MSc Thesis Task Description

Csáky Richárd
candidate for MSc degree in Computer Engineering

Open-domain Neural Dialog Modelling Issues and Remedies

Modeling conversation is an important task in natural language processing and artificial intelligence. In the past, methods for constructing chatbot architectures have relied on hand-written rules and templates or simple statistical methods. With the rise of deep learning these models were quickly replaced by end-to-end trainable neural networks. The project has multiple aims. First, the student has to summarize recent advances in open-domain neural dialog modeling, and discuss issues and tasks that still remain to be addressed in the field. Then the student has to experiment with a novel neural network architecture (the Transformer), and apply it to the task of dialog modeling. He also has to analyze some of the current open-domain dialog datasets used for training neural chatbots. Finally, he has to explore data filtering approaches in order to build higher-quality data, and train better conversational models. Specifically, entropy-based data-filtering approaches should be experimented with, in order to eliminate the one-to-many, many-to-one problems in dialog modelling.

Tasks to be performed by the student will include:

- Surveying recent advances and techniques in neural dialog modeling
- Summarizing and discussing issues with open-domain chatbots
- Experimenting with the Transformer model applied to dialog modelling
- Comparing the Transformer model with the standard Seq2seq model
- Analyzing current open-domain dialog datasets and evaluation metrics
- Exploring data-filtering approaches for building higher-quality data
- Evaluating conversational models trained on filtered data

Supervisor at the department: Kovács Ádám
External supervisor:

Budapest, 9 September 2019

Dr. Hassan Charaf
professor
head of department

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

Open-domain Neural Dialogue Modelling Issues and Remedies

Master's Thesis

Author
Richárd Krisztián Csáky

Advisor
Ádám Kovács

December 25, 2022

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Csáky Richárd Krisztián*, szigorló hallgató kijelentem, hogy ezt a diplomaterv meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2022. december 25.

Csáky Richárd Krisztián
hallgató

Kivonat

A konverzációs ágens (chatbot) egy olyan program, mely természetes nyelven képes emberekkel kommunikálni. A beszélgetés modellezése egy fontos feladat a természetes nyelvfeldolgozás és mesterséges intelligencia (MI) területén. Az MI tudományág megszületése óta egy jól működő chatbot létrehozása még mindig az egyik legnehezebb kihívás. A chatbotok sokféle feladatra használhatók, de mindegyik esetében elvárt, hogy megértsék a felhasználó mondandóját és az adott problémához releváns válaszokat generáljanak.

A múlt chatbot architektúrái kézi szabályokra és sablonokra, vagy egyszerű statisztikai módszerekre támaszkodtak. 2015 óta, a mélytanulás (deep learning) elterjedésével ezek a modellek gyorsan felcserélődtek elejétől végéig tanítható neurális hálózatokkal. Manapság a rekurrens enkóder-dekóder modell (Cho et al., 2014) és a Transformer-alapú architektúrák dominálják a dialógus modellezést (Vaswani et al., 2017). Ezek a modellek a neurális gépi fordítás területéről lettek adaptálva, ahol rendkívül jó eredményeket értek el. 2015 óta sokféle változat és kiegészítés a válaszminőség javítására.

E diploma három kontribúciót kíván nyújtani. Először egy részletes irodalomkutatást végzünk, több mint 100 publikációt vizsgálva az utóbbi 5 évből. A neurális háló alapú dialógus modellek nagy adathalmazokon való tanításának paradigmája az evaluációs módszerek hatékonyságáról és az adatminőség hatásáról fontos kérdéseket vet fel. Kísérleteinkkel alátámasztjuk hogy a jelenlegi automatikus metrikák nem effektívek konverzációs célok felmérésére. További két kontribúciónk adathalmazok minőségének javításához és létrehozásához kötődik.

Sok természetes-nyelvi feladathoz elengedhetetlenek a nagy adathalmazok. A jelenlegi publikus dialógus-adathalmazok méret és minőség közti kompromisszumokat nyújtanak (például DailyDialog (Li et al., 2017c) és Opensubtitles[1] (Tiedemann, 2012)). Második kontribúciónk egy 14.8 millió mondatból álló, jó minőségű adathalmaz létrehozása angol nyelven. A dialógusokat publikusan elérhető online könyvekből nyerjük ki. Bemutatjuk a feldolgozó rendszerünket és heurisztikáinkat, valamint a kinyert dialógusokon hiba elemzést végzünk. Zero-shot és továbbtanulási környezetben jobb válaszminőséget lehet elérni a mi adatunkon tanítva mint a hasonló méretű de zajosabb Opensubtitles adaton. Kutatók könnyen tudnak saját verziókat létrehozni az adathalmazunkból, a kompromisszum kezelő paraméterek állításával. Kódunk könnyen kiterjeszthető új nyelvekre[2].

A jelenlegi neurális háló alapú dialógus modellek unalmas és generikus válaszokat adnak nyílt-végű bemenetekre. Az olyan priorok mint a személy, hangulat, és téma,

[1] `http://www.opensubtitles.org/`

[2] `https://github.com/ricsinaruto/gutenberg-dialog` (Minden adatletöltési linket tartalmaz.)

plusz információkat szolgálnak a dialógus modelleknek a válaszminőség javítása érdekében, de ezen priorokkal való annotálás drága és ritkán elérhető. Korábbi technikák a válaszminőség javítására a modell illetve célfüggvény irányából közelítették meg a problémát. Harmadik kontribúciónk egy adatszűrési módszer, mellyel generikus mondatokat lehet eltávolítani dialógus adathalmazokból egy egyszerű felügyelet nélküli entrópia-alapú technikával. Módszerünk különböző változatait 17 metrikán hasonlítjuk össze, megmutatva hogy a szűrt adaton tanult modelleknek javul a válaszminősége mivel diverzebb válaszokat adnak.

Abstract

A conversational agent (chatbot) is a piece of software that can communicate with humans using natural language. Dialogue modelling is an important task in natural language processing and artificial intelligence (AI). Indeed, ever since the birth of AI, creating a good chatbot remains one of the field's hardest challenges. While chatbots can be used for various tasks, in general, they have to understand users' utterances and provide responses that are relevant to the problem at hand.

In the past, methods for constructing chatbot architectures have relied on hand-written rules and templates or simple statistical methods. With the rise of deep learning these models were quickly replaced by end-to-end trainable neural networks around 2015. More specifically, the recurrent encoder-decoder (Cho et al., 2014) and newer Transformer-based models (Vaswani et al., 2017) dominate the task of conversational modeling. These architectures were adapted from the neural machine translation domain, where they performs extremely well. Since then a multitude of variations and features were presented to augment the response quality.

The contributions of this work are three-fold. First, we conduct an in-depth survey of recent literature, examining over 100 publications related to chatbots, published in the last 5 years. The paradigm of training neural network-based conversational agents on large dialogue data raises the questions of how data quality affects these agents, and what evaluation methods can we use to effectively gauge the performance of trained models. We show experimentally how current automatic evaluation methods fail to capture conversational goals. The other two contributions are related to building and improving the quality of dialogue datasets.

Large datasets are essential for many NLP tasks. Current publicly available open-domain dialogue datasets offer a trade-off between size and quality (e.g. DailyDialog (Li et al., 2017c) vs. Opensubtitles[3] (Tiedemann, 2012)). Our second contribution is a high-quality dataset consisting of 14.8M utterances in English. We extract and process dialogues from publicly available online books. We present a detailed description of our pipeline and heuristics and an error analysis of extracted dialogues. Better response quality can be achieved in zero-shot and finetuning settings by training on our data than on the comparably large but much noisier Opensubtitles dataset. Researchers can easily build their versions of the dataset by adjusting various trade-off parameters. The code can be extended to further languages with limited effort[4].

[3] `http://www.opensubtitles.org/`

[4] `https://github.com/ricsinaruto/gutenberg-dialog` (Contains all data downloading links.)

One issue with current neural network-based conversational models is the lack of diversity and generation of boring responses to open-ended utterances. *Priors* such as persona, emotion, or topic provide additional information to dialogue models to aid response generation, but annotating a dataset with priors is expensive and such annotations are rarely available. Previous methods for improving the quality of open-domain response generation focused on either the underlying model or the training objective. Our third contribution is a method of filtering dialogue datasets by removing generic utterances from training data using a simple entropy-based approach that does not require human supervision. We conduct extensive experiments with different variations of our method, and compare dialogue models across 17 evaluation metrics to show that training on datasets filtered this way results in better conversational quality as chatbots learn to output more diverse responses.

Chapter 1

Introduction

A dialogue agent (chatbot) is a software that communicates with humans through natural language, by processing the users' utterances and outputting relevant and interesting responses. While there are many different ways of approaching this problem, we specifically focus on open-domain neural network-based single turn dialogue modelling. Open-domain, in contrast with goal-oriented chatbots, means that the agent should be able to have a conversation about virtually anything, similarly to humans. We focus on neural network-based models since in the last few years these have been achieving state-of-the-art results in open-domain dialogue modelling and more generally in many natural language processing (NLP) tasks (Vaswani et al., 2017, Devlin et al., 2019, Radford et al., 2019). Lastly, single-turn means that the neural network takes as input the previous turn from a dialogue and its task is to output a relevant response. This approach is also called sequence-to-sequence transduction.

The traditional approach of generating responses is to use hard-coded templates and rules to create chatbots. In contrast, the neural network approach involves training on large amounts of data to learn the process of generating relevant and grammatically correct responses to input utterances. Models have also been developed to accommodate for spoken or visual inputs. They oftentimes make use of a speech recognition component to transform speech into text (Serban et al., 2017b) or convolutional neural networks (Krizhevsky and Sutskever, 2012) that transform the input pictures into useful representations for the chatbot (Havrylov and Titov, 2017). The latter models are also called visual dialogue agents, where the conversation is grounded on both textual and visual input (Das et al., 2017).

Conversational agents exist in two main forms. The task-oriented dialogue system is limited in its conversational capabilities, however, it is very robust at executing task-specific commands and requirements. Task-oriented models are built to accomplish specific tasks like restaurant reservations (Joshi et al., 2017, Bordes et al., 2016) or promoting movies (Yu et al., 2017), to name a few. These systems often lack the ability to respond to arbitrary utterances since they are limited to a specific domain, thus users have to be guided by the dialogue system towards the task at hand. Usually, they are deployed to tasks where some information has to be retrieved from a knowledge base. They are mainly employed to replace the process of navigating through menus and user interfaces like making the activity of booking flight tickets

or finding public transportation routes between locations conversational (Zhao et al., 2017a).

The second type of dialogue agents are the non-task or open-domain chatbots. These conversation systems try to imitate human dialogue in all its facets. This means that one should hardly be able to distinguish such a chatbot from a real human, but current models are still far away from such claims. They are usually trained with dialogue examples extracted from movie scripts or from Twitter-like post-reply pairs (Vinyals and Le, 2015, Shang et al., 2015, Serban et al., 2016, Li et al., 2016b). For these models, there isn't a well-defined goal, but they are required to have a certain amount of world knowledge and commonsense reasoning capabilities to hold conversations about any topic.

Recently an emphasis has been put on integrating the two types of conversational agents. The main idea is to combine the positive aspects of both types, like the robust abilities of goal-oriented dialogue systems to perform tasks and the human-like chattiness of open-domain chatbots (Zhao et al., 2017a, Yu et al., 2017, Serban et al., 2017b). This is beneficial because the user is more likely to engage with a task-oriented dialogue agent if it is more natural, and handles out of domain responses well.

The paradigm of training neural network-based conversational agents has three main hurdles: the training data, the model, and the evaluation method. In this work, we mainly focus on the data hurdle and secondarily on evaluation methods. In Chapter 2 we discuss various dialogue datasets, neural network models, and evaluation methods used in the literature. We also give a detailed overview of the many techniques researchers have recently proposed for improving response quality. In Chapter 3 we present experimental findings regarding issues with many automatic evaluation metrics.

Current open-domain dialogue datasets offer trade-offs between quality and size. High-quality datasets are usually too small to contain the plethora of topics and knowledge chatbots should posses. Large datasets often lack good turn-segmentation and suffer from other errors, making trained models ineffective. The lack of large, high-quality dataset motivates our work. In Chapter 4 we present the building of a dataset of 14.8M utterances in English using publicly available books from Project Gutenberg[1]. Heuristics and their influence on data quality are discussed. We offer a detailed error analysis both at the utterance and dialogue level. With our MIT licensed code, researchers can easily build different dataset versions by adjusting parameters affecting the size-quality trade-off and other aspects of the dialogues. We evaluate our dataset in a generative setting using the Transformer (Vaswani et al., 2017). We compare models trained on Gutenberg and Opensubtitles (Tiedemann, 2012) by evaluating zero-shot and finetuning performance on two smaller datasets. Our modular code requires a limited amount of language-specific effort for each new language[2].

Current open-domain neural conversational models (NCM) are trained on pairs of source and target utterances to maximize the likelihood of each target given the source

[1] `https://www.gutenberg.org/`

[2] `https://github.com/ricsinaruto/gutenberg-dialog`

(Vinyals and Le, 2015). However, real-world conversations are much more complex, and a plethora of suitable targets (responses) can be adequate for a given input. We propose a data filtering approach in Chapter 5 where the "most open-ended" inputs - determined by calculating the entropy of the distribution over target utterances - are excluded from the training set. We show that dialogue models can be improved using this simple unsupervised method which can be applied to any conversational dataset. Our software for filtering dialogue data and automatic evaluation using 17 metrics is released on GitHub under an MIT license[3][4]. In Chapter 6 we discuss future work related to the presented methods. Finally, In Chapter 7 we conclude and summarise our results.

[3] github.com/ricsinaruto/NeuralChatbots-DataFiltering
[4] github.com/ricsinaruto/dialog-eval

Chapter 2

Background

2.1 Early Approaches

ELIZA is one of the first-ever chatbot programs written (Weizenbaum, 1966). It uses clever hand-written templates to generate replies that resemble the user's input utterances. Since then, countless hand-coded, rule-based chatbots have been developed (Wallace, 2009, Carpenter, 2017, Worswick, 2017). An example can be seen in Figure 2.1. Several programming frameworks have been developed specifically designed to facilitate building dialogue agents (Marietto et al., 2013, Microsoft, 2017).

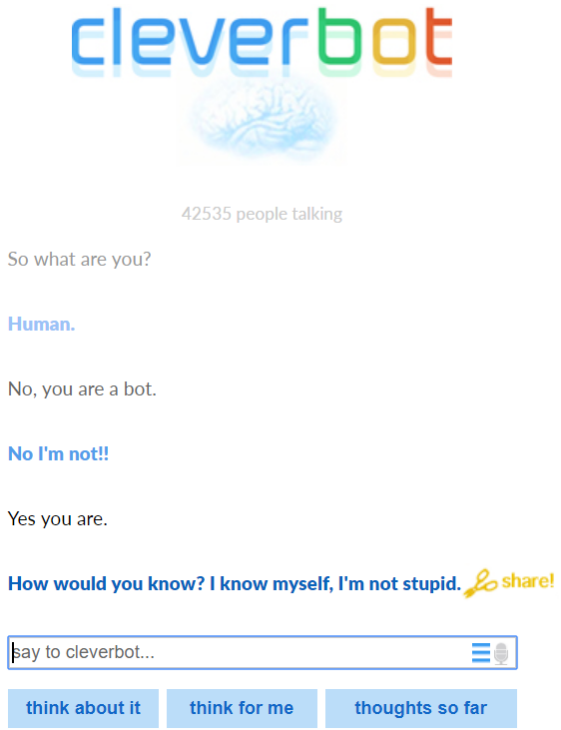

Figure 2.1: A sample conversation with Cleverbot (Carpenter, 2017)

These chatbot programs are very similar in their core, namely that they all use hand-written rules to generate replies. Usually, simple pattern matching or keyword

retrieval techniques are employed to handle the user's input utterances. Rules are used to transform a matching pattern or a keyword into a predefined reply. A simple example is shown below in AIML (Marietto et al., 2013):

<category>

<pattern>What is your name?</pattern>

<template>My name is Alice</template>
</category >

Here if the input sentence matches the sentence written between the *<pattern>* brackets the reply written between the *<template>* brackets is outputted. Another example is shown below where the *star* symbol is used for replacing words. In this case whatever word follows the word *like* it will be present in the response at the position specified by the *<star/>* token:

<category>

<pattern>I like *</pattern>

<template>I too like <star/>.</template>
</category >

2.2 Datasets

Open-domain dialogue datasets vary in size, quality, and source, among other properties (Table 2.1). Generally, smaller datasets are constructed using controlled crowdsourcing environments, making them high-quality (e.g. PersonaChat (Zhang et al., 2018b)). Crowdsourcing platforms like Amazon Mechanical Turk[1] are employed to hire instructed workers to carry out free-form conversations. Considering labour expenses crowdsourcing is not scalable, thus dialogues are automatically extracted from various text sources to build larger datasets (e.g. Opensubtitles and Reddit (Henderson et al., 2019)). Opensubtitles contains movie subtitles in multiple languages and Reddit is a discussion forum with millions of daily comments on various topics. Automatic extraction offers less quality control, and the data source heavily influences conversation characteristics. In Reddit data, everyday chit-chat is less common and comments are tied to the post. Two-party dialogues are rare as threads are almost always multi-speaker. Twitter conversations have similar problems and they are constrained by a specific character limit. Extracting conversations from Twitter and Reddit is straightforward as speaker segmentation is included and the thread chain can be used as dialogue history.

Books have seen little use as dialogue source. In DailyDialog (Li et al., 2017c), 90 000 high-quality utterances are extracted from English textbooks (extraction steps are not detailed). The quality of these dialogues and the lack of a larger book-based dataset motivates our work in Chapter 4. Dialogues extracted from books, like movie subtitles, lack context, but their usefulness is evidenced by the Cornell Corpus and DailyDialog. As argued by (Danescu-Niculescu-Mizil and Lee, 2011) and (Fainberg

[1] `https://www.mturk.com/`

Dataset	Size	Source	Quality
DailyDialog (Li et al., 2017c)	90k	textbooks	auto-extracted
Wizard-of-Wikipedia (Dinan et al., 2019b)	100k	crowdsourcing	human-written
Document-grounded (Zhou et al., 2018b)	100k	crowdsourcing	human-written
Persona-Chat (Zhang et al., 2018b)	150k	crowdsourcing	human-written
Self-dialogue (Fainberg et al., 2018)	150k	crowdsourcing	human-written
Cornell Movie Corpus (Danescu-Niculescu-Mizil and Lee, 2011)	300k	movie scripts	auto-extracted
Self-feeding chatbot (Hancock et al., 2019)	500k	human-bot dialogues	human-written (half)
Twitter corpus[2]	5M	twitter posts/replies	auto-extracted
Opensubtitles (Henderson et al., 2019)	320M	movie subtitles	auto-extracted
Reddit (Henderson et al., 2019)	730M	reddit threads	auto-extracted

Table 2.1: Comparison of open-domain dialogue datasets. *Size* is the rough number of utterances, *Source* describes where the data comes from, and *Quality* distinguishes between dataset collection techniques.

et al., 2018) artificial dialogues in movies and books generally resemble natural conversations. Unfortunately, the Cornell Corpus (Danescu-Niculescu-Mizil and Lee, 2011) is relatively small, and Opensubtitles lacks dialogue and turn segmentation, resulting in a lot of non-dialogue text and other errors. Subtitle lines are equated to turns and dialogue history consists of the previous n lines. Because of these characteristics almost zero processing and heuristics are used (e.g. (Henderson et al., 2019) remove short and long utterances). Also, the types and amount of errors has not been explicitly analyzed for these datasets. For our dataset, we build a multi-step extraction pipeline and analyze the performance of heuristics and error types in the final corpus.

2.3 Models

2.3.1 Deep Learning and NLP

The main concept that differentiates rule-based and neural network-based approaches is the presence of a learning algorithm in the latter case. An important distinction has to be made between traditional machine learning and deep learning which is a subfield of the former. In this work, only deep learning methods applied to chatbots are discussed, since neural networks have been the backbone of conversational modelling and traditional machine learning methods are only rarely used as supplementary techniques.

[2] `https://github.com/Marsan-Ma-zz/chat_corpus`

When applying neural networks to natural language processing (NLP) tasks each word (symbol) has to be transformed into a numerical representation (Bengio et al., 2003). This is done through word embeddings, which represent each word as a fixed size vector of real numbers. Word embeddings are useful because instead of handling words as huge vectors of the size of the vocabulary, they can be represented in much lower dimensions. Word embeddings are trained on large amounts of natural language data and the goal is to build vector representations that capture the semantic similarity between words. More specifically, because similar context usually is related to similar meaning, words with similar distributions should have similar vector representations. This concept is called the Distributional Hypothesis (Harris, 1954). Each vector representing a word can be regarded as a set of parameters and these parameters can be jointly learned with the neural network's parameters, or they can be pre-learned, a technique described in Section 2.3.4.3.

