Similarly, the second inference rule (2) combines three items, rule $R(A \rightarrow BC)$ with weight s , two smaller generated spans $[B, i, k]$ with weight s_1 and $[C, k, j]$ with weight s_2 to create a larger span $[A, i, j]$ with weight $s \cdot s_1 \cdot s_2$. The goal of the system is the special item $[S, 0, N]$ where S is the root node of the grammar and N is the length of the generated text.

The decoding process works in a bottom-up fashion. It starts with choosing N — the length (number of words) of the output text. We will discuss how to choose value N in a later section. Then, for each position i in the output text, it searches for the best scoring

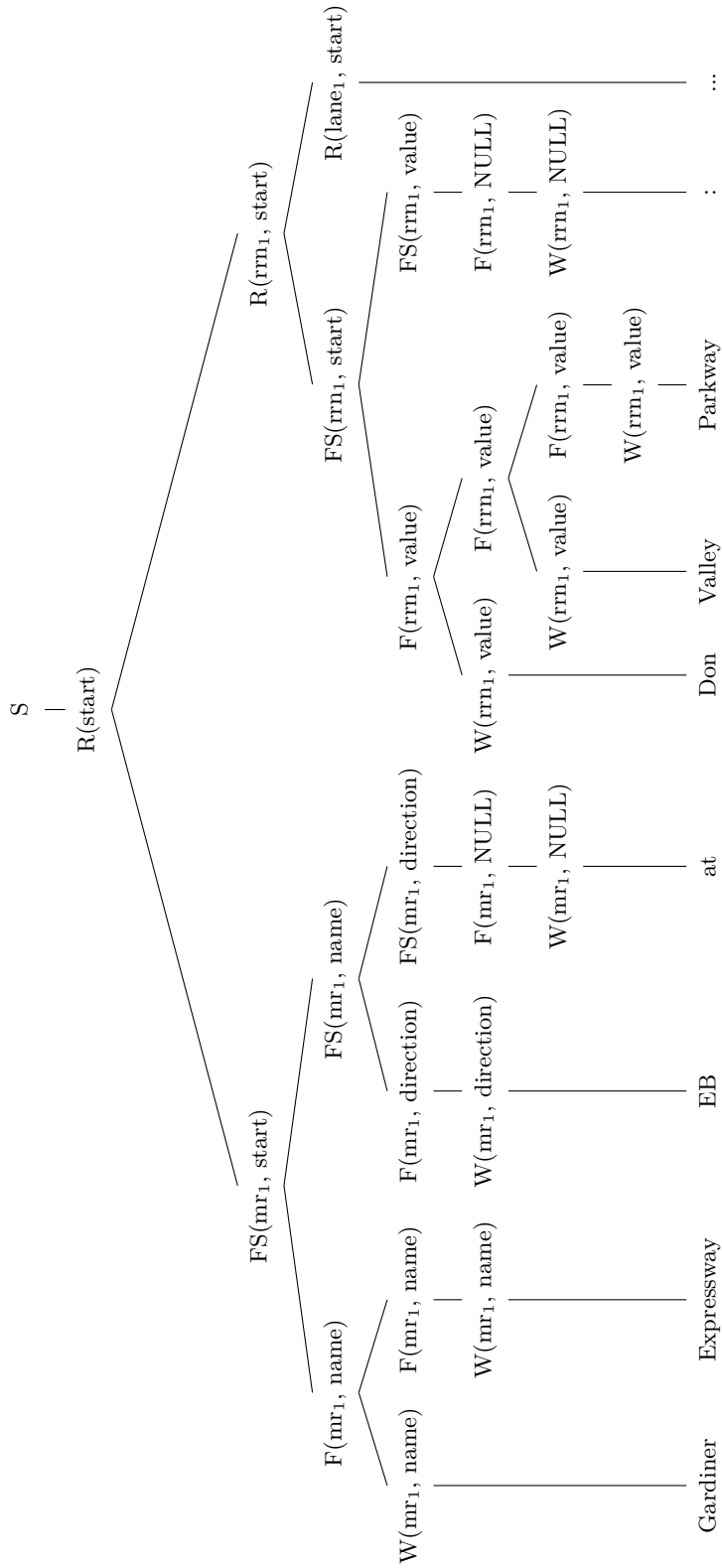


Figure 4.4: One of the derivation trees using the grammar in Table 4.1 for the text “Gardiner Express EB at Don Valley Parkway”. In this figure, we use “mr” as a shorthand for “main_road” and “rrn” as a shorthand for “reference_road_name”.

```

1: function DECODE( $G_{GEN}$ ,  $\mathbf{d}$ ,  $N$ )
2:   for  $i \leftarrow 0 \dots N$  do
3:     for all  $r \in \mathbf{d} : W \leftarrow g_{i+1} \in G_{SURF}$  do
4:        $chart[W, i, i + 1] \leftarrow [W, i, i + 1] : s$ 
5:        $bp[W, i, i + 1] \rightarrow g_{i+1}$ 
6:     end for
7:   end for
8:   for  $l \leftarrow 2 \dots N$  do
9:     for all  $i, k, j$  so that  $j - i = l$  and  $i < k < j$  do
10:    for all items  $[B, i, j]$  or  $[B, i, k]$ ,  $[C, k, j]$  inferrable from  $chart$  and rules  $r \in$ 
     $G_{CS}$  do
11:      if  $r$  is of the form  $A \rightarrow B$  then
12:         $chart[A, i, j] \leftarrow \max(chart[B, i, j] \times P(r))$ 
13:         $bp[A, i, j] \leftarrow \operatorname{argmax}([B, i, j] : s_1 \times P(r))$ 
14:      end if
15:      if  $r$  is of the form  $A \rightarrow BC$  then
16:         $chart[A, i, j] \leftarrow \max(chart[B, i, k] \times chart[C, k, j] \times P(r))$ 
17:         $bp[A, i, j] \leftarrow \operatorname{argmax}([B, i, k] : s_1 \times [C, k, j] : s_2 \times P(r))$ 
18:      end if
19:    end for
20:  end for
21: end for
22: end function

```

Figure 4.5: Konstas’s Viterbi search for the basic decoder

item that spans from i to $i + 1$ (one single word). Next, items are visited and combined in order for larger spans until it reaches the goal item $[S, 0, N]$ — symbol S spans from position 0 to N . The search process is implemented as a Viterbi search as shown Figure 4.5. Lines 2-7 are the first step of the decoding process where the best scoring items spanning from i to $i + 1$ for each single position in the output are chosen. Lines 8-21 are the next step where smaller spans are grouped into larger spans until it reaches the largest possible spans (from 0 to N). Lines 11-14 group items according to any rules with one non-terminal on the right hand side, following the form $A \rightarrow B$ while lines 15-18 group items according to rules with two non-terminals on the right hand side, following the form $A \rightarrow BC$.

The basic decoder always chooses the best scoring item during the parsing process. Konstas [19] extends the basic decoder with the k-best decoder in which a list of k-best derivations will be kept for each item. The extension significantly improves the output quality by avoiding local optima. He also intersects the grammar rules with a tri-gram language model and a dependency model to ensure fluency and grammaticality of the output text. Details of his decoders are explained in [19].

As it stands, the grammar rules in Konstas [19]’s generation model are defined to naturally capture the structure of the input data. Therefore, they can be applied to any data that is structured from the sets of records and fields. In Konstas [19]’s experiments, he ran

his model on different datasets belonging to different domains: weather forecast (Weather-Gov), robot soccer game commentaries (RoboCup), air travel (ATIS) and troubleshooting (WinHelp). Therefore, we can also apply this model to different traffic data in different cities without changing the model structure.

4.3 Location-based user model

There has been a wide range of work on location-based user models, learning and predicting users' routes and destinations. These tasks involve some uncertainties. Much work relies on GPS signal data to identify a user's location and may not be accurate. In addition, intended destinations are not always certain since they may be affected by factors such as weather, traffic, day of week, and time of day. Due to many uncertainties arising from the tasks, most systems build probabilistic models to identify and predict users' locations.