Instead of using hand-written rules, deep learning models transform input sentences into replies directly by using matrix multiplications and non-linear functions that contain millions of parameters. Neural network-based conversational models can be further divided into two categories, retrieval-based and generative models. The former simply returns a reply from the dataset by computing the most likely response to the current input utterance. This can be done using a scoring function, which can be implemented as a neural network (Cho et al., 2014) or by simply computing the cosine similarity between the word embeddings of the input utterances and the candidate replies (Li et al., 2016e). Generative models, on the other hand, synthesize the reply one word at a time by computing probabilities over the whole vocabulary (Sutskever et al., 2014, Vinyals and Le, 2015). There have also been approaches that integrate the two types of dialogue systems by comparing a generated reply with a retrieved reply and determining which one is more likely to be a better response (Song et al., 2016). As with many other applications, the field of conversational modelling has been transformed by the rise of deep learning. More specifically the encoder-decoder recurrent neural network (RNN) model (also called seq2seq (Sutskever et al., 2014)) introduced by (Cho et al., 2014) and Transformer-based models (Vaswani et al., 2017) have been dominating the field.

A recurrent neural network (RNN) (Rumelhart et al., 1988) is a neural network that can take as input a variable length sequence $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ and produce a sequence of hidden states $\boldsymbol{h} = (\boldsymbol{h}_1, ..., \boldsymbol{h}_n)$, by using recurrence. This is also called the unrolling or unfolding of the network, visualized in Figure 2.2. At each step the network takes as input $\boldsymbol{x}_i$ and $\boldsymbol{h}_{i-1}$ and generates a hidden state $\boldsymbol{h}_i$. At each step i, the hidden state $\boldsymbol{h}_i$ is updated by

$$\boldsymbol{h}_i = f(W\boldsymbol{h}_{i-1} + U\boldsymbol{x}_i) \tag{2.1}$$

where W and U are matrices containing the weights (parameters) of the network. f is a non-linear activation function which can be the hyperbolic tangent function for example. The vanilla implementation of an RNN is rarely used, because it suffers from the vanishing gradient problem which makes it very hard to train (Hochreiter, 1998). Usually long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent units (GRU) (Cho et al., 2014) are used for the activation function. LSTMs were developed to combat the problem of long-term dependencies that vanilla

RNNs face. As the number of steps of the unrolling increase it becomes increasingly hard for a simple RNN to learn to remember information seen multiple steps ago.

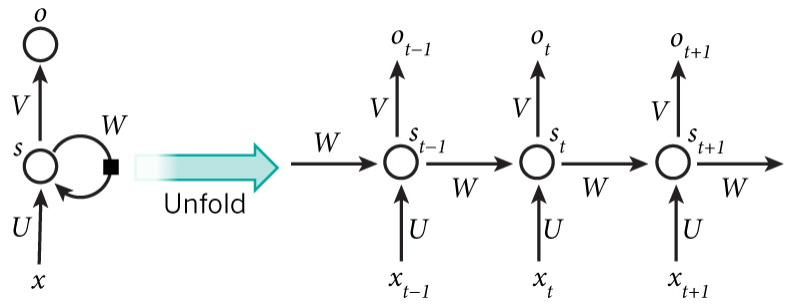

Figure 2.2: Unfolding of an RNN over 3 time-steps. Here x is the input sequence, o is the output sequence, s is the sequence of hidden states and U,W and V are the weights of the network. (Britz, 2015)

LSTM networks use a gating mechanism to control the information flow in the recurrent network. More specifically the input and forgetting gates are used to determine how to update the network's state and the output gate is used to determine what to output from the hidden state. Mathematically these gates consist of several matrix multiplications and non-linear functions applied to the input vector and the previous hidden state. LSTMs are particularly useful for language modelling because information has to preserved over multiple sentences while the network is unrolled over each word. Because of space constraints, LSTMs are not detailed further, but a good and quick explanation can be found in (Olah, 2015). An important characteristic of recurrent neural networks is that the parameters of the function f don't change during the unrolling of the network.

Language modeling is the task of predicting the next word in a sentence based on the previous words (Bengio et al., 2003). RNNs can be used for language modeling by training them to learn the probability distribution over the vocabulary V given an input sequence of words. As previously discussed an RNN receives the word embedding vectors representing words in lower dimensions. The probability distribution can be generated to predict the next word in the sequence by taking the hidden state of the RNN in the last step, and feeding it into a softmax activation function

$$p(\boldsymbol{x}_{i,j}|\boldsymbol{x}_{i-1},...,\boldsymbol{x}_1) = \frac{\exp(\boldsymbol{v}_j \boldsymbol{h}_i)}{\sum_{j=1}^{K} \exp(\boldsymbol{v}_j \boldsymbol{h}_i)} \tag{2.2}$$

for all possible words (symbols) $j = 1, ..., K$, where $\boldsymbol{v}_j$ are the rows in the V weight matrix of the softmax function. $(\boldsymbol{x}_{i-1}, ..., \boldsymbol{x}_1)$ is the input sequence and $\boldsymbol{h}_i$ is the hidden state of the RNN at step i.

Training of these networks is done via the generalized backpropagation algorithm called truncated backpropagation through time (Werbos, 1990, Rumelhart et al., 1988). Essentially the error is backpropagated through each time-step of the network to learn its parameters. The error can be computed by using the cross-entropy loss

function, which calculates the difference between predictions and true labels.

$$\text{Loss}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = -\boldsymbol{y}_i \log(\hat{\boldsymbol{y}}_i) \tag{2.3}$$

where $\hat{\boldsymbol{y}}_i$ is the vector of the predicted probabilities over all words in the vocabulary at step i, and $\boldsymbol{y}_i$ is the one-hot vector over the vocabulary. A one-hot vector is made up of zeros except at the index of the one true word that follows in the sequence, where it is equal to 1. After computing the derivative with respect to all of the weights in the network using the backpropagation through time algorithm, the weights can be updated to get closer to an optimum with optimization techniques like stochastic gradient descent (SGD) (Bottou, 2010).

2.3.2 The Encoder-Decoder Model

The sequence to sequence model (seq2seq) was first introduced by (Cho et al., 2014), but they only used it to re-rank sentences instead of generating completely new ones, which was first done by (Sutskever et al., 2014). Since then, besides NMT and conversational models a plethora of applications of these models have been introduced like text summarization (Nallapati et al., 2016), speech recognition (Chiu et al., 2017), code generation (Barone and Sennrich, 2017) and parsing (Konstas et al., 2017), to name a few.

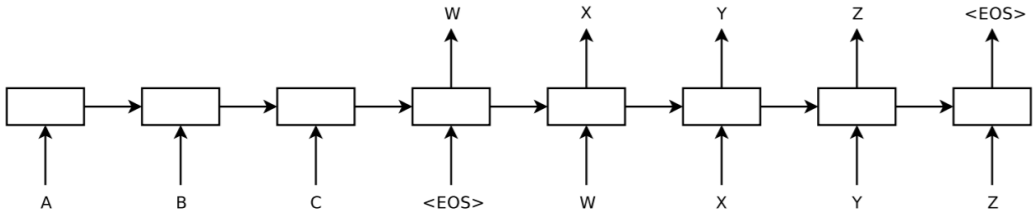

Figure 2.3: A general seq2seq model, where (A, B, C) is the input sequence, $< EOS >$ is a symbol used to delimit the end of the sentence and (W, X, Y, Z) is the output sequence (Sutskever et al., 2014)

The simplest and initial form of the model is based on two RNNs, visualized in Figure 2.3. The actual implementation of RNNs can be in the form of LSTMs or GRUs. The goal is to estimate the conditional probability of $p(\boldsymbol{y}_1, ..., \boldsymbol{y}_{N'} | \boldsymbol{x}_1, ..., \boldsymbol{x}_N)$, where $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ is the input sequence and $\boldsymbol{y}_1, ..., \boldsymbol{y}_{N'}$ is the corresponding output sequence. Since two different RNNs are used for the input and output sequences, the lengths of the sequences, N and $N^{'}$ can be different. The encoder RNN is unrolled over the words in the source sequence and its last hidden state is called the thought vector, which is a representation of the whole source sequence. The initial hidden state of the decoder RNN is then set to this representation $\boldsymbol{v}$, and the generation of words in the output sequence is done by taking the output of the unrolled decoder network at each time-step and feeding it into a softmax function, which produces

the probabilities over all words in the vocabulary:

$$p(\boldsymbol{y}_1, ..., \boldsymbol{y}_{N'} | \boldsymbol{x}_1, ..., \boldsymbol{x}_N) = \prod_{i=1}^{N'} p(\boldsymbol{y}_i | \boldsymbol{v}, \boldsymbol{y}_1, ..., \boldsymbol{y}_{i-1}) \tag{2.4}$$

Training is very similar to a normal RNN, namely the log probability of a correct target sequence T given the source sequence S is maximized

$$\frac{1}{\boldsymbol{S}} \sum_{T,S \in \boldsymbol{S}} \log(p(T|S)) \tag{2.5}$$

where $\boldsymbol{S}$ is the training set. The two networks are trained jointly, errors are back-propagated through the whole model and the weights are optimized with some kind of optimization technique like SGD.

In NMT the input sequences are sentences in one language from which they have to be translated to the target sequences, which are sentences in a different language. The individual elements of a sequence, or sentence, in this case, are vectors representing word embeddings. In conversational modelling, the simplest approach is to treat an utterance by a speaker as an input sequence and the response to that utterance from a different speaker as the target sequence. Better approaches, however, are discussed in Section 2.3.3.

Seq2seq models can also contain multiple layers of LSTM networks as seen in Figure 2.4. This is done to make the model deeper and to have more parameters, which should ideally lead to better performance (Vinyals and Le, 2015, Wu et al., 2016). There exist multiple variants, but the most straightforward one is to feed in the source sentence to the first layer of the encoder network. The output from the previous LSTM layer is fed as input to the next layer and the layers are unrolled jointly. Then the last hidden state of the final encoder layer can be used to initialize the initial hidden state of the first decoding layer. The output of the previous decoder layer is input to the next layer until the final layer, where a softmax activation function is applied over the outputs from the last layer to generate the predicted output sequence. The initialization of layers in the decoder network can be implemented in various ways, for example taking the last hidden state from each encoder layer and using it to initialise the first hidden state of each corresponding decoder layer.

To get the actual generated output sequences there are several techniques to decode from the probabilities of the words. One such technique is the Roulette Wheel selection, which is commonly used for genetic algorithms (Mitchell, 1998). Using this method, at each time-step, each word from the vocabulary can be generated with the probability computed from the softmax function. This is useful if the goal is to have some stochasticity in the decoding process because it can produce slightly different outputs for the same source sequence. However, it doesn't perform as well as the more frequent method used, which is to simply output the word with the highest probability from the softmax function. This is a greedy and deterministic approach since it always outputs the same output for the same input.

While decoding a word at each time-step is fine a better approach is to the decode the whole output sequence at the same time, by outputting the sequence with the

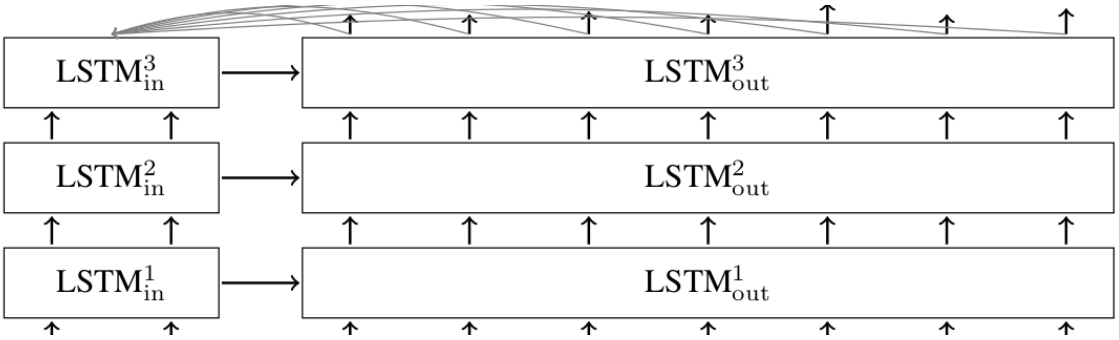

Figure 2.4: A 3 layer seq2seq model (Tensorflow, 2017). The lines pointing from the last decoder states to the last encoder represent an attention mechanism, which is presented in Section 2.3.4.1

highest probability.

$$\hat{T} = \arg\max_{T} p(T|S) \tag{2.6}$$

Here S is the source sentence and T is the target sentence. To get the sequence with the highest probability, first, all of the possible sequences have to be generated. A simple left-to-right beam search is usually employed to make the computation tractable (Graves, 2012). In the first time-step of the decoder, the top K words with the highest probabilities are kept. Then at each time-step, this list is expanded by computing the joint probability of the partial sequences in the list and the words in the current time-step and retaining the K most probable partial sequences until the end of the output sequence is reached.

Recently state-of-the-art techniques have been developed that put the sampling approach in a much better position. Top-k and top-p sampling (Fan et al., 2018, Holtzman et al., 2020, Radford et al., 2019) sample a word from the probability distribution outputted by the model at each time step. This distribution can be truncated to the top-k words or top-p probability mass. Empirical evidence and arguments against greedy and beam decoding make these sampling methods a potentially much better alternative (Murray and Chiang, 2018, Yang et al., 2018, Weston et al., 2018, Dinan et al., 2019a).

Another important aspect of sequence-to-sequence models applied to tasks involving language is the vocabulary. The vocabulary consists of all the various words and symbols present in the dataset. One problem with this approach is that the vocabulary tends to be quite large, especially for big datasets (Lison and Tiedemann, 2016, opensubtitles.org, 2017). Since the number of parameters of the model increases proportionally with the size of the vocabulary it is usually the case that the vocabulary is limited to some arbitrary size N. This way only the embeddings of the N most frequent words in the dataset are used and any other symbols are replaced with a common token representing unknown words. Many approaches have been proposed to the problem of handling out of vocabulary (OOV) or unknown words (Luong et al., 2014, Feng et al., 2017, Jean et al., 2014). Other methods involve using characters (Zhu et al., 2017) or subword units (Sennrich et al., 2015) instead of words.

2.3.3 Hierarchical models

A bidirectional RNN (BiRNN) consists of two RNNs, a forward and a backward one, visualized in Figure 2.5 (Schuster and Paliwal, 1997). The forward RNN reads the input sequence in the original order (from $\boldsymbol{x}_1$ to $\boldsymbol{x}_n$) and computes a sequence of forward hidden states $(\overrightarrow{\boldsymbol{h}}_1, ..., \overrightarrow{\boldsymbol{h}}_n)$. The backward RNN reads the reversed sequence (from $\boldsymbol{x}_n$ to $\boldsymbol{x}_1$) and computes the backward hidden states $(\overleftarrow{\boldsymbol{h}}_1, ..., \overleftarrow{\boldsymbol{h}}_n)$. Then the two hidden states can be simply concatenated for each word in the sequence (Bahdanau et al., 2014, Zhao et al., 2017b). The BiRNN resembles a continuous bag of words model (Mikolov et al., 2013a), because each concatenated hidden state $\boldsymbol{h}_i$ has information about the words surrounding the i-th word. This results in a better preservation of context and is used in several seq2seq models, usually in the encoder network (Zhao et al., 2017b, Xing et al., 2017, Wu et al., 2016, Yin et al., 2017).

So far, only conversational modelling as predicting a reply based on a single source utterance has been discussed. However, conversations are more complicated than machine translation, since they don't solely consist of utterance-reply pairs. Dialogues usually have many turns, and the reply to an utterance might depend on the information presented in previous turns. A plethora of approaches have been proposed to incorporate context or conversation history into seq2seq models to build better dialogue agents. Perhaps the most straightforward approach is to concatenate k previous utterances by appending an end-of-utterance symbol after each utterance and feeding this long sequence of symbols into the encoder (Vinyals and Le, 2015). The simplest approach which was used as a baseline in (Sordoni et al., 2015) is to use only the first preceding utterance as context to form context-message-reply triples for training. A better approach was to concatenate the bag of words representations of the context and message utterances instead of the actual utterances. By using different representations for different utterances better results can be achieved (Sordoni et al., 2015).

However, feeding long sequences of symbols into the encoder RNN of seq2seq models accentuates the vanishing gradient problem (Hochreiter, 1998). Moreover, as sequences get longer it becomes increasingly more difficult for RNNs to retain information which was inputted many time-steps ago. Consequently, it becomes even harder to encode all relevant information into the last hidden state of the encoder network from a sequence consisting of multiple sentences.

To better represent dialogue history, a general hierarchical recurrent encoder-decoder (HRED) architecture was proposed in (Serban et al., 2016). The model consists of three different RNNs, the encoder RNN, the context RNN and the decoder RNN. First, k previous utterances of a conversation are encoded separately by the encoder RNN. This produces k separate context vectors by taking the last hidden state of the encoder RNN from each encoded utterance. Then these k hidden states are fed into the context RNN step by step, thus it has to be unrolled for k steps. The last hidden state from the context is used to initialize the decoder RNN. The decoder RNN and the decoding process is very similar to the one found in a normal seq2seq model.

The HRED (Figure 2.6) differs from the basic encoder-decoder model by introducing a new level of hierarchy, the context RNN, which computes hidden states over entire

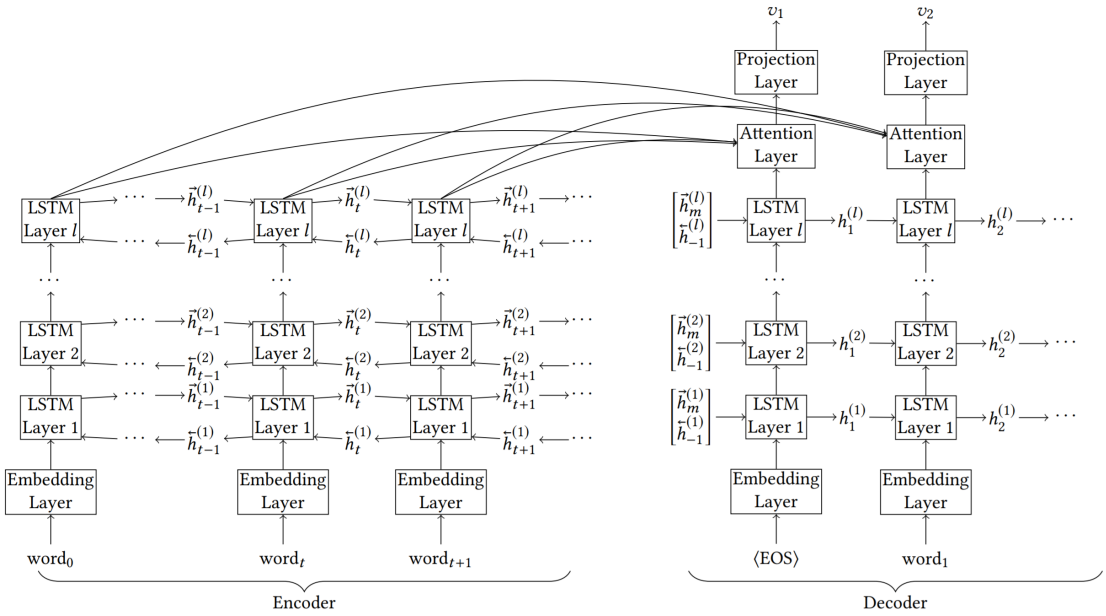

Figure 2.5: A bidirectional multilayer LSTM encoder and unidirectional LSTM decoder (Yin et al., 2017) with attention. Attention is described in Section 2.3.4.1.

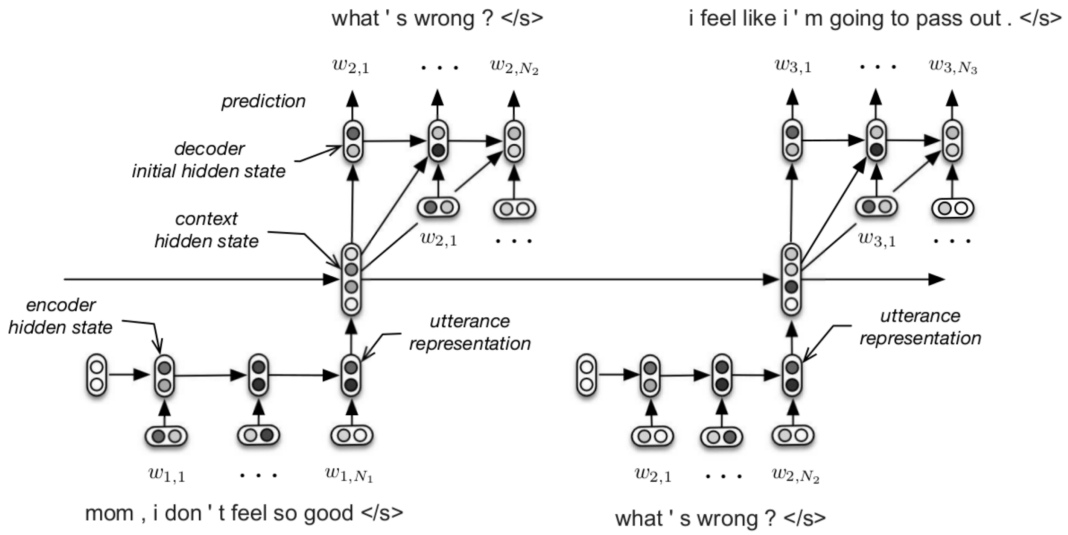

Figure 2.6: The Hierarchical Recurrent Encoder-Decoder Model (Serban et al., 2016).

utterances. With this approach, the natural hierarchy of conversations is preserved by handling word sequences with the encoder RNN and utterance sequences with the context RNN which depends on the hidden representations produced by the word-level encodings.