The comMotion system [29, 30] uses pattern recognition techniques to learn users' patterns of traveling and frequent destinations. These frequent locations can be added or removed manually by users using the graphical or speech interface. Their algorithm to detect a frequent location follows a simple idea. GPS signal is usually lost within a building. Therefore, the system recognizes locations where GPS signal is lost. After losing signal within a given radius on three different occasions, the agent infers that this must be a building and marks it as a salient location. It is later prompted to be confirmed by the user if this is a frequent location. Each frequent location is assigned to a to-do list which is displayed or reminded whenever the user travels to this location. Items in to-do lists are not necessary texts and can be recorded audio.

Krumm et al. [21] present a model that predicts the destinations and future routes drivers take. In their work, the map is modelled as a directed graph where road intersections are vertices and road segments connecting these intersections are edges. From a start location, the candidate destinations are all road intersections reachable within 60 minutes of driving. Their goal is to compute a destination probability of each candidate intersection – the higher the probability means the more likely the destination is. They develop a probabilistic model that estimates the destination probability of a candidate given the current trip (all the intersections have been passed). After collecting a list of candidate destinations and their probabilities, route probabilities are computed by summing all destination probabilities along the fastest routes. With this approach, a corpus of the driver's regular routes is not necessary since the prediction is assessed on a per trip basis. Figure 4.6 is an example of the system prediction results. The vehicle's trip starts in the lower right of the map with the complete trip shown as the black line. The intersections along the partially travelled route are shown as grey dots. The most recent passed intersection is occurring about halfway along the complete trip. The small shaded dots show the candidate destinations. Their transparency is inversely proportional to their probability. Most of the

destinations have been essentially eliminated, because the driver has passed up multiple opportunities to drive to them. The remaining destination candidates are grouped ahead of the driver around the remainder of the trip.

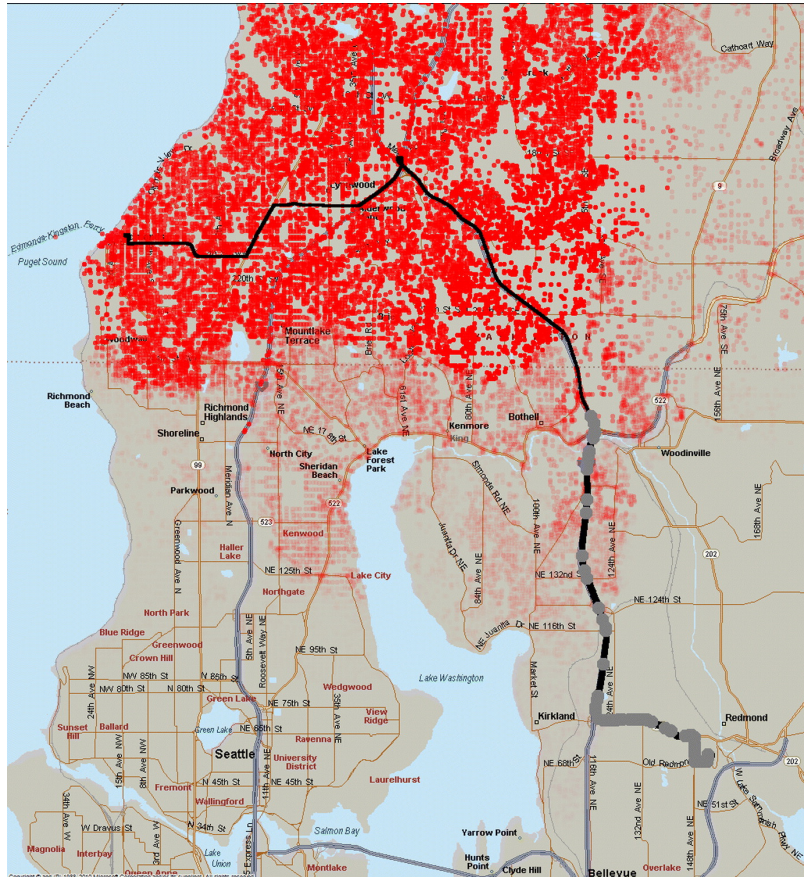


Figure 4.6: An example of Krumm et al. system’s prediction results [21]. The start location is in the lower right of the map. The black line shows the complete trip, and the grey circles show the intersections along the partial travelled route. Destinations with higher probabilities are shaded darker.

Simons et al. [40] also use the same technique as Krumm et al. [21] to model the road map. They present a basic Hidden Markov Model (HMM) with states being pairs $s = \langle l, g \rangle$ where l is a link (a directed edge connecting two intersections in the road map) and g is a goal (destination). The transition probability from state $s_j = \langle l_j, g_j \rangle$ to $s_i = \langle l_i, g_i \rangle$ is defined as: $p(s_i | s_j) = p(g_i | l_i) \times p(l_i | s_j)$. In other words, given the current state, we predict the next state by first predicting the next link the driver will go to, then predict the next destination given that link. However, the basic model does not take into account other factors such as day of week or time of travel. They extend the basic decoder by considering those factors into each state $s = \langle l, g, f_1, f_2, \dots, f_n \rangle$ where f_i is the i^{th} additional factor.

Liao et al. [24] developed a more sophisticated model with the ability to infer the user’s mode of transportation. The user’s activities are estimated by a Bayesian network model with three levels. The lowest level estimates the person’s location (including person’s location and the location of the person’s car), velocity and transportation mode from GPS measurements. The transportation mode can take on four different values {BUS, FOOT, CAR, BUILDING} and influences the motion velocity. The middle level represents the person’s next goal or predicting destination and the current trip segment. A trip segment is defined by the start and end location, and the mode of transportation user takes during that segment. Finally, the highest level, denoted the novelty mode, indicates whether the user is currently behaving normally, doing something novel, or making a transportation error.

Despite differences in the design and the complexity of the tasks, current location-based user models have one common goal of predicting users’ future routes and ultimately, the application of location-relevant information delivery. With a user’s location or GPS signal as the input, these models can predict the user’s next locations or routes. As in Figure 4.1, the predicted future routes are forwarded to a location-relevant information filter. This component simply checks whether there are traffic incidents occurring on or nearby the user’s future routes and send this information to the generation model. It can be more advanced with the ability to rank the impacts of the traffic incidents and filter out minor incidents that do not affect travel times or route choice decision. For example, if there are two incidents close to each other (e.g., happening on the same road segment) and both cause traffic congestion, the driver might only need to know about one of them since the knowledge of the another one might not affect the driver’s decision to take an alternative route.

4.4 Content-preference model

Besides the provision of location-relevant information, it is more helpful if a user can select which types of traffic information should be used in the notifications. For example, detour information is useful for a driver who does not know about the roads in the area. It could be however considered distracting from a perspective of a commuter who knows every street in the neighborhood. In that circumstance, the commuter might be only interested in the location and the cause of the incident since he or she can decide which alternative routes to take using his or her own knowledge about the area. The content-preference model is responsible for selecting which information should be notified to the users based on their profiles and preferences. It is actually the content selection step in an NLG system. In Konstas’s generation model [19], the step is integrated into the grammar rules, therefore, it is not obvious to implement such features into his model. In this section, we will discuss different issues and solutions for the content-preference model.

We first start with a simple solution to the problem using the same idea of Konstas where he intersects external models (i.e., language model and dependency model) to the grammar to ensure fluent output. During the generation process, these external models score the generated texts. Then, the scores are used together with the grammar rule weights to select the best candidates for generation. We can apply the same approach and construct a content preference model that accepts a set of selected records and fields as input and returns a score weighting how much the selected information matches with the user preferences. Then, this score is used to rank the candidates, together with the grammar rule weights and scores from other external models. However, this method also has some limitations. First, since the generation process works in a bottom-up fashion, the model can only score the generated sentence at later steps of the process where emitted words and fields are already selected and formed into records. Therefore, undesired information is not pruned out during the early steps and could use up spots of more potential candidates in the search space. Second, if we use cube-pruning to reduce the complexity of the decoding algorithm, we have the assumption that the scoring function is a monotonic function [4]. However, adding the score of the content-preference model into the grammar could potentially make the assumption invalid.

A more acceptable approach is to look at the grammar rules and modify them to support the content-preference model. As in Table 4.1, Konstas’s CFG contains two separate groups: G_{CS} for content-selection and G_{SURF} for surface-realization. If we integrate the content-preference model into rules in G_{CS} group, we may capture user preferences on which records, fields and their orders should be chosen to generate the personalized notifications. Rule (2) and (3) in Table 4.1 are responsible for record choices. We can add another variable u to indicate the user profile into the rule weights as follows:

$$\begin{aligned}
 2. R(r_i.t) &\rightarrow \text{FS}(r_j, \text{start}) R(r_j.t) && \left[P(r_j.t \mid r_i.t, u) \frac{1}{|s(r_j.t)|} \right] \\
 3. R(r_i.t) &\rightarrow \text{FS}(r_j, \text{start}) && \left[P(r_j.t \mid r_i.t, u) \frac{1}{|s(r_j.t)|} \right]
 \end{aligned}$$

With this modification, choosing which record type to mention next is conditioned on the previous record type and the user profile u . Depending on the level of personalisation, we can apply the same modification to other rules such as field choices:

$$\begin{aligned}
 4. \text{FS}(r, r.f_i) &\rightarrow \text{F}(r, r.f_j) \text{FS}(r, r.f_j) && [P(f_j \mid f_i, u)] \\
 5. \text{FS}(r, r.f_i) &\rightarrow \text{F}(r, r.f_j) && [P(f_j \mid f_i, u)]
 \end{aligned}$$

We can also apply the modification to word choices and G_{SURF} rules with the idea that each user may prefer different words and vocabularies to talk about the records and fields. However, this modification also increases the number of parameters of the model, makes it more complicated and could potentially impact the performance. We also need to formalize all the real world factors determining the user preferences (e.g., driving experience, knowledge about the area, etc.) into the variable u .

Inheriting the discussed idea into our system design, we separate the content-preference model from the generation model and turn it into an individual component in order to maintain the simplicity of the generation model. As depicted in Figure 4.1, based on the user preferences, the content-preference model provides the generation model its parameters (CFG rule weights). Then, the generation model uses the provided weights and its pre-defined CFG rules to generate the output texts. By making the content-preference model a separate component, we also allow more advanced control over which records and fields to be chosen based on not only user profile but also other factors such as historical notifications. For example, a traffic incident causing road closures could be sent to the driver. Later on, this incident is resolved and the roads are open again. The driver may want to know about the update of this incident. However, information such as the reason of the incident is no longer of interest to the driver since he already knew about the incident from the last notifications. On the other hand, the driver is only interested in the update status of the roads — from closed to open. If we integrate all the factors affecting the choice of the records and fields into the generation model, it will make the model very complicated with a large number of variables and parameters.

The details of how to build such content-preference model is beyond the scope of this research. Since building the content-preference model requires the collection of additional data, we decided to leave this for future work.

4.5 Summary

We have proposed a design of a traffic notification system that delivers personalized and location-relevant notifications about real-time traffic incidents. The system consists of the generation model, the location-based user model and the content-preference model. We explained how these models work and function in our system. Moreover, we described how the generation model is constructed using a data-driven approach, discussed previous work on location-based user models and considered different issues with the content-preference model.

Chapter 5

Experimental results

5.1 Implementation and training

As discussed in the previous chapter, the scope of this thesis does not include the detail implementation of the location-based user model and the content-preference model since additional training data is required. We implemented the k-best decoder version of Konstas’s generation model since it is shown to outperform the basic decoder in Konstas’s experiments [19]. In addition, a tri-gram language model is intersected with the grammar to ensure fluent output. With this extension, the scores used to rank the candidates in the k-best lists are computed using both grammar rule weights and language model scores. As presented by Chiang [4] and Konstas [19], the running time of the intersection algorithm is too slow to use in practice. Therefore, we follow Chiang’s suggestion [4] and apply cube-pruning in the implementation of the decoder to help reducing the complexity of the algorithm. In our implementation, we stop adding new candidates to the k-best list if we already have enough k candidates or there are no more candidates to consider. Details of how cube-pruning works are explained in [4] and [19]. We also consider adding a dependency model as in Konstas’s implementation, however, we believe that this will not help improve the output quality because of two reasons:

- We are working on tweets in which there is a lot of variability of vocabulary across documents (e.g., use of different shorthand, special characters and tokens). Therefore, using a dependency model trained on a standard corpus is not suitable for the application and may cause inaccurate results.
- In Konstas’s experiments, it is shown that adding the dependency model does not help improve the output quality in most of the cases.

Even though we can address the first issue by using a tweet dependency parser such as [18], from our observation, using the language model alone is good enough to generate fluent

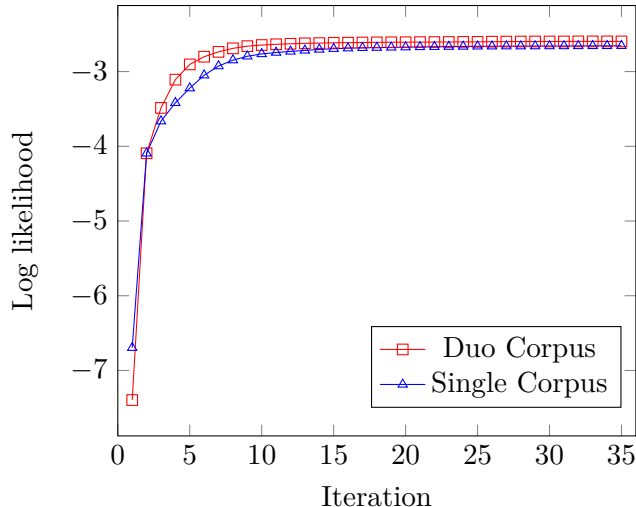


Figure 5.1: Log likelihood of the data in the two corpora during the training iterations using EM algorithm.

and grammatical English texts. The usage of other external models could be investigated further in the future.

In order to learn the grammar weights, we estimate them using an Expectation Maximization (EM) algorithm. Given a training set where each scenario is a pair of structured data \mathbf{d} and its corresponding textual description \mathbf{w} , we maximize the marginal likelihood of our data, summing out the latent variables including records \mathbf{r} and their fields \mathbf{f} :

$$\max_{\theta} \prod_{(\mathbf{w}, \mathbf{d})} \sum_{\mathbf{r}, \mathbf{f}} p(\mathbf{r}, \mathbf{f}, \mathbf{w} \mid \mathbf{d}; \theta) \quad (5.1)$$

where θ are the parameters of the model including all the multinomial distributions and weights of grammar rules. The EM algorithm iteratively switches between the E-step and the M-step. In the E-step, the expected counts for the rules are computed. In the M-step, we optimize θ by normalizing the counts computed in the E-step. We initialize EM with a uniform distribution for each multinomial distribution. The algorithm is run in approximately 35 iterations until the marginal likelihood of data is unimproved. In addition, we also experiment with different initialization settings for the EM algorithm (e.g., initializing with random perturbation around the uniform distribution). The variance of the data marginal likelihood for these experiments is quite small (around 0.006). Figure 5.1 shows the log likelihood of the data in the two corpora during the training iterations using the EM algorithm.

We use the Natural Language Toolkit (NLTK) 3.0¹ for the implementation of the language models. NLTK provides standardized and stable implementations of different utilities and functions widely used in natural language processing tasks. The tweets are

¹<http://www.nltk.org/>

tokenized using the TweetTokenizer² in the NLTK’s tokenizer package. Then we use the utility functions and the ConditionalFreqDist³ class provided by NLTK to construct and encode the conditional probabilities in the language models. We also apply additive smoothing with $\delta = 0.1$ in the language models. See Appendix A for all the functions, classes and utilities we used in NLTK.

To determine N — the length of output text — prior to generation, Konstas implemented a linear regression model that predicts the value for each data entry \mathbf{d} . The input to the regression model is a flattened version of the structured data with features being record-field pairs. We also apply the same idea. In our linear regression model, the feature values are values of the record-field pairs. We use one-hot encoding for record-field values that are not numerical.