Since the introduction of the HRED model a number of works have used and augmented it (Serban et al., 2017c, Serban et al., 2017a, Serban et al., 2017b, Shen

et al., 2017, Li et al., 2017b). A proposed extension to the HRED model is to condition the decoder RNN on a latent variable, sampled from the prior distribution at test time and the approximate posterior distribution at training time (Serban et al., 2017c). These distributions can be represented as a function of previous sub-sequences of words.

In (Serban et al., 2017a) two HRED models are employed simultaneously. One HRED operates over coarse tokens of the conversation (eg. part-of-speech tags) to generate coarse predictions. Then, the second HRED, which takes as input natural language utterances, generates natural language predictions by conditioning on the predicted coarse tokens. This conditioning is done via the concatenation of the last hidden state of the decoder RNN of the coarse predictions with the current context RNN hidden state from the natural language HRED and feeding it into the natural language decoder RNN.

To handle the two-party style of conversations, two separate hierarchical recurrent encoder (HRE) models are used in (Shen et al., 2017). The HRE is the same as the HRED without the decoder RNN. One of the HRE networks encodes only the utterances coming from one of the speakers, and the other HRE encodes the utterances of the other speaker. Thus, at each turn the two networks produce two hidden context states which are concatenated and fed into a single decoder RNN, producing the output prediction.

2.3.4 Attention-based models

2.3.4.1 Attention

The attention mechanism visualized in Figure 2.7 was first introduced to encoder-decoder models by (Bahdanau et al., 2014). The problem that it was trying to address is the limited information that the context vector can carry. In a standard seq2seq model, it is expected that this single fixed-size vector can encode all the relevant information about the source sentence that the decoder needs to generate its prediction. Additionally, since only a single vector is used, all the decoder states receive the same information, instead of feeding in information relevant to the specific decoding step.

In order to combat these shortcomings the attention mechanism creates an additional input at each decoding step coming directly from the encoder states. This additional input $\boldsymbol{c}_i$ to the decoder at time-step i is computed by taking a weighted sum over the encoder hidden states $\boldsymbol{h}$:

$$\boldsymbol{c}_i = \sum_{j=1}^{T} a_{ij} \boldsymbol{h}_j \tag{2.7}$$

where T is the number of hidden states or symbols in the source sentence. The weight a_{ij} for each hidden state $\boldsymbol{h}_j$ can be computed by

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})} \tag{2.8}$$

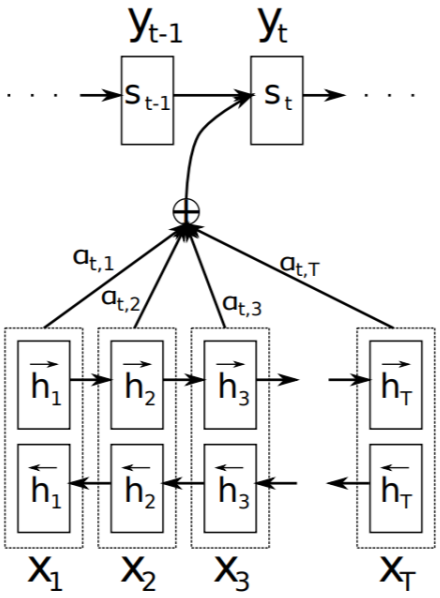

Figure 2.7: Original Attention Mechanism (Bahdanau et al., 2014).

which is basically a softmax function over e_{ij}, which is the output of a scoring function:

$$e_{ij} = f(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j). \tag{2.9}$$

Here $\boldsymbol{s}_{i-1}$ is the hidden state of the decoder in the previous time-step. Oftentimes $\boldsymbol{s}_{i-1}$ is called a query vector ($\boldsymbol{q}$) and the encoder hidden states $\boldsymbol{h}$ are called key vectors ($\boldsymbol{k}$). A good visualization of the weights a_{ij} between the source and output sentence can be seen in Figure 2.8. The scoring function f can be implemented in several ways. In Equation 2.10 a Multi-Layer Perceptron (MLP) approach can be seen, where the query and key vectors are simply concatenated and fed into a feed-forward neural network (Bahdanau et al., 2014). The Bilinear (BL) scoring function is described in Equation 2.11, proposed by (Luong et al., 2015). The Dot Product (DP) scoring function (Equation 2.12) does not use any weights, but it requires the sizes of the two vectors to be the same (Luong et al., 2015). An extension of this is the Scaled Dot Product (SDP) function (Equation 2.13), where the dot product is scaled by the square root of the size of the key vectors (Vaswani et al., 2017). This is useful because otherwise the dot product is proportional to dimension size.

$$\text{MLP}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w_2} tanh(W_1[\boldsymbol{q} : \boldsymbol{k}]) \tag{2.10}$$

$$\text{BL}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\mathsf{T} W \boldsymbol{k} \tag{2.11}$$

$$\text{DP}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\mathsf{T} \boldsymbol{k} \tag{2.12}$$

$$\text{SDP}(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\mathsf{T} \boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}} \tag{2.13}$$

In the above equations W, W_1 and $\boldsymbol{w}_2$ are parameters of the scoring functions.

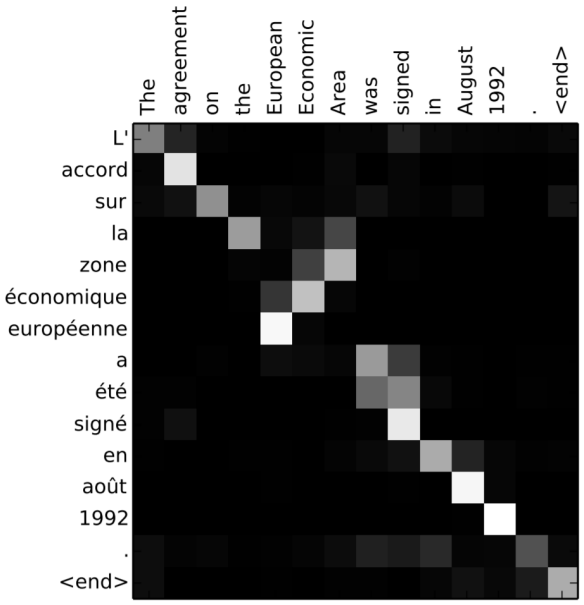

Figure 2.8: Pixels represent the weights a_{ij} between inputs and outputs (Bahdanau et al., 2014).

Outputs from the decoder network generated in previous time-steps can be attended to by incorporating them as additional inputs to the scoring functions mentioned above (Shao et al., 2017). This helps the decoder by allowing it to *see* what has already been generated. Attention can also be implemented using multiple heads (Vaswani et al., 2017), meaning that the output of multiple scoring functions is used, each with their parameters, to learn to focus on different parts of the input sequence. The parameters of the scoring functions are jointly learned with all other parameters of the seq2seq model.

Attention can also be computed using only one sequence of symbols, called self- or intra-attention. Self-attention has been successfully applied to a variety of tasks including reading comprehension, abstractive summarization, neural machine translation and sentence representations (Cheng et al., 2016, Lin et al., 2017, Vaswani et al., 2017, Parikh et al., 2016, Paulus et al., 2017). In this case, the query and key vectors both come from the same hidden states and thus an encoding of the sentence which depends on all of its parts is achieved. It can be used both in the decoder and the encoder networks since it can be implemented as a standalone layer that takes in a sequence and computes a new representation of that sequence. Figure 2.9 depicts the self-attention mechanism between the same sentence.

Attention has been used extensively for neural conversational models as an augmentation of the base seq2seq model (Yao et al., 2016, Shang et al., 2015, Xing et al., 2017, Zhao et al., 2017a). A more complex type of attention mechanism is the hierarchical attention which was used in conversational modelling (Xing et al., 2017) and abstractive text summarization (Nallapati et al., 2016) as well. A natural extension to the original HRED model discussed in Section 2.3.3 is to incorporate attention mechanisms, explored in several works (Yao et al., 2015, Yao et al., 2016, Xing et al.,

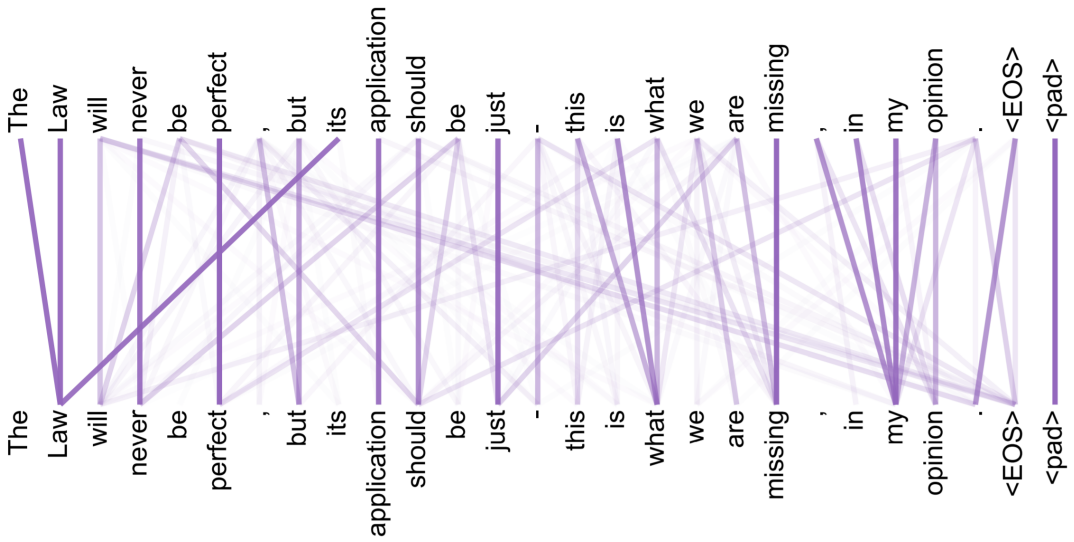

Figure 2.9: Visualizing self-attention, lines represent the weights a_{ij} (Vaswani et al., 2017).

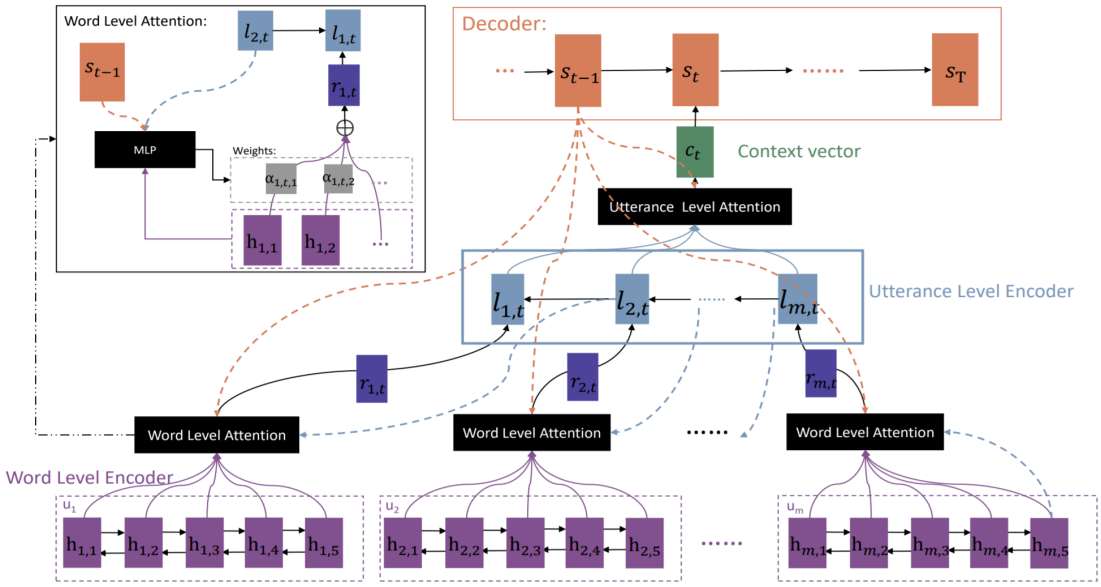

Figure 2.10: The Hierarchical Recurrent Attention Network (Xing et al., 2017).

2017). The simplest form of integrating attention into the HRED model is between the previous decoder RNN hidden state and the encoder RNN hidden states from the previous turn. This can be done in the same way as in standard seq2seq models and has been explored in (Yao et al., 2015, Yao et al., 2016).

A more interesting approach is to make use of the hierarchical structure of the HRED and integrate attention hierarchically as well (Xing et al., 2017). Accordingly, the model in this work is called the hierarchical recurrent attention network (HRAN). There are two levels of attention employed. At each time-step, the word level attention mechanism computes vector representations over the hidden states of the encoder

RNN (keys) and the previous hidden state of the decoder RNN (queries). Each utterance is encoded separately by the encoder RNN and word-level attention is also computed separately for each utterance. The produced vector representations are fed into the utterance level encoder (context RNN). Then, the utterance level attention mechanism computes a context vector based on the hidden states of the context RNN. The last step is to feed this context vector into the decoder RNN at each step, which computes the output sequence predictions. An additional technique employed in the HRAN architecture is to use context RNN hidden states as inputs to the word level attention. More specifically the context RNN is implemented in a reverse order, meaning that if there is a sequence of utterances $(u_1, ..., u_n)$, the encoder RNN first encodes u_n. Thus the utterance level encoder will also start with the vector representation from the last utterance in the conversation. Because the utterances are reversed, the word level attention mechanism for each utterance can use as additional input the hidden state of the context RNN from the next utterance in the original order. Figure 2.10 depicts the HRAN with all the components mentioned above.

2.3.4.2 The Transformer

The Transformer is an encoder-decoder model based solely on attention mechanisms and feed-forward neural networks, originally achieving state-of-the-art results in NMT tasks (Vaswani et al., 2017). It forms the basis of the recent models achieving state-of-the-art performance in a plethora of NLP tasks (Devlin et al., 2019, Radford et al., 2019, Wolf et al., 2018). In this section, its architecture is described in detail, which can be seen in Figure 2.11. The model is made up of two sub-networks, the encoder and the decoder networks. The encoder network takes as input the word embeddings of the source sentence and produces a representation. Then, the decoder network takes as input this representation and the word embeddings of already generated tokens from the output sentence. Based on these it produces output probabilities for each word. The decoder is auto-regressive because it generates predictions based on previously predicted words.

Encoder and Decoder Networks The encoder network contains 6 identical layers. The output of a previous layer serves as input to the next layer. Within 1 layer, there are two main blocks. First, the input embeddings go through a multi-head self-attention block. Then the outputs from this block serve as input to a position-wise fully connected network. Both blocks are augmented with residual connections and layer normalization.

Similarly, the decoder network also consists of 6 layers. However, it contains 3 blocks, the first and third being the same as the two blocks in the encoder network. The additional middle block is a multi-head attention mechanism over the output of the encoder, connecting the two networks. An important difference in the decoder's first multi-head self-attention block is that it takes as input already generated word embeddings from the output sentence. Since the size of the inputs is fixed, not yet generated output embeddings are masked to ensure that they don't take part in computing the attention. This, together with shifting the output embeddings by one

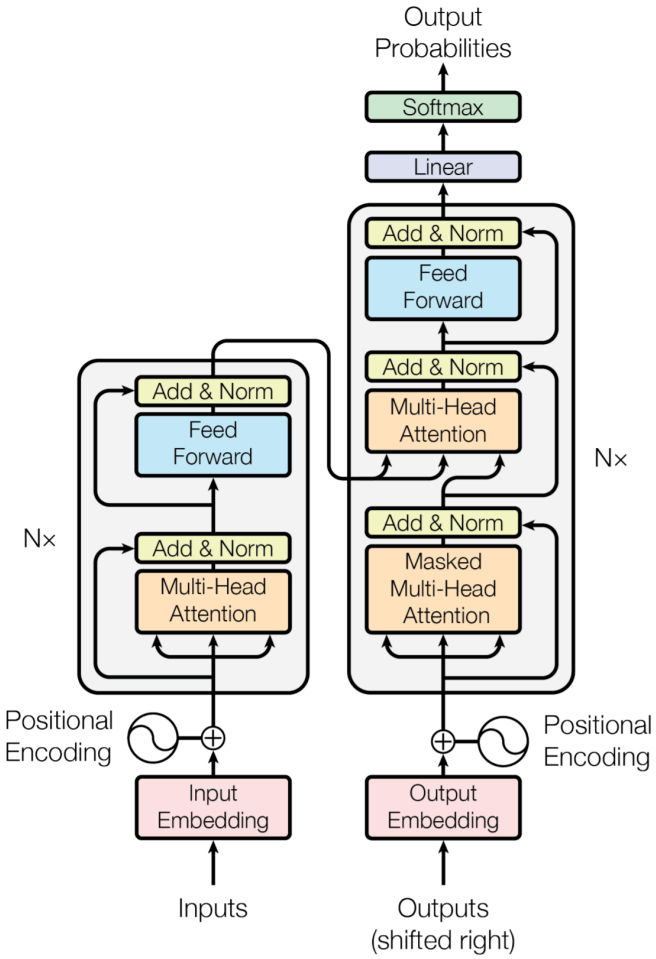

Figure 2.11: The architecture of the Transformer model (Vaswani et al., 2017). The encoder network can be seen on the left and the decoder network on the right.

position ensures that predictions for position i in the output sentence only depend on previously generated words at positions less than i.

Attention Mechanisms The attention mechanism is implemented as multi-head scaled dot-product attention over query, key and value vectors ($\boldsymbol{q}$,$\boldsymbol{k}$,$\boldsymbol{v}$). In self-attention blocks, these vectors all come from input embeddings. Each input vector is projected three times with separate projections to get queries, keys and values of the same dimensions as the original input. In the attention block which connects the encoder and decoder networks, the queries are the outputs of the previous attention block in the decoder and the key and value vectors come from projecting the output of the final encoder layer two times with separate projections. In the first self-attention block in the decoder, masking is used by setting all values to $-\infty$ in the input of the softmax functions.

For computing the scaled dot-product attention, the following equation is used:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d_k}}\right)V \tag{2.14}$$

where Q,K,V are matrices built from the query, key and value vectors respectively across the whole input sequence. d_k is the dimension of the key vectors.

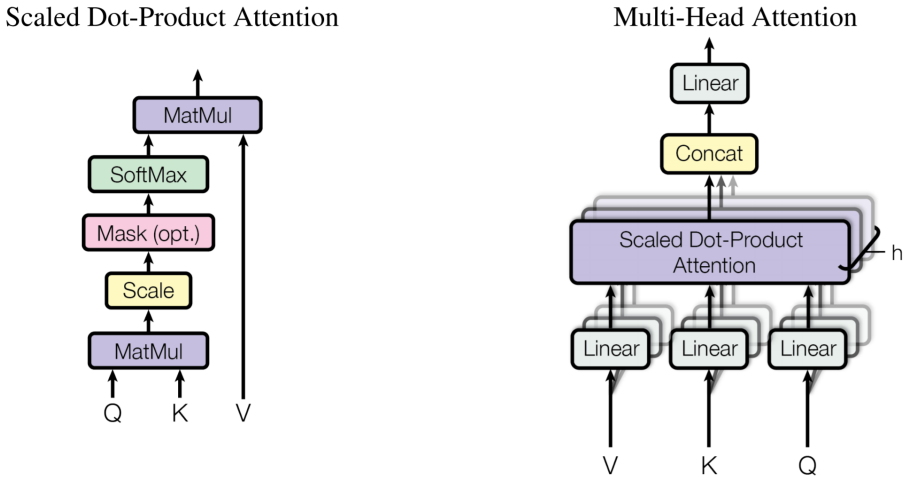

Figure 2.12: A diagram of the scaled dot-product attention can be seen on the left, and of the multi-head attention on the right (Vaswani et al., 2017). Q,K,V are the query, key and value matrices respectively and h denotes the number of attention heads.

Multi-head attention (Figure 2.12) means that there are several parallel channels of attention computations over separate inputs. To get separate inputs, the query, key and value matrices are linearly projected to a dimension size that is equal to their original dimension divided by the number of heads. For each head and each vector the projection is done via a separate learned weight matrix. This way instead of computing the attention over the whole $d_{model} = 512$ dimensional inputs at once, the computation is divided into $h = 8$ heads over $\frac{d_{model}}{h} = 64$ dimensional query, key and value vectors (Equation 2.15). Finally, the outputs of all heads are concatenated and once again projected, resulting in the final output value of the attention block (Equation 2.16).

$$\text{head}_i = \text{Attention}(QW_{Q_i}, KW_{K_i}, VW_{V_i}) \tag{2.15}$$

$$\text{MultiHead}(Q, K, V) = \text{Concatenate}(\text{head}_i, ..., \text{head}_h)W_O \tag{2.16}$$

where W_{Q_i},W_{K_i},W_{V_i},W_O are learned weight matrices.

Feed-Forward Networks The feed-forward (FFN) blocks are implemented as two consecutive convolutions with kernel size 1, with a ReLU activation between them

(Equation 2.17). This means that each filter contains a singular weight, and this weight is multiplied with each element from the input vector separately to produce an output vector:

$$\text{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{x}W_1 + \boldsymbol{b}_1)W_2 + \boldsymbol{b}_2 \tag{2.17}$$

where $\boldsymbol{x}$ is the input vector, W_1 and W_2 are the weight matrices and $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the bias vectors.