To address Twitter’s limitation of tweet length (140 characters), we implement a straightforward solution. First, the best candidate (in the k-best list) returned by the decoder is checked against the requirement. If its length is less than or equal to 140 characters, it is selected as the final output. Otherwise, the next best candidate is selected and verified if it meets the length requirement. The process is repeated until a candidate with the required length is found. However, the process stops early if none of the top 10% of the best candidates in the k-best list satisfies the length requirement⁴. In this case, instead of trying to generate an output sentence with N words, we decrease N and search for the best candidate with length $N - 1$. Note that the decoder works in a bottom-up fashion, therefore, the candidates with length $N - 1$ are easily retrievable from $chart[S, 0, N - 1]$ without running the decoding process again. Analogously, we repeat the tweet length check for the shorter candidates until the final output is found. There are other solutions can be considered to address the issue with tweet length, such as modifying the CFG rules and running the decoding algorithm in a character-based fashion. However, we want to simplify the solution since our ultimate goal is a traffic notification system in which Twitter’s length limitation might not be applied.

5.2 Evaluation metric

Previous work on NLG systems uses different evaluation techniques including common automatic metrics such as ROUGE [25], BLEU [32] or METEOR [22] and human evaluation. Human evaluation usually takes a substantial amount of time and effort to conduct while automatic evaluation methods are cheaper. Some of the automatic metrics are proven to be correlated with human judgement on specific tasks.

²<http://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.casual.TweetTokenizer>

³<http://www.nltk.org/api/nltk.html#nltk.probability.ConditionalFreqDist>

⁴We can repeat the process for all the k-best candidates. However, in order to prevent choosing a bad candidate at the bottom of the list, we stop the process early.

BLEU [32] is widely used in Machine Translation community to evaluate a text against a set of gold standards, usually generated by human. The score ranges from 0.0 to 1.0 (or 0% to 100%) indicating how close the translation sentence is to the reference sentences. BLEU is a precision-based metric in which it compares n-grams of the candidate with the n-grams of the reference translation and counts the number of matches. The higher number of matches designates the better translation. However, instead of using the standard precision measure, BLEU uses a modified precision to prevent cases where the candidate contains repeated words or phrases in the references, resulting in a very high precision but improbable translation. Details about BLEU are provided in [32].

ROUGE [25] stands for Recall-Oriented Understudy for Gisting Evaluation and is usually used to evaluate text summarization system. ROUGE includes different metrics such as ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S and ROUGE-SU. Similar to BLEU, but using recall instead of precision, ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries. ROUGE-L is based on the longest common subsequence (LCS) of the two summaries. The idea is to treat each sentence as a sequence of words. If two sentences have longer LCS, they are more similar. ROUGE-W is extended from ROUGE-L to be able to reward sentences with the consecutive matches in the LCS. ROUGE-S is a skip-bigram co-occurrence measure. Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. ROUGE-S measures the overlap of skip-bigrams between a candidate translation and a set of reference translations. However, there are cases when two sentences are similar but have a ROUGE-S score of 0 (e.g., “police killed the gunmen” and “gunmen the killed police”). To give credit to sentences in these cases, ROUGE-S is extended to ROUGE-SU by adding unigram as a counting unit.

Developed after BLEU to address some of its observed weaknesses, METEOR [22] is based on an explicit word-to-word matching between the candidate translation and one or more reference translations. Given two sentences to compare – a candidate and a reference translation – METEOR creates a mapping between unigrams of the two strings: every unigram in each string maps to zero or one unigram in the other string and to no unigrams in the same string. There are different modules to list all possible mappings between two strings. Each of them uses different criteria to create the mapping. For example, the “exact” module that maps two unigrams if they are same; the “porter stem” module that maps two unigrams if they are the same after they are stemmed using the Porter stemmer (e.g., “computers” maps to both “computers” and to “computer”); or the “WN synonymy” module that maps two unigrams if they are synonyms of each other. The details of how to compute METEOR score from the unigram matches are explained in [22]. Using this approach, METEOR is able to score two similar translations that use synonyms or paraphrases.

Lloret and Palomar [27] and Sankarasubramaniam et al. [37] evaluate their system automatically using ROUGE, while Yatskar et al. [47] use the BLEU score. On the other

hand, Fang et al. [5] evaluate their system using all of the above metrics including ROUGE, BLEU and METEOR. If we look at work most similar to ours, we see that Konstas [19] evaluates his concept-to-text generation system using BLEU and METEOR scores.

The evaluation of automatically generated tweets in our work can be approached from several perspectives. Evaluation in the context of the task outlined in Figure 4.1 can involve human subjects, looking at metrics such as the usefulness of tweets (using rating criteria like those in rating the helpfulness of reviews or comments), and the quality of tweets (involving fluency and readability). Detailed human evaluation of the tweets is beyond the scope of the current research. Our plan is to focus on automated techniques in the evaluation of the automatically generated tweets, given that we have a reference of generated texts. To evaluate our models, we use the BLEU-4 score, a standard evaluation metric used in the Machine Translation community [19]. We use the implementation of BLEU provided by the NLTK. We also consider using METEOR. However, in the application of tweet generation and the domain of traffic incident data, it is necessary to construct an additional paraphrase table of vocabularies used (e.g., “sb” is a synonym of “southbound”, “hwy” is a synonym of “highway”, etc.). Therefore, we leave this for future work.

To evaluate the automatically generated tweets, we split our corpora into training sets and test sets. We use the training sets to train the model and perform the evaluation on the test sets. For each input scenario in the test sets, the trained k-best decoder returns a k-best list of generated texts, ranked by both grammar weights and language model scores. We picked the first result from the k-best results to compute its BLEU-4 score against the reference texts. If the first result is longer than 140 characters, we have an additional process to pick another result (as described in Section 5.1) to evaluate. In addition, we also report the BLEU-4 score for the oracle results. The oracle results are generated texts in the final k-best lists that have the best BLEU-4 score against the their reference texts.

5.3 Experiments

5.3.1 Experiment 1

In the first experiment, we split the corpora into a training set and test set randomly. We use 95% of the data for training and 5% for testing. For the Single corpus, the training set contains 10,785 records of traffic incidents, while the test set consists of 568 records. For the Duo corpus, the training set and the test set contains 1,219 and 68 records respectively. We use different values of k for our k-best decoder: $k = 10$, $k = 20$ and $k = 50$. We also tried other values of k , however, the scores do not improve with $k > 50$. We run the experiment three times and report the average scores of three times. Table 5.1 shows the results of our experiment. As discussed in Section 5.2, we report two types of scores: the BLEU-4 score for the one-best results (the top results in the k-best lists) and the oracle results (generated texts that have highest BLEU-4 scores against their reference texts).

Value of k	Duo corpus		Single corpus	
	One-best	Oracle	One-best	Oracle
10	10.01	10.17	15.96	16.3
20	9.79	10.42	16.99	17.7
50	10.76	11.04	17.39	18.34

Table 5.1: Results (BLEU-4 scores) of experiment 1.

With higher k , we are able to get better scores. However, the overall scores are not high. There are various reasons, mainly because of the inaccurate alignments between the structured data and the tweets caused by noise in the training dataset.

First, each Twitter user has a different writing style. For example, “680 NEWS Traffic” mentions the direction of the main road followed by its name (e.g., “EB 401”) while “OPP GTA Traffic” expresses the same information in the opposite way (e.g., “#Hwy401 EB”). Moreover, they also have different vocabularies to describe the traffic incidents. “OPP GTA Traffic” uses hash-tags for the main road names while “680 NEWS Traffic” does not. “680 NEWS Traffic” and “OPP GTA Traffic” use abbreviations for road direction (e.g., “SB” for “southbound”) while “Ontario Roads” and “Traffic Toronto” use the full forms. Table 5.2 shows some example tweets from different users. The EM algorithm updates the parameters to maximize the marginal likelihood of the data. Therefore, depending on which writing style the majority of tweets belong to, utilizing the learnt parameters could result in generated tweets with the combination of all the writing styles or some writing styles that are dominating in the training data. On the other hand, since the scenarios in the test set do not always have multiple reference tweets from different users, the candidates and their references could belong to different writing styles. Evaluating these tweets against their references in different writing styles makes their BLEU scores lower.