Positional Encoding Since the model does not use convolutions or recurrence it lacks positional information about the words in a sequence. In order to address the issue, positional encodings are used. Each word has a separate representation related to its position in the sentence. They are the same size as the word embeddings, so the two vectors can be summed up, to produce a final representation for each word. The positional encoding vectors are not learned, but rather constructed using sine and cosine functions of different frequencies:

$$PE[pos_j, 2i] = \sin\left(\frac{pos_j}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.18}$$

$$PE[pos_j, 2i+1] = \cos\left(\frac{pos_j}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.19}$$

where $PE[pos_j]$ is the positional encoding for the j-th word in the sequence and $i \in \{0, 1, ..., d_{model} - 1\}$.

Regularization Layer normalization (Ba et al., 2016) is used over each block of the model. This is a regularization technique which computes statistics over a layer and normalizes the output of the layer based on them. Additionally, layer normalization is implemented over the sum of the output of the layer *Layer* and the original input X to that layer:

$$\text{LayerNormalization}(X + \text{Layer}(X)) \tag{2.20}$$

This connection between the input and output of a layer is called a residual connection (He et al., 2016). Dropout (Srivastava et al., 2014) is employed as a regularization technique. Dropout is a stochastic function that takes as input an array and replaces each element with 0 according to a given probability. Dropout is applied to the output of each block (before layer normalization) and during the computation of the scaled dot-product attention.

2.3.4.3 Pretraining and GPT

Pretraining for encoder-decoder models means that instead of initializing the parameters of a model randomly the model is first pre-trained on some data or some other task that is different from the main task that the model needs to be applied to. One of the most common approaches among many NLP tasks is to pre-train the parameters of the word embeddings (Chen and Manning, 2014, Serban et al.,

2016, Akasaki and Kaji, 2017, Lample et al., 2016, Serban et al., 2017b). The most popular techniques to pre-train word embeddings are presented in (Mikolov et al., 2013a, Mikolov et al., 2013b). An advantage of pretraining word embeddings is that during the actual training of the seq2seq model these embeddings can be fixed and thus the model has to learn less parameters.

In addition to pretraining word embeddings, all of the parameters of an encoder-decoder model can be pre-trained as well. For conversational modelling, this is very beneficial, because oftentimes well-labelled datasets are relatively small. By pretraining on a big but noisier dataset the parameters of a model, it will already achieve somewhat good performance. Thus the model will have learned a good amount of general knowledge about the task (Li et al., 2016b, Serban et al., 2016). Then, by finetuning on a smaller dataset it can achieve better performance without overfitting. For example, a conversational model can learn general knowledge like answering with yes/no to a yes-or-no question. Then, during finetuning the model doesn't have to learn all of this knowledge again and thus can focus on more subtle properties of conversations or domain-specific knowledge. In (Li et al., 2016b) a seq2seq model was pre-trained on the OpenSubtitles dataset (Lison and Tiedemann, 2016) and then the pre-trained model was adapted to a much smaller TV series dataset. A somewhat different approach was employed in (Ramachandran et al., 2016), where the authors pre-trained the encoder and decoder RNNs of a seq2seq model separately as language models. Thus both networks already had some knowledge about language before putting them together and training the whole seq2seq model for NMT. Similarly in (Sriram et al., 2017) a pre-trained language model (LM) was used to augment the performance of a standard seq2seq model. With the information from the LM, the seq2seq model was able to improve its performance for domain adaptation, which means that it performed almost as good on a different domain as on the domain it was trained on.

A different style of pretraining was employed in (Wiseman and Rush, 2016). Instead of pretraining a seq2seq model with a different dataset, they pre-trained the model with a different loss function. This was necessary since the authors tried to integrate beam search into the loss function, but found that without first pretraining with the standard cross-entropy loss function the model was unable to learn anything with their new loss function.

Recently, pre-training and finetuning large language models on specific tasks (including dialogue modelling) has gained popularity (Wolf et al., 2018, Devlin et al., 2019, Radford et al., 2019, Zhang et al., 2019). GPT-2 (Radford et al., 2019) is Transformer-based language model. Since it only has to predict one word at a time, only the decoder part of the Transformer is used. Its effectiveness comes from its size (100-1000M parameters) and amount of training data: 40GB of text from the internet. The authors showed that even though the model was trained as a language model it could be used for a plethora of NLP tasks, like question answering, summarization, and translation. This prompted (Wolf et al., 2018) who leveraged the powerful transfer learning capabilities of GPT-2 in adapting it to dialogue modelling. In this work, dialogue history is concatenated and fed to the model as input. As mentioned before this trivial approach has some downsides compared to the hierarchical approaches. They remedy this by adding special embeddings to the input which are different

for the two speakers. With these embeddings, the model sees which utterances are uttered by different speakers. In addition to the language modelling loss, they use a classification loss (next sentence prediction). The model has to classify whether a sentence is a probable next utterance. The model is simultaneously trained on both losses by sometimes getting incorrect and correct next targets (Figure 2.13).

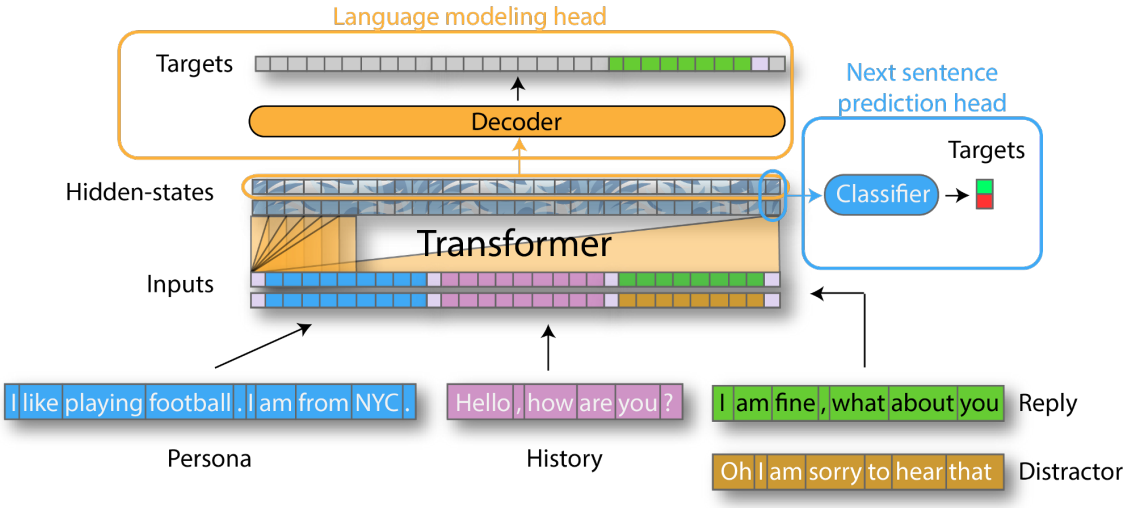

Figure 2.13: GPT based dialogue modeling (Wolf, 2019).

2.3.5 Additional Techniques

2.3.5.1 Objective Functions

In addition to the standard cross-entropy loss, various loss functions have been proposed for conversational models to generate more varied and interesting replies. One such class of loss functions are reinforcement learning-based, discussed in detail in Section 2.3.5.2. In (Ramachandran et al., 2016) a seq2seq model is pre-trained as a language model and to avoid overfitting, monolingual language modelling losses are added to the standard loss function. These losses act as a regularizer by forcing the seq2seq model to correctly model both the normal seq2seq task and the original monolingual language modelling tasks. In (Lison and Bibauw, 2017) it is argued that not all context-response pairs in a dialogue dataset have the same importance for conversational modelling. Accordingly, each context-response pair is weighted with a scoring model and these weights are used in the loss function when training a seq2seq model. Hence, examples associated with bigger weights in the dataset will have a larger impact on the gradient update steps through the loss function.

Approaches to incorporate beam search into the training procedure have also been explored. In (Wiseman and Rush, 2016) a simple approach is taken to construct the loss function by penalizing when the gold target sequence is not present in the beam consisting of the most likely predictions. The issue with incorporating beam search directly into the training procedure of seq2seq models is that it uses the argmax function which is not differentiable and hence is not amenable to backpropagation. Since the predictions, which serve as input to the standard loss function are discrete

(from beam search), the evaluation of the final loss is also discontinuous. A relaxation-based technique is proposed in (Goyal et al., 2017), where the standard loss function is gradually relaxed to the beam search based loss function as training progresses.

Perhaps the most extensively used objective function besides the standard one is based on Maximum Mutual Information (MMI) (Li and Jurafsky, 2015). In MMI the goal is defined as maximizing pairwise or mutual information between a source sentence S and the corresponding target T:

$$\log \frac{p(S,T)}{p(S)p(T)} = \log p(T|S) - \log p(T) \tag{2.21}$$

By introducing a weighting constant λ and by using the Bayes' theorem the training objective can be written in the following two ways:

$$\hat{T} = \arg\max_{T}[\log p(T|S) - \lambda \log p(T)] \tag{2.22}$$

$$\hat{T} = \arg\max_{T}[(1-\lambda) \log p(T|S) + \lambda \log p(S|T)] \tag{2.23}$$

However, it is not trivial to use these equations as loss functions in practice. In Equation 2.22 the term $\lambda \log p(T)$ acts as an anti-language model since it is subtracted from the loss function. This leads to ungrammatical output sentences in practice. Training using Equation 2.23 is possible, since the only requirement is to train an additional seq2seq model for the $\log p(S|T)$ term by using targets as inputs and inputs as targets, but direct decoding is intractable since it requires completion of target generation before $p(S|T)$ can actually be computed. Solutions to these issues have been proposed in (Li and Jurafsky, 2015).

2.3.5.2 Reinforcement Learning

Reinforcement learning (RL) (Sutton et al., 1998) is a type of learning framework which has seen increasing success recently (Mnih et al., 2013, Mnih et al., 2015, Silver et al., 2016). RL performs very well in tasks where there is no defined loss function or it is not known what the gold truth is. Instead, learning is implemented using a reward which is automatically given to the model based on its state. A general visual description of the reinforcement learning framework can be seen in Figure 2.14.

In RL an agent is some kind of function consisting of parameters that are optimized with a learning algorithm. The agent receives as input the state of the environment and can output actions based on it. The goal is to maximize the expected reward through a good combination of actions. The environment is the task or setting in which the agent is situated. One of the most popular applications of RL is to games since it is not known what the correct actions are in all steps of a game (Mnih et al., 2013). Instead, games have a clear end-goal and a learning agent can be rewarded based on whether it achieves the end-goal through its actions. An *episode* refers to a sequence of actions and states $(s_1, a_1, ..., s_t, a_t, s_T)$ until the agent reaches a terminal state s_T. The terminal state is referred to the state in which the agent receives a reward for the episode. This process, often called a Markov decision process (MDP)

consists of the triplet (S, A, R), where S is the set of states, A is the set of actions and R is the reward for the episode (Kandasamy et al., 2017). A policy π is a function that the agent implements to select an action to a given state. Stochastic policies $\pi(a|s)$ represent the probability of an agent executing action a in state s.

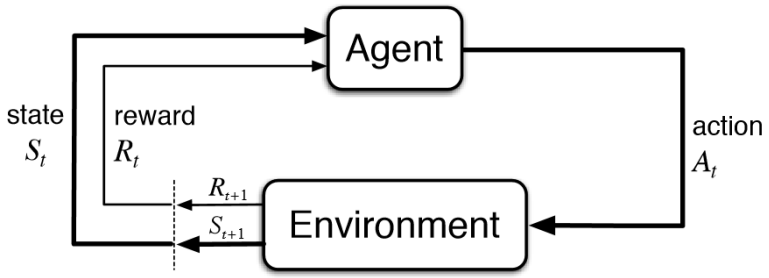

Figure 2.14: The reinforcement learning framework (Sutton et al., 1998).

The application of RL to neural conversational models is fairly straightforward. For a dialogue agent, the observed state at time-step t consists of the input sentence and what the agent has already outputted so far. The actions are the words in the vocabulary since at any time-step it can only produce a word from the vocabulary. When the agent finishes generating the reply a reward is observed, which can be simply the difference between the generated sentence and the gold truth. However, other more sophisticated rewards are usually used, which are described in the next paragraph. Thus, the agent has to maximize this reward through a sequence of actions, generating one word at each time-step. The policy according to which the agent takes actions can be implemented as a normal seq2seq model. Generally, to train RL dialogue agents the future reward of the episode has to be estimated at each time-step to get a reward for each action. This is usually done via the REINFORCE algorithm (Williams, 1992). However, this approach is not perfect for dialogue agents. Since rewards can only be observed at the end of an episode, all actions in that episode get the same reward. This means that if a response is of poor quality, like answering *I don't know* to the question *What's your name?* even the word *I*, which is mostly neutral will receive a negative reward. Instead as is customary in RL literature, a different reward for each action should be given. A possible solution is offered in (Kandasamy et al., 2017) using batch policy gradient methods. A different solution uses Monte Carlo search to sample tokens for partially decoded sentences (Li et al., 2017b). If there is a reward function, which is similar to a loss function, the agents can be trained by simple stochastic gradient descent.

Reward functions are often hand-crafted, meaning that the mathematical formulation of important properties of conversations is required in the beginning. In (Li et al., 2016d) the weighted sum of three different reward functions is used. The first one attempts to make responses easy to answer to. A set of dull and boring responses S is constructed, and if the response to the current action a is similar to these, then a is given a negative reward:

$$r_1 = -\frac{1}{N_S} \sum_{s \in S} \frac{1}{N_s} \log p_{seq2seq}(s|a) \tag{2.24}$$

where N_S denotes the cardinality of S and N_s denotes the number of tokens in s. $p_{seq2seq}$ represents the probability output from a standard seq2seq model. The second reward attempts to capture the information flow in a conversation. Each utterance should contribute with new information and it should not be repetitive compared to previous ones. In order to achieve such a reward the semantic similarity between sentences can be penalized by

$$r_2 = -\log \frac{\boldsymbol{h}_{p_i} \boldsymbol{h}_{p_{i+1}}}{||\boldsymbol{h}_{p_i}||||\boldsymbol{h}_{p_{i+1}}||} \tag{2.25}$$

where $\boldsymbol{h}_{p_i}$ and $\boldsymbol{h}_{p_{i+1}}$ denote hidden representations from the encoder RNN for two consecutive turns p_i and p_{i+1}. The final reward function focuses on semantic coherence. This rewards generated responses that are coherent and grammatical by taking into account the mutual information between action a and previous dialogue turns:

$$r_3 = \frac{1}{N_a} \log p_{seq2seq}(a|q_i, p_i) + \frac{1}{N_{q_i}} \log p^{backward}_{seq2seq}(q_i|a) \tag{2.26}$$

where $p_{seq2seq}(a|q_i, p_i)$ denotes the probability computed by a seq2seq model of generating response a given previous dialogue utterances $[p_i, q_i]$. $p^{backward}_{seq2seq}$ is a seq2seq model trained with swapped targets and inputs as discussed in Section 2.3.5.1. N_a and N_{q_i} are the number of tokens in utterances a and q_i respectively.

Similarly in (Yu et al., 2017), a linear combination of four reward functions is proposed. In this work, the functions implemented try to address turn-level appropriateness, conversational depth, information gain and conversation length to produce better quality responses. In (Yao et al., 2016) the reward function is based on calculating the sentence level inverse document frequency (Salton and Buckley, 1988) of the generated response.

However, hand-crafting reward functions are not ideal, since it might not be known exactly what conversational properties should be captured and how they should be formulated mathematically in an expressive way. A solution to this problem is to let agents figure out the appropriate reward by themselves, or use another agent to assign the reward. This has been explored before by using an adversarial approach (Li et al., 2017b). Generative adversarial networks (Goodfellow et al., 2014) consist of two networks competing with each other. A generator network (in this case a seq2seq model) tries to generate responses that mimic the training data and a discriminator network (implemented as an RNN based binary classifier in this case) tries to figure out whether the generated response came from the dataset or the generator network. Both of the networks are trained on whether the discriminator guessed correctly. Hence, the reward for the generator network comes from the discriminator network and represents whether the generator managed to fool the discriminator.

In (Havrylov and Titov, 2017) and (Kottur et al., 2017) a different line of research is explored using similar methods. The setting is still conversational, however, conversation arises from a two-agent collaborative game, and it is argued whether natural language-like dialogue can arise from such a setting. The sender agent (implemented as an RNN) receives as input an image and outputs a message, while the receiver agent (implemented as an RNN) receives as input this message and

a set of pictures K and outputs a probability distribution over the pictures. The probability for a picture $k \in K$ gives the likeliness of that picture being the one shown to the sender agent. Thus, the sender agent has to formulate a message which contains enough information so that the receiver can choose between the pictures. This is a collaborative RL setting since both agents receive the reward related to whether the receiver agent choose the right picture.

The approaches mentioned above are all off-line, since the reward is assigned by a predetermined reward function, and learning is based on a training dataset. However, on-line reinforcement learning has also been explored in a conversational setting (Li et al., 2016c). In online learning, the reward for generating a response is given by a human teacher interacting with the agent. The teacher can rate the response of the system on a scale and assign a numerical reward to it. This approach is very advantageous when a labelled dataset of dialogues is not available. The drawback is that human supervision is required.

2.3.5.3 Input Features

In addition to the raw dialogue turns a plethora of other inputs can be integrated into the seq2seq model. Several attempts have been made since the birth of the encoder-decoder model to augment it with additional input features. The goal of these features is mainly to provide more information about the conversation or the context. Models infused with additional features can learn to differentiate between various styles of dialogue. For example, by conditioning a seq2seq model on speaker information it can learn to output replies in the style of the speaker it was trained on (Li et al., 2016b). In this paper, the authors input additional speaker embeddings into the decoder. These speaker embeddings are similar to word embeddings, basically representing the speakers for the utterances in the dataset. The additional speaker embedding vector is fed into the decoder RNN at each time-step jointly with the previously generated word. The result is a more consistent conversational model, visualized in Figure 2.15. Because of the speaker embeddings, it learns general facts associated with the specific speaker. For example to the question *Where do you live?* it might reply with different answers depending on the speaker embedding that is fed into the decoder.

Additionally, the authors have also experimented with speaker-addressee models. This is a simple extension to the previously discussed speaker model. Instead of simply feeding in speaker embeddings to the decoder RNN, a weighted sum of the speaker and addressee embeddings is inputted. The addressee is the speaker to which the utterance is directed. The intuition behind this addition is that people talk differently depending on who they talk to (boss, friend, lover). Thus, the model not only responds differently with different speaker embeddings but it also conditions its response on the addressee embeddings. Hence, it might output a different reply for the same utterance and speaker embedding, but different addressee embedding.

Similarly, many other approaches have been made to condition the response generation of seq2seq models on various types of categories. For example in (Xing et al., 2017) topic words are extracted from each utterance in the dataset. Then, a seq2seq model is trained with these additional topic words together with the utterance they belong

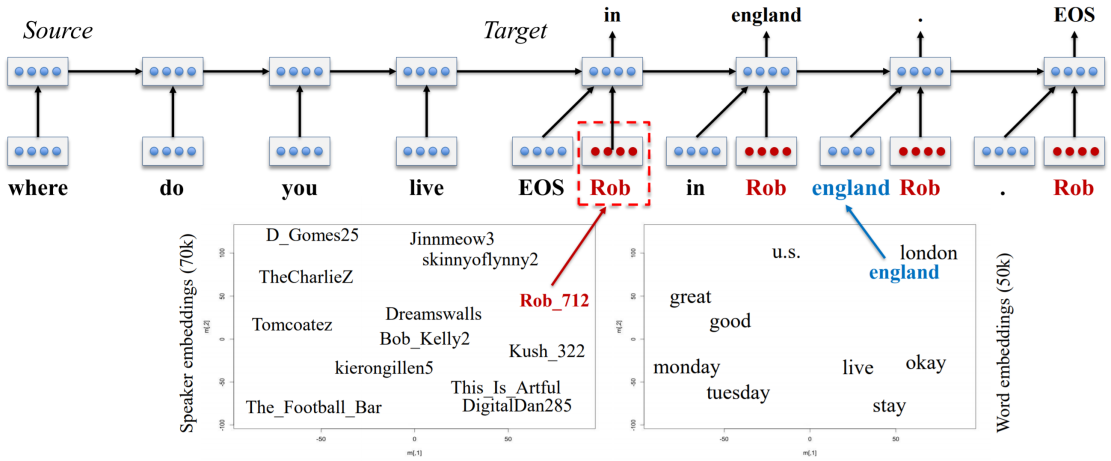

Figure 2.15: A seq2seq model augmented with speaker embeddings (Li et al., 2016b).

to. More specifically, a separate attention mechanism is implemented that attends over the topic words. The two vectors generated from the normal utterance attention and the topic attention are fed into the decoder RNN at each time-step. With this additional topic information, the model manages to produce more relevant replies to the topic of the conversation. A somewhat different approach to include topic information into the response generation process was employed in (Choudhary et al., 2017). In this work, three seq2seq models were trained separately on three datasets related to different topics (domains). Additionally, a domain classifier based on logistic regression was trained to compute the probability that the utterance is related to a domain, for all three domains. At test time, for an input utterance, all the seq2seq models generate a reply and the domain classifier predicts a domain. Then, a re-ranker takes the generated responses and the predicted domain probabilities and based on their product, determines the most probable reply-domain pair.