Users	Example tweets
680 NEWS Traffic	<p>SB DVP ramp to York Mills is closed due to a collision. Emerg on scene.</p> <p>COLLISION: EB 401 in the express, East of Islington. Left lane blocked, slow from approaching Kipling. More on the ones!</p> <p>Problems on the WB 401 ramp to NB 400 - A collision is blocking the top of the ramp but you can still access the NB 400 from the WB 401.</p>
OPP GTA Traffic	<p>COLLISION: #Hwy401 EB off ramp to Leslie - Single vehicles partially blocking left lane of ramp</p> <p>UPDATE: #Hwy401 WB is CLOSED at Hespler #Cambridge for vehicle removal and debris clearing</p> <p>CLEARED: #Hwy400 NB before Steeles: Vehicle towed from roadway</p>
Ontario Roads	<p>*-CLR-* 401 Eastbound [HWY 427 -] - Other ...</p> <p>*-NEW-* 401 Eastbound COLLECTOR Off-ramp [- LESLIE STREET] - Collision ...</p> <p>*-NEW-* 401 Eastbound Transfer [Eastbound Warden Express to Collector -] - Collision ...</p>
Traffic Toronto	<p>Toronto - northbound Don Valley Pky at Don Mills Rd collision blocking three left lanes - CLEAR</p> <p>Mississauga - eastbound Hwy-407 Etr at 401 stalled vehicle in the right hand lane - CLEAR</p> <p>Toronto - northbound Don Valley Pky at Don Mills Rd collision blocking three left lanes</p>

Table 5.2: Example tweets from different users. Each user has a different style of writing their own tweets.

#	Weight Distribution	Top-5 scoring items
1	$P(r_{i.t} \text{START})$	incident_type, main_road, ref_road, stream, NULL
2	$P(r_{i.t} \text{main_road})$	ref_road, stream, main_road, incident_type, condition
3	$P(r_{i.t} \text{ref_road})$	ref_road, lane, reason, incident_type, NULL
4	$P(f_{i.t} \text{main_road.START})$	direction, road_name, NULL, STOP
5	$P(f_{i.t} \text{main_road.name})$	direction, STOP, name, NULL
6	$P(f_{i.t} \text{main_road.direction})$	STOP, direction, name, NULL
7	$P(\alpha \text{main_road, direction, wb})$	“401”, “wb”, “west”, “/”, “b”
8	$P(\alpha \text{lane, value, right_lanes})$	“right”, “lane”, “the”, “blocking”, “,”
9	$P(\alpha \text{condition, value, ems})$	“on”, “:”, “emergency”, “services”, “and”
10	$P(\alpha \text{incident_type, value, maintenance})$	“to”, “maintenance”, “,”, “crew”, “,”

(a) Duo corpus

#	Weight Distribution	Top-5 scoring items
1	$P(r_{i.t} \text{START})$	main_road, NULL, incident, condition, ref_road_name
2	$P(r_{i.t} \text{main_road})$	ref_road_name, lane, main_road, condition, NULL
3	$P(r_{i.t} \text{ref_road_name})$	ref_road_name, incident, lane, condition, main_road
4	$P(f_{i.t} \text{main_road.START})$	name, direction, NULL, STOP
5	$P(f_{i.t} \text{main_road.name})$	STOP, direction, name, NULL
6	$P(f_{i.t} \text{main_road.direction})$	STOP, direction, name, NULL
7	$P(\alpha \text{main_road, direction, sb})$	“sb”, “southbound”, “#hwy400”, “#hwy427”, “#hwy404”
8	$P(\alpha \text{lane, name, right_lanes})$	“right”, “lane”, “the”, “in”, “hand”
9	$P(\alpha \text{condition, value, ems_police})$	“police”, “:”, “on”, “scene”, “...”
10	$P(\alpha \text{incident, value, maintenance_roadwork})$	“roadwork”, “emergency”, “-”, “...”, “]”

(b) Single corpus

Table 5.3: Top 5-scoring items of the multinomial distributions for some of the grammar rules in the experiment 1.

The second reason is the unbalanced number of records in the corpora. Since the majority of traffic incidents in the corpora happen on highways such as “401”, “400” and “427”, the number of **main_road** record types with these values are overwhelming in the corpus. This introduces noise in the alignment model, especially when it is trained in an unsupervised fashion. Table 5.3 shows examples of the top scoring items of the multinomial distributions for some of the grammar rules trained with the Duo corpus and the Single corpus respectively. In the top 5-scoring items for the multinomial distribution $P(\alpha \mid \text{main_road}, \text{direction}, \text{wb})$ in the Duo corpus (row 7 of Table 5.3a), “401” is the highest scoring item while the correct word – “wb” – is the second highest scoring item. Similarly, for the multinomial distribution $P(\alpha \mid \text{main_road}, \text{direction}, \text{sb})$ in the Single corpus, we also have noise where highway names (i.e., “#hwy400”, “#hwy427” and “#hwy404”) are also in the top 5-scoring item (row 7 of Table 5.3b).

Finally, the amount of information in the structured data **d** is usually limited compared to the information contained in the tweets. Figure 5.2 shows two examples of scenarios where tweets contain more details than the structured data. Detour information is missing in the structured data of both examples. In the second example (Figure 5.2b), there are two traffic incidents are mentioned in the tweets: the first incident happens at Highway 427 and Highway 407, in the middle lane, and the second incident is at Highway 427 and Highway 409, in the right lane. However, the information in the structured data indicates that the traffic incident happens at Highway 427 and Highway 407, in the right lane. Although the alignment model introduces a special NULL record type and NULL field to map to the missing information, the limited and incorrect data also produce noise in learning the multinomial distributions.

Data d	Main road		Reference road name			
	Name	Direction	Value			
	401	EB	Bayview Avenue			
	Lane		Lane		Lane	
	Name	Count	Name	Count	Name	Count
	Express	1	Right lanes	2	Collectors	1
	Condition		Condition		Incident	
	Value		Value		Value	
	Bad traffic - Blocked		EMS - EMS		Accident - Collision	

Text **w** Collision EB 401 east of Bayview express blocks the 2 right lanes. Emerg on scene. Back up from east of the Allen

(a) In this example, detour information included in the tweet is missing in the structured data.

Data d	Main road		Reference road name		
	Name	Direction	Value		
	427	SB	407		
	Lane		Incident		
	Name	Count	Value		
	Right lanes	1	Problem - Problem		

Text **w** 2nd problem SB 427 approaching HWY 407 in the middle lane. And on going problem appr HWY 409 in the right lane. Back up from Zenway!

(b) In this example, the tweet talks about two traffic incidents. However, the structured data only has information about the reference road of the first incident (i.e., “407”) and affected lane of the second incident (i.e., “Right lanes”). Detour information is also missing in the structured data.

Figure 5.2: Examples of scenarios where structured data does not have all information mentioned in tweets.

Figure 5.3 shows two examples of tweets generated by our k-best decoder in experiment 1, with $k = 50$. In the first example (Figure 5.3a), as discussed above, the generated tweet **g** follows a combination of different writing styles from major users in the training data:

“OPP GTA Traffic” that starts the sentence with “TRAFFIC HAZARD:”, uses a hash-tag for the main road name and mentions road direction after road name; and “Ontario Roads” that puts the reference road name between two square brackets (i.e., “[” and “]”). However, there is only one reference tweet \mathbf{w} by “680 NEWS Traffic” for this scenario. The reference tweet mentions road direction before road name and does not use a hash-tag for the main road name. In this example, even though the generated text \mathbf{g} might look good in a human perspective, the BLEU-4 score is only 18.81.