Another example involves using emotion categories for post-response pairs (Zhou et al., 2018a). Here the authors first classify all of the post-response pairs in the dataset with an LSTM model into several emotion categories like happy, sad, angry, etc. Then, they feed in these categories as real-valued vector representations into the decoder of an encoder-decoder model during training. Additionally, a more complex memory-based augmentation is proposed to the encoder-decoder model to better capture emotional changes in the response, which is not detailed here. In essence, by feeding in these emotion categories a reply conditioned on a specific type of emotion can be generated. For example, the response for the question *How was your day?* will differ in style for different emotion categories.

A different approach is taken in (Ghazvininejad et al., 2018), where the emphasis is put on taking into account relevant facts to an utterance. A seq2seq model is upgraded with a fact encoder operating over a knowledge base, which stores facts about various locations (eg. restaurants, theatres, airports). Before generating the reply to an encoded utterance, a location or entity is extracted from it with keyword matching or named entity recognition. Then, based on the location or entity, relevant facts are selected from the knowledge base, which are encoded by a memory network

(Sukhbaatar et al., 2015). Afterwards, the vector representation from the encoded facts and the vector representation from the encoded utterance are summed and fed into the decoder RNN of the encoder-decoder model which generates the response. The authors used Twitter post-reply-reply 3-turn conversations which included a handle or hashtag about locations for which relevant facts were selected from a knowledge-base. With this approach, the conversational model managed to produce more diverse and relevant replies if there was a location identified in the source utterance. Similar methods involve leveraging information from knowledge bases as additional inputs, described in detail in Section 2.3.5.4.

Other approaches try to integrate more features into the seq2seq model without using additional information. In a standard seq2seq model the only information the model has about the source utterance is through the vector representations of the words. To enrich the representation of the natural language utterances many other features have been proposed to be fed into the encoder RNN together with the word embeddings (Sordoni et al., 2015, Serban et al., 2017a, Serban et al., 2017b). Such features include part-of-speech tags, dialogue acts and n-gram word overlaps, to name a few.

2.3.5.4 Knowledge Bases and Task-Oriented Systems

Knowledge bases (KB) are powerful tools that can be used to augment conversational models. Since knowledge bases usually entail some kind of domain-specific information, these techniques are mainly used for task-oriented dialogue systems. In a KB, information related to the task at hand can be stored, for example, information about nearby restaurants or public transportation routes. Simple dictionaries or look-up-tables can be used to match an entity with information about it. Since KBs store information discretely, their integration with neural network-based encoder-decoder models is not trivial.

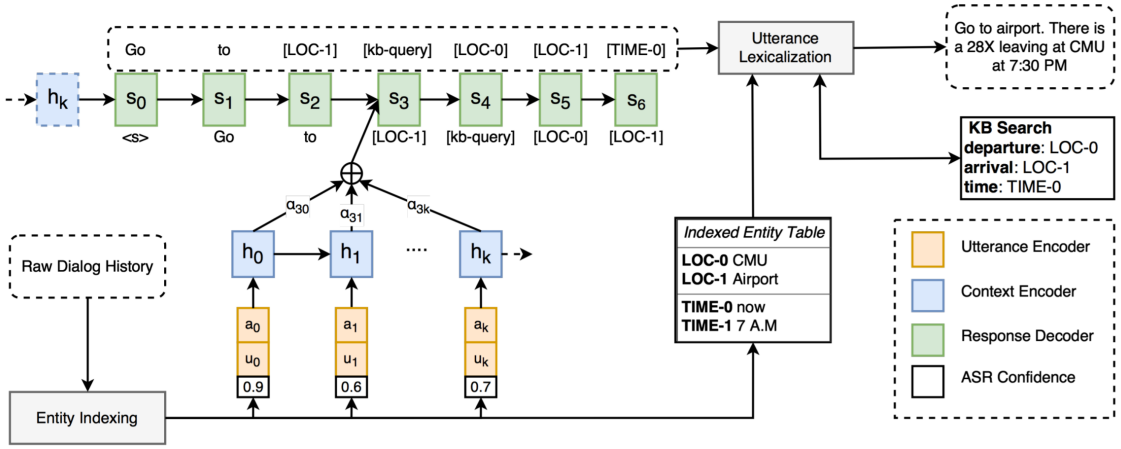

Figure 2.16: A HRED with attention upgraded with a KB component (Zhao et al., 2017a).

Perhaps the most straightforward integration method is proposed in (Zhao et al., 2017a). In this work, an entity indexing step is introduced before feeding in the

source utterance into a seq2seq model. This works by replacing locations or time specifying words with general tokens using named entity recognition or keyword matching. For example in the sentence *I want to go to Paris from New York*, the word *Paris* is replaced with the token *[LOC-1]* and *New York* is replaced with *[LOC-0]*. The connection between the general token and the original word is stored in a table. Then, the encoder-decoder model produces a response that also uses general tokens for locations and times and a special placeholder token for the KB result. Finally, the general tokens are transformed back to actual words using the stored table, a KB is employed which uses these general tokens to search for a route between the two places and its output is incorporated in the response. Thus, an open-domain dialogue system is achieved, which is augmented with a task-oriented, robust feature handling user requests related to finding routes between two locations. A visualization of the architecture of this model can be seen in Figure 2.16.

A similar, but more complex approach is taken in (Wen et al., 2016), where a KB augmented encoder-decoder model is used for the task of recommending restaurants. Here, besides a standard encoder RNN the source utterance is also processed with a belief tracker, implemented as a convolutional neural network (CNN). Belief tracking is an important part of task-oriented spoken dialogue systems (Henderson, 2015). The belief tracker network produces a query for a MySQL database containing information about restaurants. The final input to the decoder RNN is the weighted sum consisting of the last state of the decoder RNN and a categorical probability vector from the belief tracker. Then the decoder outputs a response in the same way as in the previous example, with lexicalised general tokens. These tokens are then replaced with the actual information that they point to in the KB. Similarly, in (Ghazvininejad et al., 2018) a more general fact-based KB is employed to augment an encoder-decoder model, which is presented in detail in Section 2.3.5.3.

Named entity detection is also used together with a KB to solve a different problem in conversational modelling (Li et al., 2016e). In this work, the authors try to solve the problem of users getting disinterested in the conversation with a dialogue agent. When the model detects that the user is bored (eg. writes ...) the normal dialogue system is replaced with a content introducing mechanism. More specifically, the last couple of utterances are searched to retrieve any named entities. Then, these entities are inputted to a KB, which contains associations between various entities, for example between a movie title and actor names in the movie. This information is then used to produce the response. Hence, the dialogue system can introduce new and relevant content based on a KB.

Task-oriented dialogue agents are a sub-type of conversational models. They are more narrow concerning language understanding and conversation topics, but they make up with robustness, making them better suited for commercial deployment. In task-oriented dialogue systems, there is usually a general task defined, and a goal to be achieved through conversation. A common property of task-oriented systems is that they learn from smaller and task-specific datasets. Since it is usually not required to generalize to other domains, learning on smaller, but more focused datasets gives much better performance. Furthermore, KB components are often used in combination with neural network models. KBs are an essential component of task-oriented dialogue systems since they provide the model with accurate and rich

information. Thus, neural network models only have to learn how to carry general conversations, how to form queries to KBs and how to incorporate the result from KB look-ups. Also, rule-based systems often perform equally well at task-oriented conversations, since only certain types of questions related to the tasks are expected from the user.

A prevalent task and goal is the recommendation of restaurants to users searching for specific types of cuisine. One of the most popular benchmark tasks is situated in this domain, named bAbi (Bordes et al., 2016, Joshi et al., 2017, Facebook, 2017). Since this is a simulated dataset, a rule-based system can achieve 100% task success rate. Nonetheless, it is an important benchmark for neural conversational models. The task is further divided into 5 sub-tasks to measure the performance on individual parts of the dialogue. Task 1 consists of a dialogue conducted until the first API call. Here the dialogue agent has to gather sufficient information from the user's utterances to call an API function. This function usually takes as input the type and location of the restaurant, the number of people for which the reservation should be made and the price range. Based on these inputs it automatically returns a list of candidate restaurants using a KB. Task 2 measures the ability of the dialogue agent to handle instances when the user forgot to mention an option or would like to change one. Then, in Task 3 the agent presents a possible candidate and converses with the user until the final restaurant is selected by the user. Finally, Task 4 measures the ability of the dialogue agent to provide additional information about the restaurant like the address or the phone number. Task 5 measures the overall success of the full dialogue.

A visualization of the different tasks is given in Figure 2.17. Various approaches to using KBs with neural models within the restaurant domain have been proposed (Wen et al., 2016, Eric and Manning, 2017, Williams et al., 2017). Other bAbi tasks have also been explored with neural models augmented with KBs (Williams et al., 2017, Li et al., 2016c), for example dialogue based reasoning (Weston et al., 2015) or movie recommendation tasks (Miller et al., 2016, Dodge et al., 2015). A more general recommendation based dialogue system is explored in (Yin et al., 2017). The system presented in this work uses an information extraction component which takes in a query and returns results from a search engine.

Several papers address the problem of integrating open-domain and task-specific approaches into one conversational model (Zhao et al., 2017a, Yu et al., 2017, Akasaki and Kaji, 2017). Essentially, the issue is that the bAbi toy tasks mentioned above do not represent real-life interactions between users and task-oriented dialogue systems. For conversational models to remain robust they have to able to handle out of domain utterances. Otherwise, users might get annoyed with the program, for example, many rule-based systems might resort to output responses like *Sorry, I couldn't understand that, but I am happy to talk about movies.* to get the user back on track. One approach to handle these issues is to build a model that can differentiate between in-domain and non-task user utterances. Then, two different models trained separately on open-domain and task-specific datasets can produce response candidates and a scoring function can output the most probable response (Akasaki and Kaji, 2017).

Another interesting line of research in conversational modelling is the integration of copying mechanisms to deal with out-of-vocabulary words (Eric and Manning,

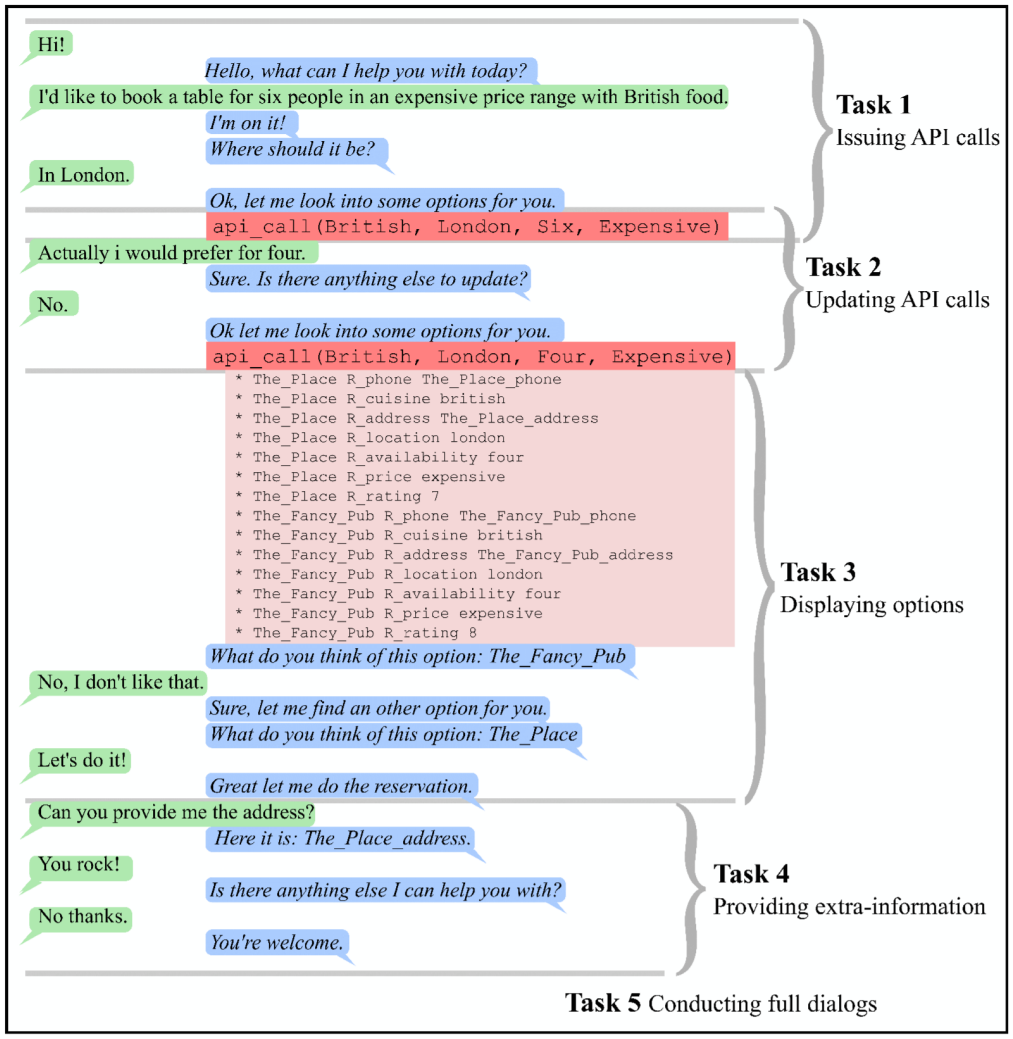

Figure 2.17: The 5 bAbi tasks for restaurant reservations. The dialogue agent's utterances are in blue, the user's utterances are in green and API function calls are in red (Bordes et al., 2016).

2017, Gu et al., 2016). In (Gu et al., 2016) the probability of generating a word $\boldsymbol{y}_t$ at time-step t is given by summing the probabilities from generate- and copy-modes. These probabilities are given by scoring functions. The scoring function of the generate-mode produces a score for each word in the vocabulary based on simple multiplication of the current decoder hidden state with a learned weight matrix. In contrast, the scoring function for copy-mode produces scores for each OOV word $\boldsymbol{x}_i$ in the source sentence. This scoring function is very similar to an attention mechanism:

$$f(\boldsymbol{x}_i) = \sigma(\boldsymbol{h}_i^\mathsf{T} W_c)\boldsymbol{s}_t \tag{2.27}$$

where $\boldsymbol{h}_i^\mathsf{T}$ is the hidden state of the encoder at step i, $\boldsymbol{s}_t$ is the hidden state of the decoder at the current time-step, W_c is a learned weight matrix and σ is a non-linear activation function. Using this copy-mode the model can efficiently handle OOV words and even integrate them into the response, making replies more diverse and unique.

2.4 Evaluation Methods

Evaluation of dialogue systems is perhaps a more controversial topic. Indeed it is still an open research problem to find good automatic evaluation metrics that can effectively compare the performance of conversational models. The traditional approach is to use the same metrics that are used for NMT and language modelling. Bleu and perplexity are widely used metrics for evaluating dialogue systems (Vinyals and Le, 2015, Yao et al., 2016, Zhao et al., 2017a, Serban et al., 2016). In the following paragraphs, we detail these and other metrics which we will use for evaluating our trained models.

Response length. Widely used as a simple engagement indicator (Serban et al., 2017c, Tandon et al., 2017, Baheti et al., 2018).

Word and utterance entropy. The per-word entropy $H_w = -\frac{1}{|U|}\sum_{w \in U} \log_2 p(w)$ of responses is measured to determine their non-genericness (Serban et al., 2017c). Probabilities are calculated based on frequencies observed in the training data. We introduce the bigram version of this metric, to measure diversity at the bigram level as well. Utterance entropy is the product of H_w and $|U|$, also reported at the bigram level.

KL divergence. We introduce the KL divergence between model and ground truth (GT) response sets to measure how well a model can approximate the GT distribution of words. Specifically, we define distributions p_{gt} and p_m based on each set of responses and calculate the KL divergence $D_{kl} = \frac{1}{|U_{gt}|}\sum_{w \in U_{gt}} \log_2 \frac{p_{gt}(w)}{p_m(w)}$ for each GT response. The bigram version of this metric is also reported.

Embedding metrics. Embedding *average*, *extrema*, and *greedy* are widely used metrics (Liu et al., 2016, Serban et al., 2017c, Zhang et al., 2018c). *average* measures the cosine similarity between the averages of word vectors of response and target utterances. *extrema* constructs a representation by taking the greatest absolute value for each dimension among the word vectors in the response and target utterances and measures the cosine similarity between them. Finally, *greedy* matches each response token to a target token (and vice versa) based on the cosine similarity between their embeddings and averages the total score across all words.

Coherence. We measure the cosine similarity between pairs of input and response (Xu et al., 2018b). Although a coherence value of 1 would indicate that input and response are the same, generally a higher value seems better as model responses tend to have lower coherence than targets.

Distinct metrics. *Distinct-1* and *distinct-2* are widely used in the literature (Li et al., 2016a, Shen et al., 2018a, Xu et al., 2018b), measuring the ratio of unique unigrams/bigrams to the total number of unigrams/bigrams in a set of responses.

However, they are very sensitive to the test data size, since increasing the number of examples in itself lowers their value.

Bleu. Measuring n-gram overlap between response and the target is widely used in the machine learning and dialogue literature (Shen et al., 2018a, Xu et al., 2018b). We report BLEU-1, BLUE-2, BLEU-3, and BLEU-4 computed with the 4th smoothing algorithm described in (Chen and Cherry, 2014). Similar to this, but less commonly used is the simple accuracy of the whole sentence or the word error rate, which is prevalent for speech recognition models (Shannon, 2017, Park et al., 2008). It calculates how many words are incorrect in the predicted sentence.

Perplexity (Manning and Schütze, 1999) is another common metric, which measures how well a probability model predicts a sample. Given a model, sentences can be inputted that weren't used during training and the perplexity on the test set can be computed. If a model q exists (encoder-decoder for conversational modelling), the perplexity can be computed by

$$2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 q(\boldsymbol{x}_i)} \tag{2.28}$$

where $\boldsymbol{x}_i$ are the test samples or words in the sentence and N is the length of the sentence. The goal is for models to assign a very high probability to test samples, meaning that the lower the perplexity score the better.

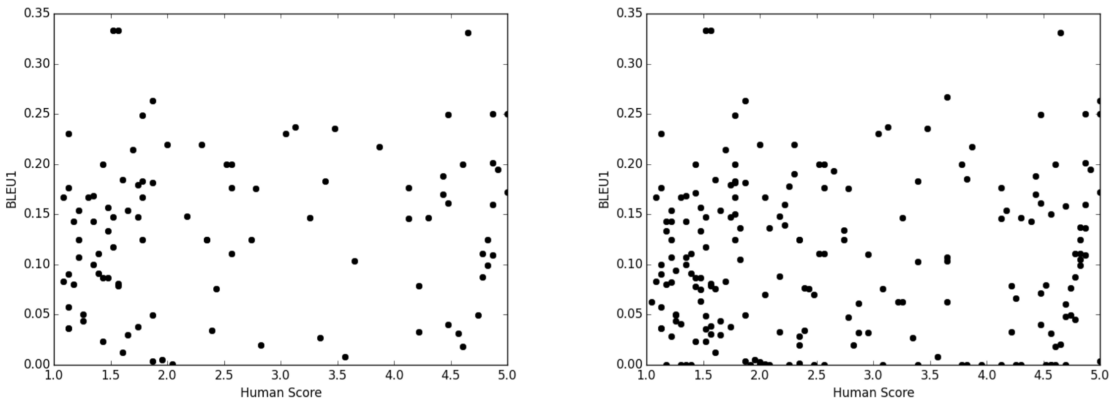

Figure 2.18: Bleu score correlation with human scores (Liu et al., 2016) on a Twitter corpus (left) and on the Ubuntu dialogue corpus (Lowe et al., 2015) (right).

It has been shown that standard metrics like perplexity and Bleu, do not correlate at all with human judgment for evaluation of dialogue agents (Tao et al., 2018, Liu et al., 2016). Figure 2.18 depicts the correlation of Bleu with human judgment for evaluating a seq2seq model trained on two different datasets. Attempts have been made in order to design better automatic evaluation metrics based on neural networks (Tao et al., 2018, Lowe et al., 2017, Li et al., 2017b). Since they are neural network-based they can be trained with labelled data. They receive a source utterance and the reply to it as input and they have to output a score. Then, this

score can be compared with the true score assigned by a human annotator and thus the model learns from its errors. As can be seen in Figure 2.19 these metrics indeed correlate better with human scoring. However, they are harder to conduct than simple metrics like BLEU or perplexity.

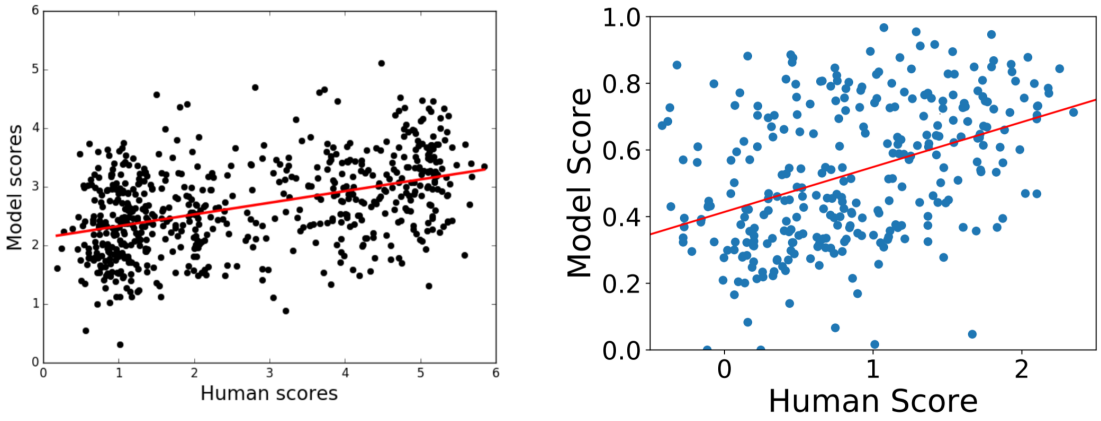

Figure 2.19: Correlation of ADEM (left) (Lowe et al., 2017) and RUBER (right) (Tao et al., 2018) with human scorers.

For reinforcement learning and task-oriented dialogue agents, evaluation is more straightforward. In reinforcement learning a goal that the dialogue agent has to achieve is specified, and its performance can be simply measured based on the per cent of cases in which it achieves the goal (Li et al., 2017b, Havrylov and Titov, 2017). Similarly for task-oriented dialogue agents usually there is a clearly defined task and the accuracy of accomplishing the given task serves as a good performance metric (Joshi et al., 2017, Zhao et al., 2017a, Li et al., 2016c).

Finally, one of the better methods to evaluate dialogue systems is to ask other people what they think about the quality of the responses that an agent produces. Human judgment has been one of the most prevalent metrics throughout recent literature related to conversational modeling (Shang et al., 2015, Vinyals and Le, 2015, Zhou et al., 2018a, Li et al., 2016d, Zhao et al., 2017a, Li and Jurafsky, 2015). Usually, human judges are asked to either rate the quality of individual responses from a model on a scale or to choose the better response from responses produced by different models for the same input. Several properties of conversations can be rated, like naturalness, grammaticality, engagement or simply the overall quality of the dialogue. However, even human evaluation has its downsides, like high variance, high cost, and difficulty of replicating experimental setups (Zhang et al., 2018b, Tao et al., 2018). Some works resort to human evaluations (Krause et al., 2017, Fang et al., 2018), others use automatic metrics only (Olabiyi et al., 2018, Xing and Fernández, 2018, Kandasamy et al., 2017, Shalyminov et al., 2018, Xu et al., 2018b), and some use both (Shen et al., 2018a, Xu et al., 2018a, Baheti et al., 2018, Ram et al., 2018).

Chapter 3

Evaluation and Loss Issues

Most open-domain NCMs are based on neural network architectures developed for machine translation (MT, (Sutskever et al., 2014, Cho et al., 2014, Vaswani et al., 2017)). Conversational data differs from MT data in that targets to the same source may vary not only grammatically but also semantically (Wei et al., 2017, Tandon et al., 2017): consider plausible replies to the question *What did you do today?*. Dialogue datasets also contain generic responses, e.g. *yes*, *no* and *i don't know*, that appear in a large and diverse set of contexts (Mou et al., 2016, Wu et al., 2018). Following the approach of modeling conversation as a sequence to sequence transduction of single dialog turns, these issues can be referred to as the *one-to-many*, and *many-to-one* problem. `seq2seq` architectures are not suited to deal with the ambiguous nature of dialogs since they are inherently deterministic, meaning that once trained they cannot output different sequences to the same input. Consequently they tend to produce boring and generic responses (Li et al., 2016a, Wei et al., 2017, Shao et al., 2017, Zhang et al., 2018a, Wu et al., 2018).

Previous approaches to the *one-to-many*, *many-to-one* problem can be grouped into three categories. One approach, discussed in Section 2.3.5.3, involves feeding extra information to the dialogue model such as dialogue history (Serban et al., 2016, Xing et al., 2018), categorical information like persona (Li et al., 2016b, Joshi et al., 2017, Zhang et al., 2018b), mood/emotion (Zhou et al., 2018a, Li et al., 2017c), and topic (Xing et al., 2017, Liu et al., 2017, Baheti et al., 2018), or through knowledge-bases (Dinan et al., 2019b, Ghazvininejad et al., 2018, Zhu et al., 2017, Moghe et al., 2018). A downside to these approaches is that they require annotated datasets which are not always available, or might be smaller in size. Augmenting the model itself, with e.g. latent variable sampling (Serban et al., 2017c, Zhao et al., 2017a, Zhao et al., 2018, Gu et al., 2019, Park et al., 2018, Shen et al., 2018b, Gao et al., 2019), or improving the decoding process (Shao et al., 2017, Kulikov et al., 2018, Mo et al., 2017, Wang et al., 2018) is also a popular approach (discussed in Section 2.3.2). Sampling provides a way to generate more diverse responses, however such models are more likely to output ungrammatical or irrelevant responses. Finally, directly modifying the loss function (Li et al., 2016a), or training by reinforcement (Li et al., 2016d, Serban et al., 2017b, Li et al., 2016c, Lipton et al., 2018, Lewis et al., 2017) or adversarial learning (Li et al., 2017b, Ludwig, 2017, Olabiyi et al., 2018, Zhang et al., 2018c) has also been proposed, but this is still an open research problem, as it

is far from trivial to construct objective functions that capture conversational goals better than cross-entropy loss. These approaches have been discussed in more detail in Section 2.3.5.1 and Section 2.3.5.2. Our approach to this problem will be discussed in Chapter 5.

The *one-to-many*, *many-to-one* problem may also have effects on evaluation. Normally metrics are computed at the validation loss minimum of a model, however in the case of chatbot models loss may not be a good indicator of response quality. Thus we looked at how our metrics progress during training of a Transformer model on the DailyDialog dataset. Figure 3.1 shows how coherence and the 3 embedding metrics saturate after about 80-100k steps (left graph), and never decrease (we ran the training for 300k steps, roughly 640 epochs). Most metrics show a similar trend of increasing until 100k steps, and then stagnating (see Figure 3.3, 3.4, and 3.5).

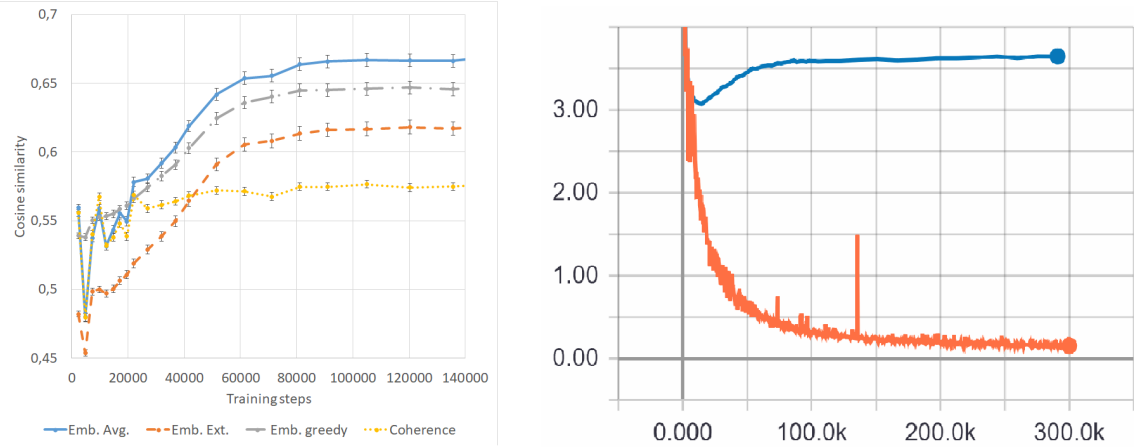

Figure 3.1: Embedding metrics and coherence (on validation data) as a function of the training evolution of `transformer` on DailyDialog data on the left. The evolution of the validation loss (blue) and train loss (orange) on the right for the same training.

In contrast, validation loss for the same training reaches its minimum after about 10-20k steps (Figure 3.1 right graph). This again suggests the inadequacy of the loss function, but it also questions the validity of these metrics, as they seem to favour a model that overfitted the training data, which we can assume after 640 epochs. Most interesting are embedding metrics and BLEU scores since they show that even after overfitting responses do not get farther from targets. This is in line with other findings reporting that qualitatively responses are better after overfitting (Csaky and Recski, 2017, Tandon et al., 2017), however occasionally they also tend to be too specific and irrelevant.

We found that this issue is partly because the DailyDialog official train and test splits overlap significantly. 20% of the examples in the test set appear in the training set as well. Because of this if the model overfits it will perform extremely well on that 20% which has a bigger impact than performing worse on the other examples. However, after removing this overlap we still observed this phenomenon of metrics increasing after overfitting.

Another issue we found with the popular distinct-1 and distinct-2 metrics is that they are very sensitive to the test data size since increasing the number of examples in itself lowers their value. While the number of total words increases linearly, the number of unique words is limited by the vocabulary, and we found that the ratio decreases even in human data (Figure 3.2). It is therefore important to only compare *distinct* metrics computed on the same test data.

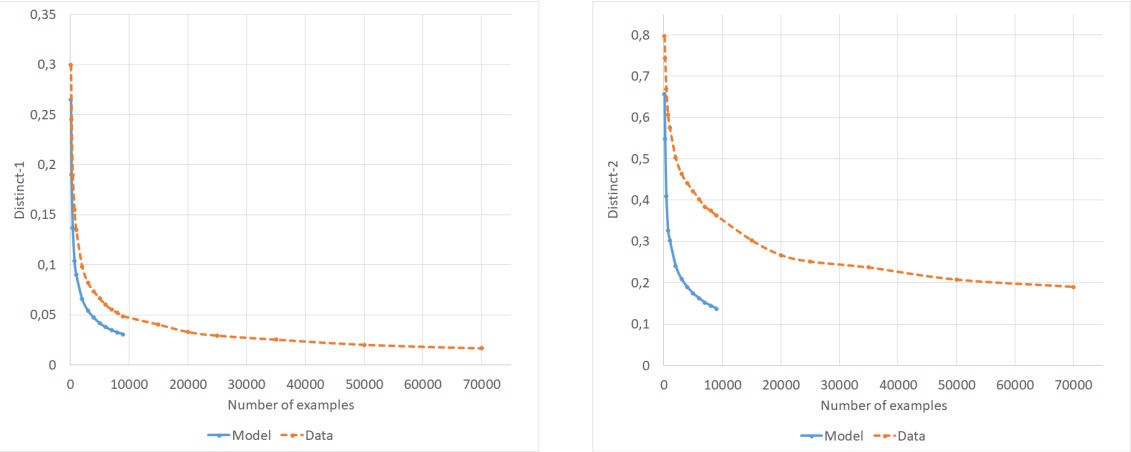

Figure 3.2: Distinct-1 (left) and Distinct-2 (right) metric with respect to number of test examples, on the DailyDialog dataset. Model responses were evaluated on 9000 examples only, since the rest were training examples.

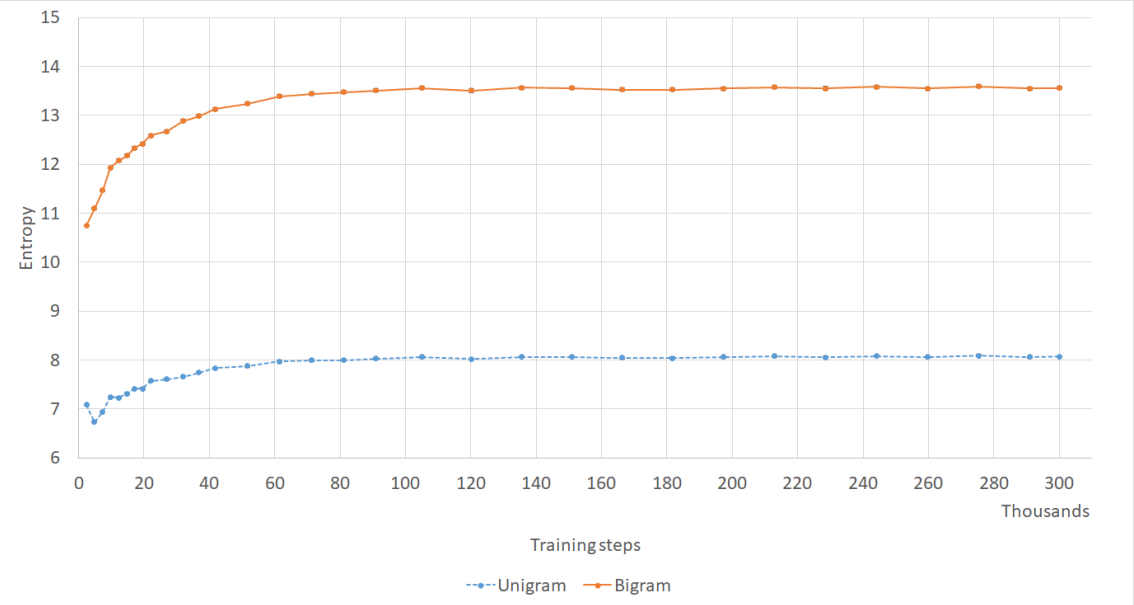

Figure 3.3: Word entropy of responses (computed on the validation set) with respect to the number of training steps of the `transformer` trained on DailyDialog.

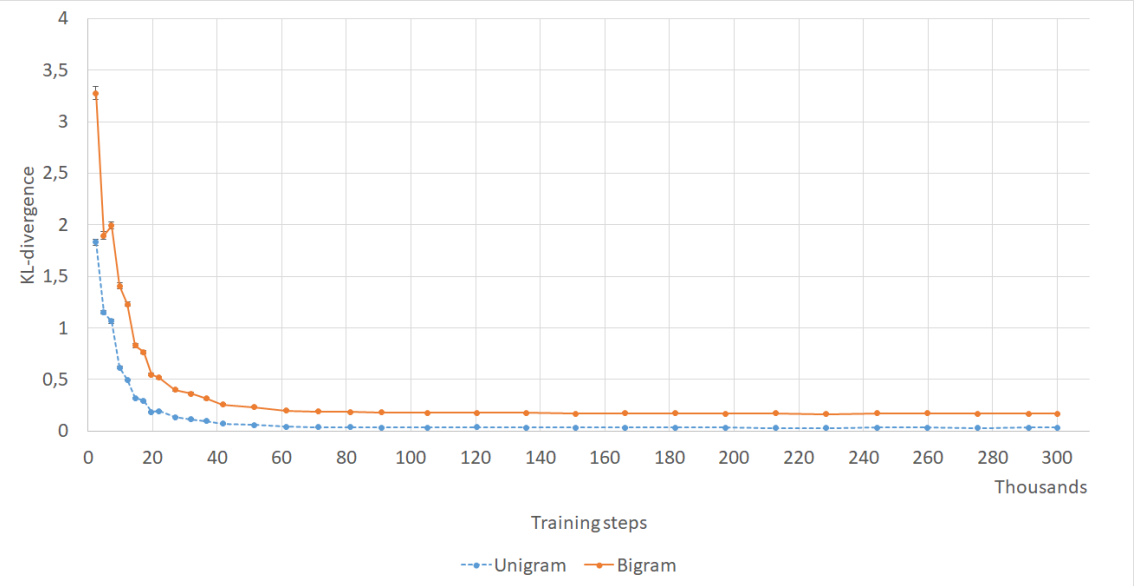

Figure 3.4: KL divergence of responses (computed on the validation set) with respect to the number of training steps of the `transformer` trained on DailyDialog.

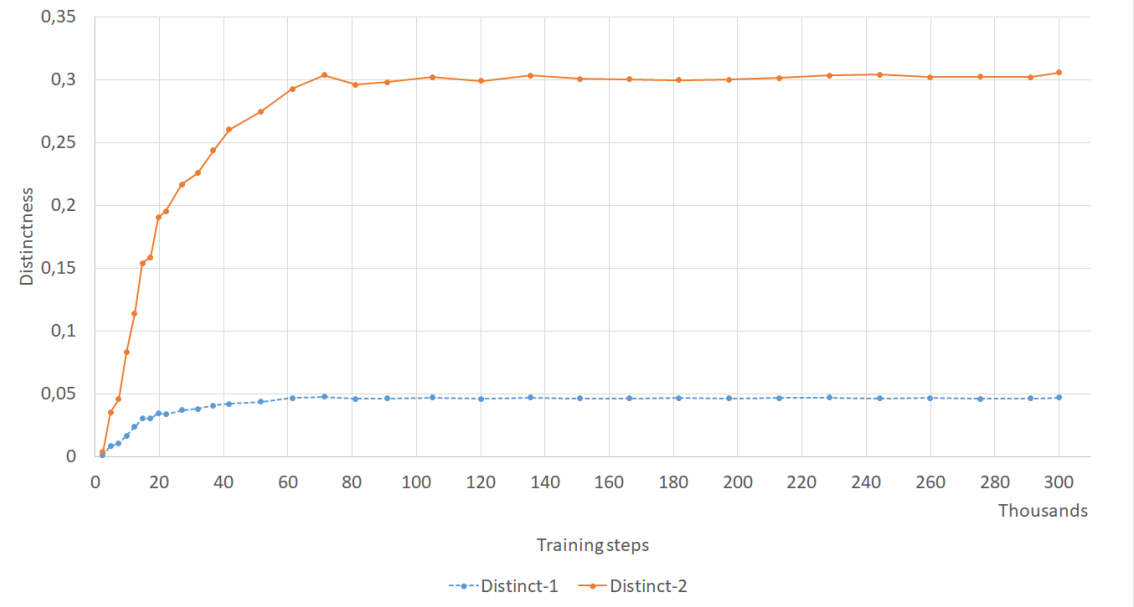

Figure 3.5: Distinct-1 and distinct-2 metrics (computed on the validation set) with respect to the number of training steps of the `transformer` trained on DailyDialog.

Chapter 4

The Gutenberg Dialogue Dataset

4.1 Dataset

4.1.1 Project Gutenberg

Most of Project Gutenberg's 60 000 online books are in English (47 300 books; 3 billion words), followed by a long tail of various languages. French, Finnish, and German, the most common foreign languages, contain 3000, 2000, 1750 books, and 194M, 74M, 82M words, respectively. Dutch, Spanish, Italian, Portuguese, and Chinese are all above 10M words. Book and word counts for the top 40 languages (out of 60) can be seen in Figure 4.1. We used the Gutenberg python package[1] to download books and query their license, language, and author metadata.

4.1.2 Pipeline

This section describes heuristics and methods used to extract dialogues from books and remove noise. Conversations in text are typically delimited by some special characters, such as quotation marks in English or em-dashes in Hungarian. The main challenges are identifying changes between speakers within a dialogue and separating sets of utterances that do not belong to the same dialogue, i.e. they have separate locations, time, or speakers. We develop simple heuristics that can extract relatively high-quality dialogues at scale. Tunable parameters of our system offer various trade-offs between data quality and size. Using our open-source system researchers can build custom versions of the Gutenberg Dialogue Dataset that best suit their applications.

Pre-filtering After downloading and separating by language, copyrighted books are removed. We filter books containing different or older language: if the KL divergence between a book's word distribution and the total (all books) distribution is above a threshold (2) it is removed. The method is less accurate for short books with less than 20 000 words, thus these are not filtered. In the English dataset, 2090

[1] `https://github.com/ageitgey/Gutenberg`

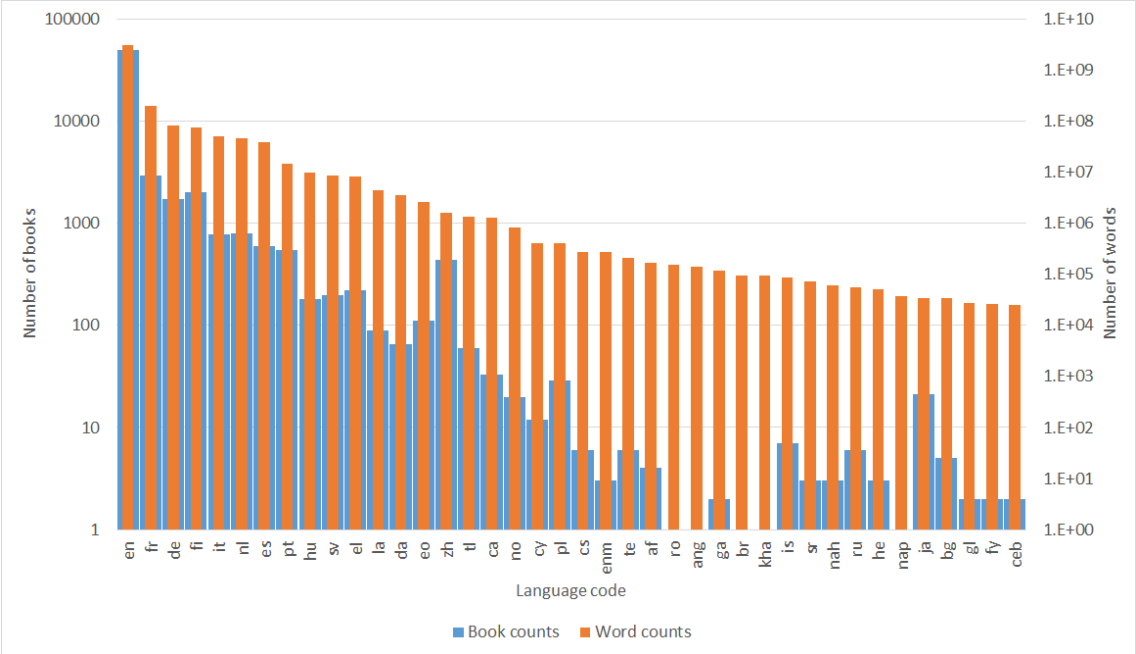

Figure 4.1: Number of books and words for top 40 languages on logarithmic scales.

books were removed (4.42%). By analyzing 100 filtered and 100 non-filtered books randomly, the precision and recall are 0.92 and 0.91, respectively.