The second example (Figure 5.3b) is the case where the decoder generates incorrect information because of alignment noise. We extract some of the examples in Table 5.3b and put them into Table 5.4 to help the reader easily follow the discussion. As discussed above, the number of traffic incidents on southbound Highway 427 and Highway 400 are significantly larger than other locations. Therefore, the alignment model “thinks” that words such as “#hwy427” and “#hwy400” should be aligned to the **direction** field of **main_road** record (as shown in row 4 of Table 5.4). This leads to the inaccurate alignments of the higher levels — the order of fields in **main_road** record. According to the weight distribution of $P(f_i.t \mid \text{main_road.START})$ in Table 5.4 (row 1), to talk about **main_road** record, we can start with the **name** field or **direction** field. However, when we choose either one of them, the next field should be a **STOP** field (shown in the distributions in row 2 and 3 of Table 5.4). It means that the decoder prefers to talk about either the **direction** field or the **name** field when mentioning a **main_road** record. In this example, it chooses to talk about the **direction** field and emits two words “#Hwy427” and “SB” given the value of the field is “SB”. Even though in the candidate sentences, “#Hwy400” is also chosen, the score is slightly lower than “#Hwy427”.

#	Weight Distribution	Top-5 scoring items
1	$P(f_i.t \mid \text{main_road.START})$	name, direction, NULL, STOP
2	$P(f_i.t \mid \text{main_road.name})$	STOP, direction, name, NULL
3	$P(f_i.t \mid \text{main_road.direction})$	STOP, direction, name, NULL
4	$P(\alpha \mid \text{main_road, direction, sb})$	“sb”, “southbound”, “#hwy400”, “#hwy427”, “#hwy404”

Table 5.4: Top 5-scoring items of the multinomial distributions for some of the grammar rules in the experiment 1 extracted from Table 5.3b.

Overall, training the model parameters in an unsupervised fashion with a training corpus with inaccurate and missing data could leads to several issues as discussed. To clarify our assumptions, we perform another experiment. Details of the second experiment are discussed in the next section.

Data d	Main road		Reference road name	
	Name	Direction	Value	
	401	EB	Warden Avenue	
	Lane		Lane	
	Name	Count	Name	Count
	Express	1	Right lanes	1
	Condition		Condition	Incident
	Value		Value	Value
	Bad traffic - Blocked		EMS - EMS	Fire - Fire

Generated **g** TRAFFIC HAZARD: #Hwy401 EB at Warden Ave -] - vehicle in the right lane. Emerg on scene.

Reference **w** UPDATE: Vehicle fire EB 401 approaching Warden express, blocking the right lane - emerg on scene.

(a) Generated tweet **g** following different writing styles from majority Twitter users such as “OPP GTA Traffic” and “Ontario Roads” is evaluated against reference tweet **w** of “680 NEWS Traffic”.

Data d	Main road		Reference road name	
	Name	Direction	Value	
	400	SB	Steeles Avenue	
	Condition		Incident	
	Value		Value	
	Bad traffic - Blocked		Disabled vehicle - Disabled	

Generated **g** TRAFFIC HAZARD: #Hwy427 SB - at Steeles Avenue -] - disabled vehicle.

Reference **w** TRAFFIC HAZARD: #Hwy400 SB at Steeles Avenue - disabled dump truck blocking the on ramp. MTO advised.

(b) Because of alignment noise, the k-best decoder generates tweet with incorrect information — “#Hwy427” instead of “#Hwy400”.

Figure 5.3: Examples of tweets generated by our k-best decoder in experiment 1 with $k = 50$.

5.3.2 Experiment 2

In the second experiment, instead of using all scenarios in the corpora, we train the model using only scenarios with tweets from specified users. We choose two users, “680 NEWS Traffic” and “Ontario Roads”, in this experiment. These users have the majority of tweets in our corpora. Table 5.5 shows the number of tweets of some major users in our corpora.

Users	Duo corpus		Single corpus	
	Number of tweets	Percentage	Number of tweets	Percentage
OPP GTA Traffic	90	2.6%	6132	35.9%
680 NEWS Traffic	779	22.8%	3484	20.4%
Ontario Roads	413	12.1%	2073	12.1%
TRAFFIC Toronto	233	6.8%	3057	17.9%

Table 5.5: The number of tweets of some majority users and their percentage in our corpora.

As shown in Table 5.2, tweets written by “Ontario Roads” follow a pre-defined template and are likely automatically generated by machine using a template-based approach. The tweets begin with a status indicating whether the traffic incident is newly occurred (e.g., “*-NEW-*)” or already resolved (e.g., “*-CLR-*)”). After that, main road names and directions are mentioned, followed by reference names. The reference road names are placed between a pair of square brackets (e.g., “[HWY 427 -]”). Finally, other information about the incidents such as incident type, reason or condition is included and separated with the location by a dash (i.e., “-”). On the other hand, “680 NEWS Traffic” does not have a specific template for writing the tweets. They can start with mentioning the location or the type of the incidents. In addition, the tweets usually contain more information about the traffic incidents than those included in the structured data.

		Duo corpus	Single corpus
680 NEWS Traffic	Training set	418	2774
	Test set	22	146
Ontario Roads	Training set	249	1397
	Test set	13	74

Table 5.6: Number of scenarios in training set and test set for each corpus in experiment 2

Similar to experiment 1, we split the filtered corpora into a training set and test set randomly. The number of scenarios in the training set and test set of the two filtered

corpora in experiment 2 are shown in Table 5.6. We run the experiment three times with $k = 50$ and report the average score in Table 5.7.

		One-best	Oracle
Duo corpus	680 NEWS Traffic	26.26	29.12
	Ontario Roads	65.04	70.86
Single corpus	680 NEWS Traffic	15.8	18.45
	Ontario Roads	57.36	65.08

Table 5.7: Results (BLEU-4 scores) of experiment 2.

As expected, since the tweets written by “Ontario Roads” follow a specific template, it is easier for the alignment model to learn the parameters maximizing the data’s marginal likelihood. With a more accurate alignment and parameters, the k-best decoder is able to generate tweets similar to the original ones. Note that the amount of information provided in the structured data and tweets written by “Ontario Roads” is almost equivalent, therefore, producing less noise when training the model in an unsupervised fashion. On the other hand, usually including additional information and being written using different sentence structures, tweets by “680 NEWS Traffic” are more unpredictable. This explains the lower score in the experiment with tweets by “680 NEWS Traffic”. With the Duo corpus, we are able to get a higher score than the Single corpus. However, the quality of generated tweets is not actually better. In fact, the score is higher because we only have a small amount of scenarios in the test set of the Duo corpus (only 22 scenarios).

Data \mathbf{d}	Main road		Reference road name	Incident
	Name	Direction	Value	Value
	401	WB	410	Debris - Debris
	Lane			
	Name	Count		
	Collectors	1		

Generated \mathbf{g} *- * 401 westbound collector (s) [hwy 410 -] - Debris ...

Reference \mathbf{w} *-NEW-* 401 westbound collector off-ramp [- hwy 410] - Debris ...

Figure 5.4: Example of a generated tweet in experiment 2 with training data extracted from the Single corpus containing tweets from “Ontario Roads” only. We use $k = 50$ in this experiment.

Examples of generated tweets in experiment 2 are shown in Figure 5.4 and 5.5. In Figure 5.4, when the model is trained with only tweets by “Ontario Roads”, it easily captures the structure of the template and is able to generate a tweet \mathbf{g} very similar to the reference \mathbf{w} . The extra information about the status of the incident – whether it is newly occurred or already resolved – cannot be generated since it is not included in the structured data \mathbf{d} . In the case of training with tweets by “680 NEWS Traffic”, Figure 5.5a is an example where the k-best decoder successfully generate a meaningful tweet. The generated tweet contains all the information provided in the structured data. However, the type of incident (i.e., “collision”) included in the generated tweet is caused by noise in the alignment. Specifically, since most of the traffic incidents in the training data are associated with vehicle collisions, the model thinks that “collision” is a general word of **main_road** record type and aligns word “collision” to the **NULL** field of **main_road** record. Therefore, the word is emitted in this example. In another example shown in Figure 5.5b, we face a different issue with the generated tweet where pieces of information are repeated: the location of the incident (e.g., “SB DVP South of Don Mills”) and the status informing that the incident is resolved (e.g., “collision cleared”). Since the amount of information in the structured data \mathbf{d} is limited (only 4 records) while the predicted output length is too long, the same information has to be chosen repeatedly for generation. In this case, the linear regression model fails to predict the reasonable output length given the structured data \mathbf{d} because tweets by “680 NEWS Traffic” are usually long and contain extra information.

Data d	Main road		Reference road name	Lane	
	Name	Direction	Value	Name	Count
	401	EB	Weston Road	Express	1

Incident	Condition	Condition
Value	Value	Value
Disabled vehicle - Disabled	Bad traffic - Blocked	Tow truck

Generated **g** Collision EB 401 approaching Weston Express, a lane is blocked with a stalled vehicle. Tow on scene

Reference **w** Stalled vehicle EB 401 East of Weston Express blocks the 2nd lane from the right. Tow on scene

(a)

Data d	Main road		Reference road name
	Name	Direction	Value
	DVP	SB	Danforth Avenue

Incident	Condition
Value	Value
Incident end - Clear	Bad traffic - Delay

Generated **g** SB DVP South of Don Mills, collision cleared from the SB DVP South of Don Mills, collision cleared from the SB DVP South of Don Mills

Reference **w** CLEAR NOW... it's gone!! Still a significant delay heading SB DVP to Don Mills as a result

(b)

Figure 5.5: Examples of generated tweets in experiment 2 with training data extracted from the Single corpus, contains tweets from “680 NEWS Traffic” only. We use $k = 50$ in this experiment.

5.3.3 Summary

In the first experiment, we trained and tested the model using all tweets from the two corpora. The resulting BLEU-4 scores ranged from 9.79 to 10.76 with the Duo corpus and from 15.96 to 17.39 with the Single corpus depending on the value of k (Table 5.1).

We analyzed the results and believe that the inaccurate alignment parameters are the main causes of the poor results. Since each Twitter user has different writing styles (e.g., the order of mentioning records and fields), noise is produced during the training process. Noise also originated from the in-equivalence of the amount of information in the structured data and the tweets: tweets usually contains more details about traffic incidents than structured data does. Moreover, by using BLEU as an automatic evaluation metric, we cannot capture the usage of different vocabularies by different users. Therefore, the scores are penalized when the number of reference tweets are limited.

In the second experiment, to reduce the noise affecting the accuracy of the alignment model, we trained and tested the model using only tweets by specific users. Tweets by “Ontario Roads” follow a template and contain sufficient traffic incident details included in the structured data. Therefore, using these tweets to train and test the model eliminated most of the noise and resulted in a better BLEU-4 score, ranging from 57.36 to 65.04 (Table 5.7). On the other hand, tweets by “680 NEWS Traffic” are likely generated by human and do not follow any specific order of mentioning records and fields. They also usually contain more information covered in the structured data. Training and testing using these tweets resulted in a lower BLEU-4 score, ranging from 15.8 to 26.26 (Table 5.7). Even though not directly comparable, our BLEU-4 scores are not far from to Konstas’s results using other different datasets that range from 24.88 to 38.26 [19].

Finally, one issue that appears in both experiments affecting the alignment accuracy is the unbalanced the training data. For example, most of the traffic incidents are vehicle collisions; and most of them happen on major highways such as Highway 401, 400 and 427. Since there are not enough scenarios belonging to other minor values in the training data, the EM algorithm cannot compute the expected counts and probabilities correctly for those records and fields. It makes the model incorrectly align phrases and words in the observed tweets \mathbf{w} to the records and fields in the structured data \mathbf{d} . For example, “collision” is aligned to the **NULL** field of **main_road** record type or “#hwy404” is aligned to the **direction** field of **main_road** record type as discussed in previous sections. The issue can be prevented by using a better training data or manually annotating the training data with the correct alignments.

Chapter 6

Conclusion and Future work

Traffic congestion continues to be a major problem for large cities around the world despite current management efforts. Provision of real-time traffic information helps drivers make better route planning and avoid congestion. It also helps alleviate congested flows by distributing vehicles to alternative routes.

There are different types of traffic related data including traffic flows, road closures, road constructions and traffic incidents. We explored different currently available traffic datasets, each has different structures and information. However, most are suitable for visualization purposes since they lack the linguistic descriptions associated with the data. The CVST APIs provide easy access to real-time traffic incident data in the greater Toronto area together with the tweets mentioning them. We constructed two corpora using different versions of the APIs. Each scenario in the training corpus is a pair of structured data about traffic incidents and its related tweets. The tweets are crawled from Twitter and written by different Twitter users including human-generated and machine-generated tweets.

We proposed a design of a traffic notification system that provides personalized and location-relevant notifications about traffic incidents to road users. Our system consists of the location-based user model, the generation model and the content-preference model. Current research on a location-based user model focuses on the location-relevant information delivery, therefore, can be applicable to our system design. For the generation model, we used and implemented a data-driven approach to automatically generate tweets about traffic incidents since it is applicable to different types of traffic datasets without any domain-specific knowledge and redesigning or reconstructing of rules and templates in the system. In addition, different issues with the content-preference model are discussed. By considering different usage scenarios and applications, we provided explanations to the question of why the content-preference model should be a separate component in our system.

We performed two experiments with the constructed corpora and our implementation of Konstas's generation model [19]. Even though the data from CVST APIs is the most appropriate data we are able to collect, it still has different issues causing noise when training

the alignment parameters. We were able to clarify the causes of this noise and reduce some of it by performing experiment 2 that resulted in the improvement of BLEU-4 scores.

The preliminary results show that it is possible to generate real-time tweets for inclusion in a real-time traffic notification system, using techniques that are otherwise applicable to different domains and datasets. However, further work needs to be done in order to improve the system output. It would be very useful to have a better corpus of traffic data with structured data and associated textual descriptions to train the data-driven approaches. If the model is trained in an unsupervised fashion, the amount of information covered in the structured data and its textual description must be equivalent to prevent alignment noise. Alternatively, manual annotation with the correct alignment could help prevent inaccuracy.

Given the relatively constrained domain, template based models could be used with the data-driven approach introduced in this thesis. A template approach requires a different set of patterns and rules for each traffic data type, but integrating techniques involving semantic role labels [26] may assist in applying the approach to different datasets and different locations.

With respect to the ultimate goal of a personalized traffic notification system in Figure 4.1, we have only addressed the surface of the issues concerning personalized tweets. Further work needs to be done based on the discussion of the location-based user model and content-preference model in order to build a complete personalized traffic notification system. Especially, additional data is required to train and evaluate these user models.

Finally, we need to consider issues related to evaluation techniques. As discussed, the BLEU score is widely used in the community, thus, is a good metric to use to compare different systems. But, it cannot capture the variation of vocabularies and paraphrases. Using METEOR can solve the issue but that requires the construction of the paraphrase table if it is applied to tweets. In addition, human evaluation is also an important aspect because it can determine the usefulness of the system. Conducting human evaluation of the real-time traffic notification system should be considered carefully as we do not want to distract the drivers while they are driving. With the availability of current technology, most of the professional drivers are equipped with tablets or similar touch-screen monitor devices in their vehicles. We can utilize these already equipped systems to establish the evaluation of the automatically generated real-time traffic notifications. The notification can be displayed on the screen or spoken through a text-to-speech system. Participating drivers can quickly vote whether a notification is useful or not by a single action (e.g., touch on the notification to indicate that it is useful or swipe to ignore unhelpful information) without being distracted.