Extracting Extracting high-quality dialogues from books is comprised of three main steps.

1. Conversational and non-conversational (narrative) text is separated.
2. Dialogic text is split into separate dialogues.
3. Dialogues are segmented into separate turns (utterances).

In most books conversational text is highlighted; e.g. placed between single/double quotation marks in English, or started by an em-dash in Hungarian. Naturally, these delimiters have other uses as well, but such cases are rare (about 5% of utterances; Section 4.1.3). Figure 4.2 shows a sample dialogue highlighting our heuristics.

Before dialogue extraction books with less than 150 delimiters per 10 000 words are removed. We assume that such books do not contain dialogues, and we empirically set this ratio by increasing it until the assumption starts failing (Figure 4.3). Since many books do not contain dialogues, almost half were removed (20 500) in the English pipeline. Sampling 100 filtered and non-filtered books, the precision and recall are 0.92 and 0.81, respectively. In a sample of the final dataset, less than 5% of utterances were non-conversational (Section 4.1.3).

If the number of characters in the non-conversational text between two dialogue segments highlighted by delimiters is high (above 150) they will not be considered part of the same dialogue. This heuristic, the dialogue gap, will always offer a false

"Read what I have written," she gasped. "It may be utterly unintelligible."

For answer, Morton folded the sheet and placed it in an envelope.

"Address this, if you please," he said.

She obeyed his request, limply forcing herself to make the effort; and, as the pen once more fell from her fingers, she glanced up at him with a haggard piteousness in her eyes.

"Will you not read what I have written?" she asked again.

"I see no reason why I should," he answered. "I have no wish to intrude. You are simply doing your duty towards your daughter; such a proceeding is not open to criticism."

Figure 4.2: A dialogue example. Utterances are in separate paragraphs, sometimes broken up by non-conversational text.

positive/negative trade-off since text length variability between dialogues is high. We tuned this trade-off by reasoning that shorter dialogues are less problematic than incoherent dialogues (3.5 times less false negatives as shown in Section 4.1.3). We observe a linear relationship between the dialogue gap and average dialogue length (Figure 4.4). In most books, utterances are in separate paragraphs, which provides our turn segmentation. This assumption fails in roughly 4% of utterance pairs as shown in Section 4.1.3.

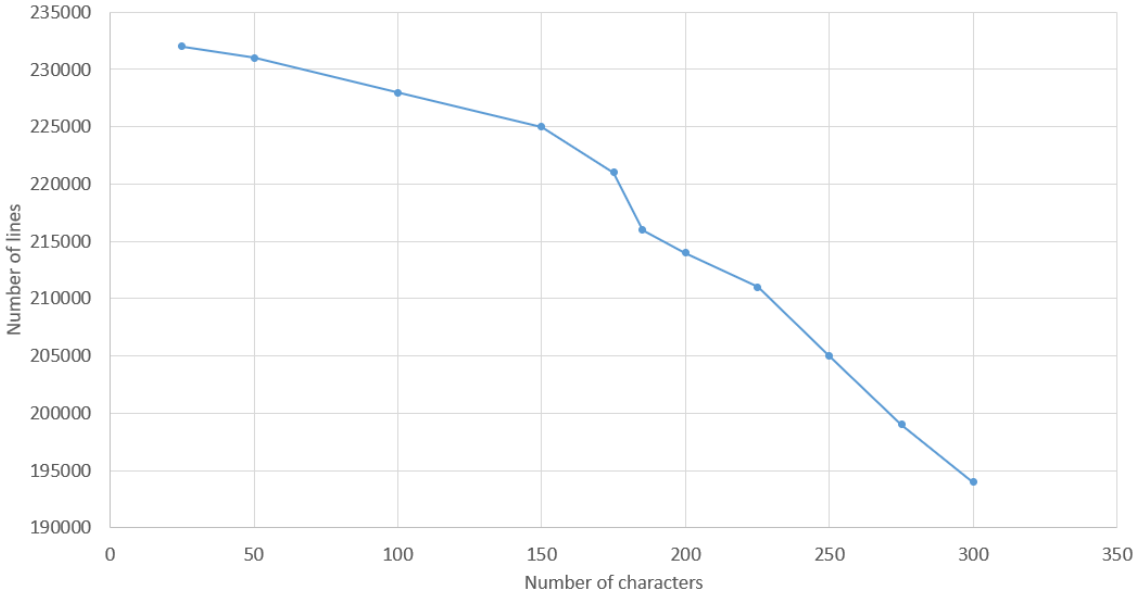

Figure 4.3: Number of utterances in a sample dataset with respect to the delimiter threshold (in characters).

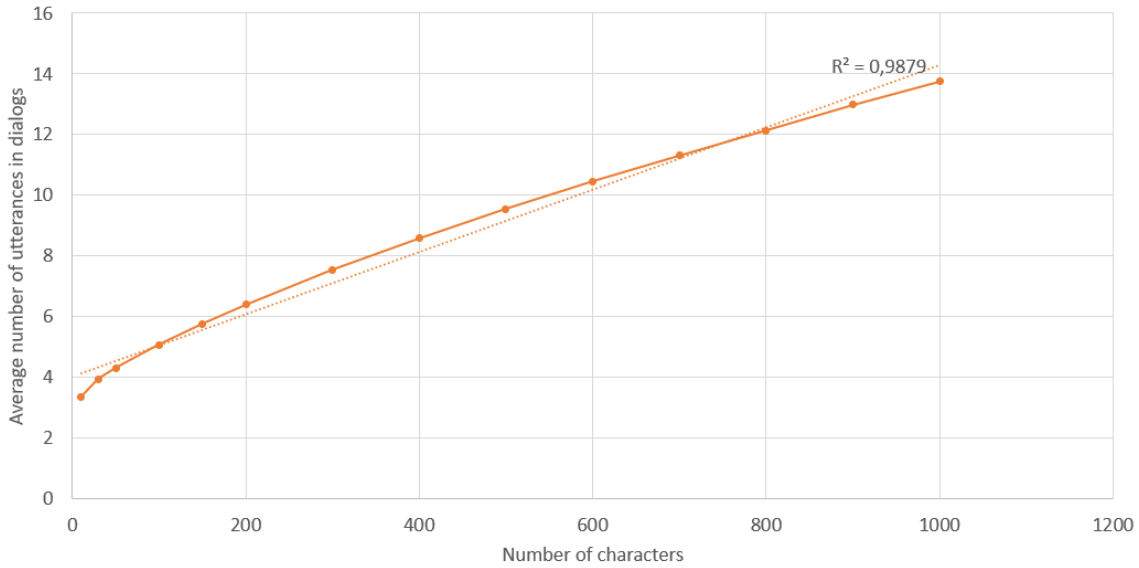

Figure 4.4: Average dialogue length with respect to the dialogue gap parameter (in characters).

Post-filtering During dialogue extraction utterances with more than 100 words are removed to ensure that remaining utterances are truly conversational and to facilitate neural model training (Dai et al., 2019). As all other parameters in the pipeline, this is adjustable to the needs of the person or task. We remove dialogues with more than 20% rare words (not in the top 100 000), removing noise and facilitating neural model training. Finally, dialogues are split randomly into train (90%), validation (5%), and test (5%) data. Dialogues from the same book are in exactly one split.

Method	**Parameter**	**Filtered**	**What**
Pre-filter	2 (KL-div)	2090 books (4.42%)	Old books and noise
Delimiter filter	150 delimiters / 10 000 words	20 500 books (43.3%)	Books with no dialogues
Long utterances	100 words	610 000 utterances (3.95%)	Non-conversational utterances
Post-filter	20% rare words	20 478 dialogues (0.8%)	Dialogues containing many rare words

Table 4.1: The various filtering steps for the English dataset.

Languages differ only in the dialogue extraction step. The modular pipeline can be easily extended to new languages by specifying conversational delimiters and a minimal implementation of dialogue and turn segmentation, generally adaptable from English. In some languages delimitation is less clear, inducing noise (e.g. a French utterance: "*–Eh bien! la mère, qu'est-ce que vous la vendez donc? demanda Buteau à la paysanne.*"). Delimiters and parameters for other languages were not analysed as profoundly as for English, leaving room for improvements in future work.

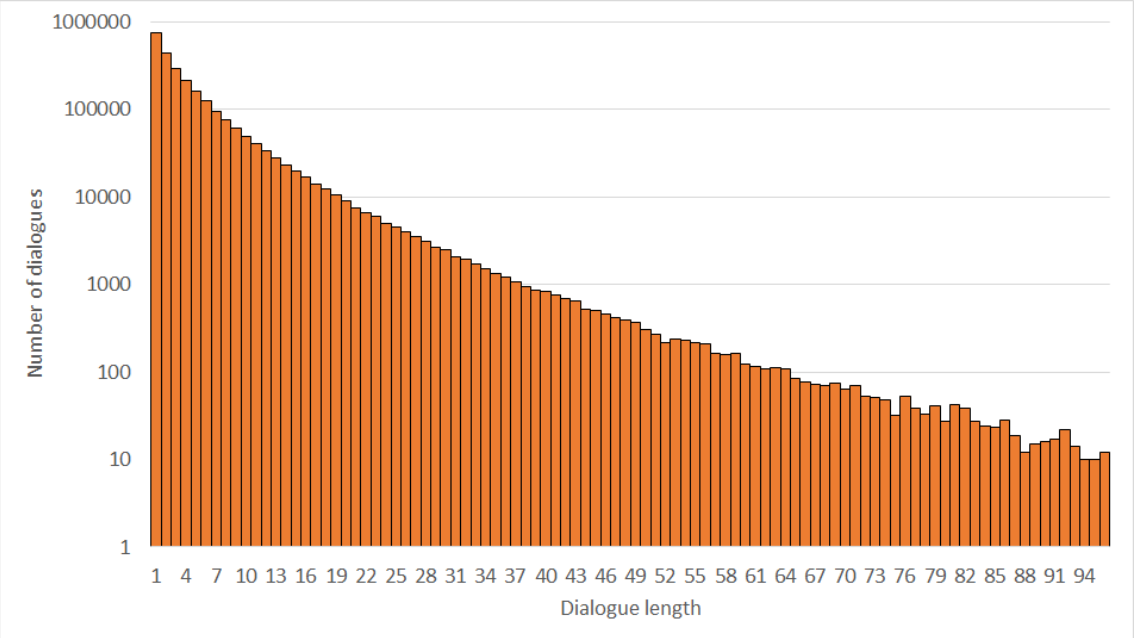

Figure 4.5: Distribution of dialogue lengths.

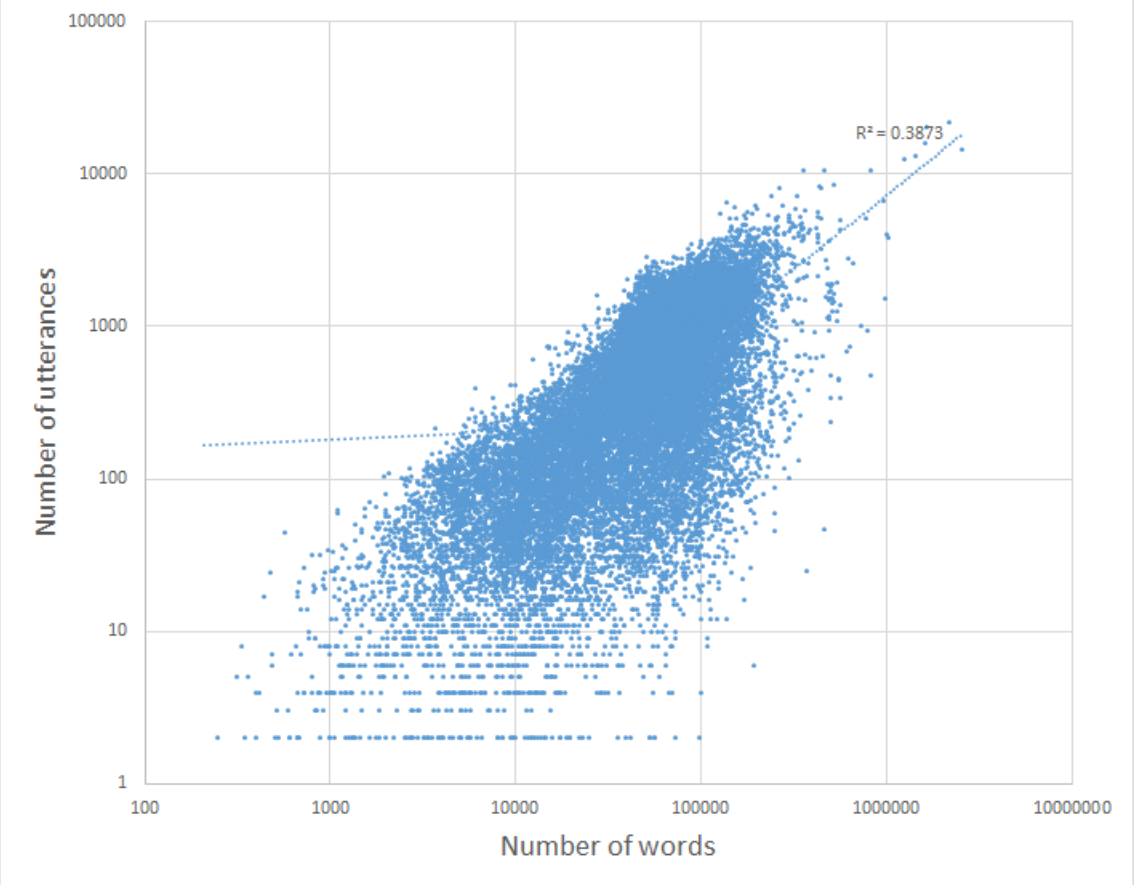

Figure 4.6: Number of extracted utterances with respect to number of words in each book on logarithmic scales.

We aim to show that good dialogue datasets can be constructed with minimal effort, as a first step towards a high-quality multi-language dataset ensemble.

In total, the four filtering steps removed about 12.5% of utterances (detailed in Table 4.1). 14 773 741 utterances were extracted (in 2 526 877 dialogues), with an average utterance and dialogue length of 22.17 and 5.85, respectively. The standard deviation of dialogue length in English is 6.09, and there are 87 500 dialogues with at least 20 utterances. The distribution of dialogue lengths can be seen in Figure 4.5. The average dialogue length can be linearly adjusted with the dialogue gap parameter as mentioned before. The number of dialogues extracted from each book can be seen in Figure 4.6.

4.1.3 Error Analysis

Utterance-level To asses single-turn quality in the English dataset we analyzed 100 random utterance pairs with book context. We found 2 major error types, remaining errors occurring in only 1% of cases. The extracted text is not conversational in 5% of pairs, a consequence of the delimiter threshold and other sources of noise. Utterances of a single speaker were falsely treated as multiple turns in 4% of cases, most often because of our "paragraph breaks signal dialogue turns" assumption (e.g. Figure 4.7).

> In his progress he passed the door of the dormitory of his victim—he paused a moment, and listened attentively. Then in a voice of deep anguish he said,—
>
> "She can sleep—she can sleep—no ghostly vision scares slumber from her eyes—while—"
>
> He shuddered, and passed a step or two on, then pausing again, he said,—
>
> "Oh, if she, the young and innocent—the loved of Heaven—if she would but bid me a 'good night,' I think I could sleep—I asked her once, and she would not—no, she would not give me so much peace; she would not say good night to me!"

Figure 4.7: Two consecutive turns uttered by the same speaker.

Dialogue-level Errors in whole dialogues exhibit a much greater variety. Based on a manual analysis of just 50 dialogues in the English dataset we identified 7 error categories (Figure 4.8). 32% of dialogues contained 0 errors, 42% contained 1 error type, 22% contained 2 types, remaining dialogues containing 3.

Utterances from the same conversation ended up in different dialogues frequently (34% of cases) because of the dialogue gap threshold (e.g. Figure 4.11). The inverse, a dialogue containing utterances from multiple conversations, occurred in 10% of cases (e.g. Figure 4.10). While it is challenging to set this parameter, we

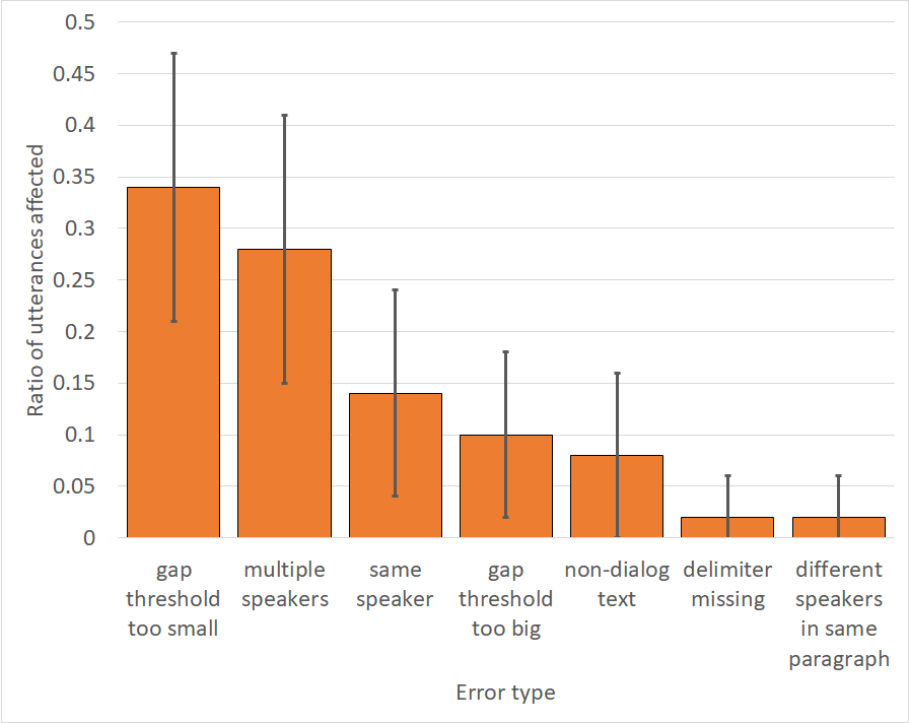

Figure 4.8: Ratio of dialogues affected by the various errors.

This fresh and clear morning, with a south wind blowing and a blue sky overhead, made even the back yard of Quiney's premises look cheerful, though the surroundings were mostly empty barrels and boxes. And he was singing, too, as he went on with his task; sometimes–

"Play on, minstrèl, play on, minstrèl, My lady is mine only girl;"

Figure 4.9: Non-dialogue text detected as an utterance.

think this is a good trade-off. Shorter dialogues equate to less data, which is less problematic, than dialogues containing utterances from multiple conversations, which are incoherent, thus affecting data quality. In Chapter 6 we discuss better ways to segment conversational text.

Books often contain dialogues between more than two speakers, our second most frequent error. However, such conversations are still coherent and provide useful data for model training. In contrast, the same speaker uttering at least two consecutive turns can break coherence in 14% of cases. Tackling these issues would have to involve speaker identification and increase the complexity of our pipeline (see Chapter 6). As in the utterance-level analysis, there were some dialogues (8%) in which non-conversational text got mixed in (e.g. Figure 4.9). The remaining errors, *delimiter missing* and *different speakers in same paragraph* occurred in only 1 dialogue out of 50.

"Carry pins, is it?" said Tom. "Ye can carry yer head level, me boy. So at it ye go, an' ye'll bate Rory fer me, so ye will."

"Well then," cried Barney, "I will, if you give me first choice, and I'll take Tom here."

"Hooray!" yelled Tom, "I'm wid ye." So it was agreed, and in a few minutes the sides were chosen, little Ben Fallows falling to Rory as last choice.

"We'll give ye Ben," said Tom, whose nerve was coming back to him. "We don't want to hog on ye too much."

"Never you mind, Ben," said Rory, as the little Englishman strutted to his place among Rory's men. "You'll earn your supper to-day with the best of them."

Figure 4.10: First three and last two utterances are not part of the same conversation, but they were merged because of the dialogue gap threshold.

Richard curbed an impatient rejoinder, and said quietly, "William Durgin had an accomplice."

Mr. Taggett flushed, as if Richard had read his secret thought. Durgin's flight, if he really had fled, had suggested a fresh possibility to Mr. Taggett. What if Durgin were merely the pliant instrument of the cleverer man who was now using him as a shield? This reflection was precisely in Mr. Taggett's line. In absconding Durgin had not only secured his own personal safety, but had exonerated his accomplice. It was a desperate step to take, but it was a skillful one.

"He had an accomplice?" repeated Mr. Taggett, after a moment. "Who was it?

Figure 4.11: A single conversation cut up because of the long paragraph between the two utterances.