Bibliography

- [1] Joao Pinto Barbosa Machado Aires. *Automatic Generation of Sports News*. dissertation, University of Porto, 2016.
- [2] Gabor Angeli, Percy Liang, and Dan Klein. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10*, pages 502–512, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [3] Kalina Bontcheva and Yorick Wilks. *Automatic Report Generation from Ontologies: The MIAKT Approach*, pages 324–335. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [4] David Chiang. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June 2007.
- [5] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. *CoRR*, abs/1411.4952, 2014.
- [6] Eli Goldberg, Norbert Driedger, and Richard I. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert: Intelligent Systems and Their Applications*, 9(2):45–53, April 1994.
- [7] Catalina Hallett. Multi-modal presentation of medical histories. In *Proceedings of the 13th International Conference on Intelligent User Interfaces, IUI '08*, pages 80–89, New York, NY, USA, 2008. ACM.
- [8] Fahim Hasan. Automatic generation of multilingual sports summaries, 2007.
- [9] Martin Hassel. *Resource lean and portable automatic text summarization*. PhD thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden, 2007.
- [10] James Hunter, Yvonne Freer, Albert Gatt, Ehud Reiter, Somayajulu Sripada, and Cindy Sykes. Automatic generation of natural language nursing shift summaries in neonatal intensive care: Bt-nurse. *Artif. Intell. Med.*, 56(3):157–172, November 2012.
- [11] Amy Isard, Jon Oberlander, Ion Androutsopoulos, and Colin MathesonIsard. Speaking the Users’ Languages. *IEEE Intelligent Systems Magazine*, 18(1):40–45, January-February 2003. Special Issue “Advances in Natural Language Processing”.

- [12] Michael Johnston, Srinivas Bangalore, and Gunaranjan Vasireddy. MATCH: multi-modal access to city help. In *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, pages 256–259, 2001.
- [13] Rong-Chang Jou, Soi-Hoi Lam, Yu-Hsin Liu, and Ke-Hong Chen. Route switching behavior on freeways with the provision of different types of real-time traffic information. *Transportation Research Part A: Policy and Practice*, 39(5):445 – 461, 2005.
- [14] Tadao Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts, 1966.
- [15] KattanLina, HabibKhandker M. Nurul, TazullIslam, and ShahidNadeem. Information provision and driver compliance to advanced traveller information system application: case study on the interaction between variable message sign and other sources of traffic updates in calgary, canada. *Canadian Journal of Civil Engineering*, 38(12):1335–1346, 2011.
- [16] Stephan M. Kerpedjiev. Automatic generation of multimodal weather reports from datasets. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC '92, pages 48–55, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.
- [17] Joohyun Kim and Raymond J. Mooney. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 543–551, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [18] Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archana Bhatia, Chris Dyer, and Noah A. Smith. A dependency parser for tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.
- [19] Ioannis Konstas. *Joint Models for Concept-to-Text Generation*. PhD thesis, University of Edinburgh, UK, 2014.
- [20] Eric Krokos and Hanan Samet. A look into twitter hashtag discovery and generation. In *Proceedings of the 7th ACM SIGSPATIAL Workshop on Location-Based Social Networks (LBSN'14)*, Dallas, TX, Nov 2014.
- [21] John Krumm, Robert Gruen, and Daniel Delling. From destination prediction to route prediction. *J. Locat. Based Serv.*, 7(2):98–120, June 2013.
- [22] Alon Lavie and Abhaya Agarwal. METEOR: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.
- [23] Percy Liang, Michael I. Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 91–99, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

- [24] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5):311 – 331, 2007.
- [25] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10, 2004.
- [26] David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. Generating natural language questions to support learning on-line. *ENLG 2013*, pages 105–114, 2013.
- [27] Elena Lloret and Manuel Palomar. Towards automatic tweet generation: A comparative study from the text summarization perspective in the journalism genre. *Expert Systems with Applications*, 40(16):6624 – 6630, 2013.
- [28] Christoph Lofi and Ralf Krestel. *iParticipate: Automatic Tweet Generation from Local Government Data*, pages 295–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [29] Natalia Marmasse and Chris Schmandt. Location-aware information delivery with commotion. In *Proceedings of the 2Nd International Symposium on Handheld and Ubiquitous Computing*, HUC '00, pages 157–171, London, UK, UK, 2000. Springer-Verlag.
- [30] Natalia Marmasse and Chris Schmandt. A user-centered location model. *Personal Ubiquitous Comput.*, 6(5-6):318–321, January 2002.
- [31] Michale O'donnell, Chris Mellish, Jon Oberlander, and Alistair Knott. ILEX: An architecture for a dynamic hypertext generation system. *Nat. Lang. Eng.*, 7(3):225–250, September 2001.
- [32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.
- [33] François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7):789 – 816, 2009.
- [34] Alejandro Ramos-Soto, Alberto J. Bugarín, Senén Barro, and Juan Taboada. Linguistic descriptions for automatic generation of textual short-term weather forecasts on real prediction data. *IEEE Transactions on Fuzzy Systems*, 23(1):44–57, Feb 2015.
- [35] Alejandro Ramos-Soto, Manuel Lama, Borja Vázquez-Barreiros, Alberto J. Bugarín, Manuel Mucientes, and Senén Barro. Towards textual reporting in learning analytics dashboards. In *2015 IEEE 15th International Conference on Advanced Learning Technologies*, pages 260–264, July 2015.
- [36] Ehud Reiter, Chris Mellish, and John Levine. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9(3):259–287, 1995.
- [37] Yogesh Sankarasubramaniam, Krishnan Ramanathan, and Subhankar Ghosh. Text summarization using wikipedia. *Information Processing & Management*, 50(3):443 – 461, 2014.

- [38] Stuart M. Shieber, Yves Schabes, and Fernando C.N. Pereira. Computational linguistics and logic programming principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(1):3 – 36, 1995.
- [39] Priya Sidhaye and Jackie Chi Kit Cheung. Indicative tweet generation: An extractive summarization problem? In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 138–147. The Association for Computational Linguistics, 2015.
- [40] Reid Simmons, Brett Browning, Yilu Zhang, and Varsha Sadekar. Learning to predict driver route and destination intent. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 127–132, Sept 2006.
- [41] Amanda Stent and Srinivas Bangalore. *Natural Language Generation in Interactive Systems*. Cambridge University Press, New York, NY, USA, 2014.
- [42] Ali Tizghadam and Alberto Leon-Garcia. Application platform for smart transportation. In *Future Access Enablers of Ubiquitous and Intelligent Infrastructures*, pages 26–32. Springer, 2015.
- [43] Khoa Tran and Fred Popowich. Automatic tweet generation from traffic incident data. In *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web*, pages 59–66, 2016.
- [44] Yin-Yen Tseng, Jasper Knockaert, and Erik T. Verhoef. A revealed-preference study of behavioural impacts of real-time traffic information. *Transportation Research Part C: Emerging Technologies*, 30:196 – 209, 2013.
- [45] Marilyn Walker, Steve Whittaker, Amanda Stent, Preetam Maloor, Johanna D. Moore, Michael Johnston, and Gunaranjan Vasireddy. Speech-plans: Generating evaluative responses in spoken dialogue. In *In Proc. of INLG-02*, pages 73–80, 2002.
- [46] Yezhou Yang, Ching Lik Teo, Hal Daumé, III, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11*, pages 444–454, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
- [47] Mark Yatskar, Michel Galley, Lucy Vanderwende, and Luke Zettlemoyer. See no evil, say no evil: Description generation from densely labeled images. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 110–120, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.
- [48] Daniel H. Younger. Recognition and parsing of context-free languages in time n^3 . *Information and Control*, 10(2):189 – 208, 1967.

Appendix A

NLTK Usage

The following table summarizes all the NLTK's functions, classes and utilities with their usage in order to implement the k-best decoder and the evaluation described in our work.

Name	Documentation URL	Usage
TweetTokenizer	http://www.nltk.org/api/nltk.tokenize.html#nltk.tokenize.casual.TweetTokenizer	Tokenize the tweets in the data.
NLTK Utilities - Bigrams	http://www.nltk.org/api/nltk.html#nltk.util.bigrams	Extract bigrams from a sequence of tweets's tokens in order to build the language model.
NLTK Utilities - Trigrams	http://www.nltk.org/api/nltk.html#nltk.util.bigrams	Extract trigrams from a sequence of tweets's tokens in order to build the language model.
ConditionalFreqDist	http://www.nltk.org/api/nltk.html#nltk.probability.ConditionalFreqDist	Encode the conditional probabilities in the language model.
NLTK Translate - BLEU score	http://www.nltk.org/api/nltk.translate.html#nltk.translate.bleu_score.corpus_bleu	Compute BLEU-4 scores in the evaluation.