4.2 Experiments

We conduct an extensive automatic evaluation using the DIALOG-EVAL repository[2] which implements 17 metrics used frequently in the literature (Csáky et al., 2019), discussed in Section 2.4. Metrics notations: Response length ($|U|$); per-word and per-utterance unigram (H^u_w, H^u_u) and bigram (H^b_w, H^b_u) entropy; unigram and bigram-level KL divergence (D^u_{kl}, D^b_{kl}); distinct-1 and distinct-2 (d1, d2); embedding metrics

[2] https://github.com/ricsinaruto/dialog-eval

average (AVG), *extrema* (EXT), and *greedy* (GRE); coherence (COH); the 4 BLEU metrics (b1, b2, b3, b4).

		$\|U\|$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	d1	d2
ZS	GUT	**7.5**	**6.93**	**11.7**	**53**	**73**	**.90**	**1.71**	**.0258**	**.111**
	OPEN	4.8	6.65	10.6	32	41	2.05	3.55	.0009	.002
FT	GUT	8.7	**7.09**	**11.8**	**63**	**90**	**.51**	**1.02**	**.0292**	**.147**
	OPEN	**8.8**	6.68	10.2	59	80	2.93	4.15	.0020	.005
DD	TRF	**9.9**	7.12	11.5	**72**	**95**	.89	1.60	.0132	.063
	RT	13.6	8.38	14.1	117	179	.04	.16	.0671	.411
	GT	13.8	8.37	13.7	118	153	0	0	.0635	.408

Table 4.2: Entropy and distinct metrics computed on the test set of DailyDialog. Pre-trained models on Gutenberg (GUT) and Opensubtitles (OPEN) are compared. TRF is a Transformer trained only on DailyDialog, evaluated at the validation loss minimum. RT refers to randomly selected responses from the DailyDialog training set, and GT to the ground truth response set. Significantly better results (95% confidence interval) are highlighted separately for the zero-shot (ZS) and finetuned (FT) scenarios. The TRF row is only highlighted if significantly better than every pre-trained setting.

		AVG	EXT	GRE	COH	b1	b2	b3	b4
ZS	GUT	**.638**	.463	**.638**	**.683**	**.098**	**.095**	**.091**	**.083**
	OPEN	.606	.466	.610	.604	.075	.068	.063	.056
FT	GUT	**.674**	**.479**	**.659**	**.701**	**.140**	**.132**	**.126**	**.115**
	OPEN	.645	.466	.625	.643	.106	.117	.118	.110
DD	TRF	.663	.461	.642	.666	.127	.128	.127	.117
	RT	.667	.390	.604	.666	.086	.117	.127	.122
	GT	1	1	1	.717	1	1	1	1

Table 4.3: Embedding and BLEU metrics computed on the test set of DailyDialog. Pre-trained models on Gutenberg (GUT) and Opensubtitles (OPEN) are compared. TRF is a Transformer trained only on DailyDialog, evaluated at the validation loss minimum. RT refers to randomly selected responses from the DailyDialog training set, and GT to the ground truth response set. Significantly better results (95% confidence interval) are highlighted separately for the zero-shot (ZS) and finetuned (FT) scenarios. The TRF row is only highlighted if significantly better than every pre-trained setting.

		$\|U\|$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	d1	d2
ZS	GUT	**8.3**	**6.99**	**11.9**	**58.8**	**83**	**.98**	**2.21**	**.0159**	**.078**
	OPEN	6.6	6.70	11.5	45.2	67	2.02	2.84	.0005	.001
FT	GUT	**11.0**	**6.49**	**10.4**	**72.4**	**105**	**1.26**	**2.13**	**.0104**	**.048**
	OPEN	10.6	6.37	10.1	68.3	98	2.58	2.66	.0011	.002
PC	TRF	11.1	6.89	11.0	**76.6**	**111**	1.28	2.21	.0047	.018
	RT	11.6	8.50	14.0	98.5	148	.03	.14	.0489	.349
	GT	11.5	8.46	13.4	97.9	126	0	0	.0416	.336

Table 4.4: Entropy and distinct metrics computed on the test set of PersonaChat. Pre-trained models on Gutenberg (GUT) and Opensubtitles (OPEN) are compared. TRF is a Transformer trained only on PersonaChat, evaluated at the validation loss minimum. RT refers to randomly selected responses from the PersonaChat training set, and GT to the ground truth response set. Significantly better results (95% confidence interval) are highlighted separately for the zero-shot (ZS) and finetuned (FT) scenarios. The TRF row is only highlighted if significantly better than every pre-trained setting.

		AVG	EXT	GRE	COH	b1	b2	b3	b4
ZS	GUT	**.620**	.478	.617	**.680**	.091	.092	.091	.084
	OPEN	.607	**.497**	.617	.611	**.094**	**.098**	**.095**	**.088**
FT	GUT	**.653**	**.500**	**.651**	**.713**	**.165**	**.163**	**.164**	**.155**
	OPEN	.580	.482	.624	.585	.148	.151	.154	.146
PC	TRF	**.658**	.488	.642	.695	.164	.163	.165	.156
	RT	.672	.422	.589	.671	.099	.127	.136	.131
	GT	1	1	1	.717	1	1	1	1

Table 4.5: Embedding and BLEU metrics computed on the test set of PersonaChat. Pre-trained models on Gutenberg (GUT) and Opensubtitles (OPEN) are compared. TRF is a Transformer trained only on PersonaChat, evaluated at the validation loss minimum. RT refers to randomly selected responses from the PersonaChat training set, and GT to the ground truth response set. Significantly better results (95% confidence interval) are highlighted separately for the zero-shot (ZS) and finetuned (FT) scenarios. The TRF row is only highlighted if significantly better than every pre-trained setting.

In all experiments, a Transformer is trained on utterance pairs using the TENSOR2TENSOR repository[3]. We used the Adam optimizer (Kingma and Ba, 2014) and

[3] https://github.com/tensorflow/tensor2tensor

hyperparameters can be seen in Table 4.6. The vocabulary is set to the top 100 000 words for Gutenberg and Opensubtitles trainings. On these datasets, models were trained for 21 epochs, because of time and hardware constraints, but the validation loss was still decreasing. Training took about 80 hours on a single RTX 2080 Ti, with batch size set to the memory limit. We evaluate Gutenberg and Opensubtitles pre-trained models in zero-shot and finetuning scenarios on DailyDialog and PersonaChat. After pre-training for 21 epochs on the same amount of data, models are finetuned until the validation loss minimum on target data.

Gutenberg pre-training performs better than Opensubtitles on DailyDialog across nearly all metrics in both zero-shot and finetuned settings (Table 4.2 and Table 4.3). On some metrics (e.g. KL-div and embedding metrics) Gutenberg pre-training outperforms even the model trained only on DailyDialog. There are only three metrics in which the baseline DailyDialog training is significantly better than the finetuned model, pointing to the effectiveness of pre-training on Gutenberg.

Name	**Value**
Hidden size	512
Number of hidden layers	6
Label smoothing	0.1
Filter size	2048
Number of attention heads	8
Layer dropout	0.1
Relu dropout	0
Attention dropout	0
Learning rate	0.2
Learning rate warmup steps	8000

Table 4.6: Transformer hyperparameters.

	$\|U\|$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	AVG	EXT	GRE	COH	d1	d2	b1	b2	b3	b4
trf	9.4	7.30	12.4	69	103	1.03	2.16	.654	.447	.621	.713	.00216	.020	.0651	.0650	.063	.0582
rt	21.9	9.16	16.2	201	343	.03	.08	.694	.377	.591	.695	.00654	.136	.0578	.0877	.100	.0992
gt	21.8	9.18	16.1	201	329	0	0	1	1	1	.730	.00547	.132	1	1	1	1

Table 4.7: Metrics computed on the test set of Gutenberg. TRF is the trained Transformer, RT refers to randomly selected responses from the training set, and GT is the ground truth response set.

Gutenberg pre-training achieves better results than Opensubtitles in all metrics after finetuning on PersonaChat (Table 4.4 and Table 4.5). In the zero-shot scenario, Opensubtitles achieves better BLEU scores, however, both scores are much lower than randomly selected responses, questioning the validity of this comparison. Gutenberg pre-training outperforms the baseline PersonaChat training on some metrics after finetuning, further testament to Gutenberg's usefulness. Considering the domain

mismatch between the older Gutenberg books and the modern chit-chat style datasets this is especially impressive.

Table 4.7 presents the Gutenberg training's results on its own test set. In some metrics, it performs worse than random responses from the training set. This is expected for entropy and distinct metrics, but we think that BLEU scores would be higher after further training. Overfitted models have been shown to perform better on these metrics (Csaky and Recski, 2017, Csáky et al., 2019), thus many more epochs could be required for optimal performance. This lack of stopping criteria also makes a fair comparison challenging. Sample responses to test inputs from Gutenberg can be seen in Table 4.8.

Input	Model Response	Target
i will ! i will ! i will ! i will !	you will not ! you will not ! you will not ! you will not ! you will not ! you will not ! you will n	no , no ! no , no !
oh , please go away ! i - - i do n't want any help . i 'm only looking for crabs .	i 'm going to help you ,	mais , mademoiselle permettez - moi , je vous prie , ah , pauvre petite ! pauvre petite !
your nephew ? your ladyship never told me you had a nephew .	i have never heard of him ,	i must have had it on the tip of my tongue to tell you , over and over again , but we have had so ma
you ai n't going to hit my mother , grandfather !	i 'm going to hit your grandfather ,	no , no , lad i learned my manners better than that ! whatever i was going to say , i was thinking o
and here is the chance ,	i do n't know , i do n't know , i do n't know .	the chance is here ,
it is pleasant when a character we admire is consistent .	it is a book that will be read with interest .	margaret is very consistent .
mammy , of course ,	i 'm glad you 're going to be a good girl ,	and whats your name ?
for looking out for myself . for my part , i had thought it praiseworthy , but he says none of the r	i dont know , i dont know . i dont know . i dont know . i dont know . i dont know . i	rotifer , as before .
did england belong to the english before the jews came ?	no , sir .	how do you mean ? of course it did
certainly not . i am quite independent .	i am glad to hear it .	so i should have supposed . then why not come ?
you were a thousand miles away . i passed you twice and you did n't even see me . if it had n't been	i 'm glad you did n't , i 'm glad you did n't .	i 'm very glad you did n't . i really wanted to talk to you .
that 's it , that 's it , wo n't you risk a look - in at the exposition du lis with us , monsieur ge	i do n't know , i do n't know .	well , no , i want to walk . i shall go with monsieur l'abbe froment to the chamber .
why , yes i think i can .	you can ?	i am not jesting i am suffering , rosalind . you will not leave that girl alone with him ?
what is it , john ?	i do n't know ,	good news , wife , such news as i never hoped for or even dreamed of .
that 's your own statement ! fosdick never had brains enough to figure a thing out like that . all h	i do n't know what you mean ,	so you see , crime does not pay . the net has closed over your head . you erred a score of times . y
kate morton ! why , i have n't seen her for ten years !	i do n't know what you mean ,	was it a hopeless affection , then ? are you only true to her memory ?
twenty - five dollars , well , it 's fortunate that i have them . and who are you ? not one of campb	i am a poor man , sir , and i have no money .	i am a confidential messenger , i carry messages and execute commissions that require more or less d

Table 4.8: Model responses and targets to 17 randomly selected test inputs. All utterances are truncated to 100 characters.

Chapter 5

Data Filtering

5.1 Methods

5.1.1 Intuition

Improving dataset quality through filtering is frequently used in the machine learning literature (Sedoc et al., 2018, Ghazvininejad et al., 2018, Wojciechowski and Zakrzewicz, 2002) and data distillation methods in general are used both in machine translation and dialogue systems (Axelrod et al., 2011, Li et al., 2017a). (Xu et al., 2018b) introduced coherence for measuring the similarity between contexts and responses, and then filtered out pairs with low coherence. This improves datasets from a different aspect and could be combined with our present approach. However, natural conversations allow many adequate responses that are not similar to the context, thus it is not intuitively clear why filtering these should improve dialogue models.

We approach the *one-to-many*, *many-to-one* problem (discussed in Chapter 3) from a relatively new perspective. Instead of adding more complexity to NCMs, we reduce the complexity of the dataset by filtering out a fraction of utterance pairs that we assume are primarily responsible for generic/uninteresting responses. Of the 72 000 unique source utterances in the DailyDialog dataset (see Section 5.2.1 for details), 60 000 occur with a single target only. For these, it seems straightforward to maximize the conditional probability $P(T|S)$, S and T denoting a specific source and target utterance. However, in the case of sources that appear with multiple targets (*one-to-many*), models are forced to learn some "average" of observed responses (Wu et al., 2018).

The entropy of response distribution of an utterance s is a natural measure of the amount of "confusion" introduced by s. For example, the context *What did you do today?* has high entropy, since it is paired with many different responses in the data, but *What colour is the sky?* has low entropy since it's observed with few responses. The *many-to-one* scenario can be similarly formulated, where a diverse set of source utterances are observed with the same target (e.g. *I don't know* has high entropy). While this may be a less prominent issue in training NCMs, we shall still experiment with excluding such generic targets, as dialogue models tend to

generate them frequently. Figure 5.1 visualizes the distributions of these low- and high-entropy examples.

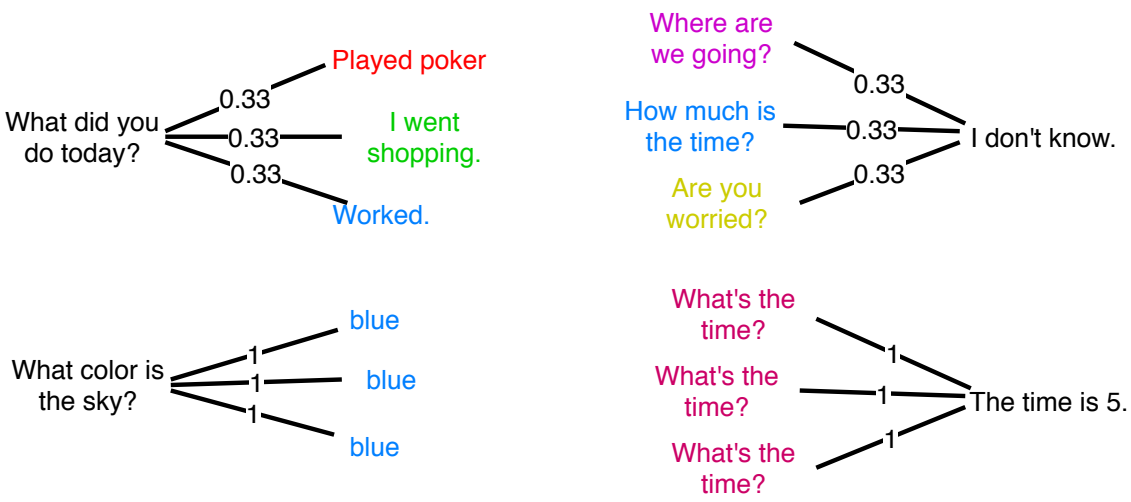

Figure 5.1: Visualization of example utterance pair distributions. Upper row is high-entropy, while lower row is low-entropy. Left column is the *one-to-man* problem, while right column is the *many-to-one* problem.

5.1.2 Clustering Methods and Filtering

We refer with IDENTITY to the following entropy computation method. For each source utterance s in the dataset we calculate the entropy of the conditional distribution $T|S = s$, i.e. given a dataset D of source-target pairs, we define the *target entropy* of s as

$$H_{\text{tgt}}(s, D) = -\sum_{(s,t_i)\in D} p(t_i|s) \log_2 p(t_i|s) \tag{5.1}$$

Similarly, *source entropy* of a target utterance is

$$H_{\text{src}}(t, D) = -\sum_{(s_i,t)\in D} p(s_i|t) \log_2 p(s_i|t) \tag{5.2}$$

The probabilities are based on the observed relative frequency of utterance pairs in the data (Figure 5.1).

For this entropy-based filtering, we considered the possibility of also including some form of similarity measure between utterances that would allow us to detect whether a set of responses is truly diverse, as in the case of a question like *What did you do today?*, or diverse only on the surface, such as in the case of a question like *How old are you?* (since answers to the latter are semantically close). Measuring the entropy of semantic clusters as opposed to individual utterances may improve our method by reducing data sparsity. For example *How are you?* can appear in many forms, like *How are you <name>?* (see Section 5.2.2). While the individual forms have low entropy (because they have low frequency), we may decide to filter them

all if together they form a high-entropy cluster. Figure 5.2 visualizes cluster-level distributions for these examples.

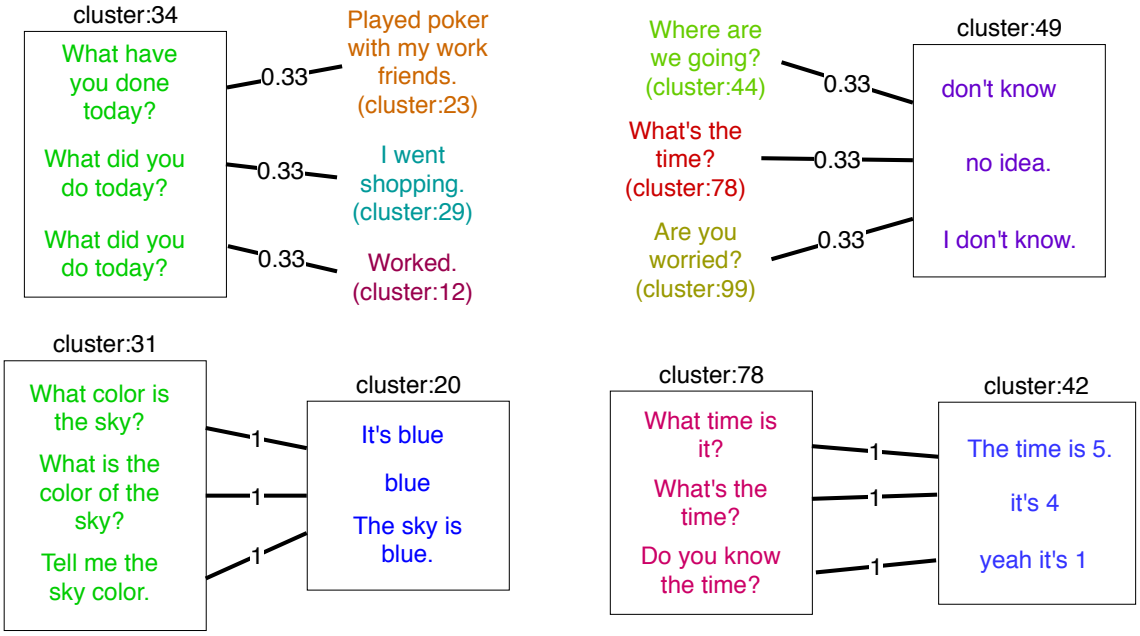

Figure 5.2: Visualization of example cluster pair distributions. Upper row is high-entropy, while lower row is low-entropy. Left column is the *one-to-man* problem, while right column is the *many-to-one* problem.

To this end we performed the filtering based not only on the set of all utterances, as in the case of IDENTITY, but also on clusters of utterances established by clustering their vector representations using the Mean Shift algorithm (Fukunaga and Hostetler, 1975). Source and target utterances are clustered separately. In the AVG-EMBEDDING setup the representation $R(U)$ of utterance U is computed by taking the average word embedding weighted by the smooth inverse frequency $R(U) = \frac{1}{|U|} \sum_{w \in U} \frac{E(w) \cdot 0.001}{0.001 + p(w)}$ of words (Arora et al., 2017), where $E(w)$ and $p(w)$ are the embedding and the probability[1] of word w respectively. We also experiment with SENT2VEC[2], a more sophisticated sentence embedding approach, which can be thought of as an extension of word2vec to sentences (Pagliardini et al., 2018).

The *target entropy* of a source cluster c_s is

$$H_{\text{tgt}}(c_s, C) = -\sum_{c_i \in C} p(c_i|c_s) \log_2 p(c_i|c_s) \tag{5.3}$$

where C is the set of all clusters and $p(c_i|c_s)$ is the conditional probability of observing an utterance from cluster i after an utterance from cluster s (Figure 5.2. In the context of these methods, the entropy of an utterance will mean the entropy of its cluster. Note that IDENTITY is a special case of this cluster-based entropy computation method, since in IDENTITY a "cluster" is comprised of multiple examples

[1]Based on the observed relative frequency in the data.

[2]`https://github.com/epfml/sent2vec`

of one unique utterance. Thus a target cluster's entropy is computed similarly to Equation 5.2, but using clusters as in Equation 5.3.

Entropy values obtained with each of these methods were used to filter dialogue data in three ways. The SOURCE approach filters utterance pairs in which the source utterance has high entropy, TARGET filters those with a high entropy target, and finally the BOTH strategy filters all utterance pairs that are filtered by either SOURCE or TARGET. Some additional techniques did not yield meaningful improvement and were excluded from further evaluation. Clustering based on the Jaccard similarity of the bag of words of utterances only added noise to IDENTITY and resulted in much worse clusters than SENT2VEC. Clustering single occurrences of each unique utterance (as opposed to datasets with multiplicity) lead to less useful clusters than when clustering the whole dataset, probably because it resulted in less weight being given to the frequent utterances that we want to filter out. K-means proved inferior to the Mean Shift algorithm, which is a density-based clustering algorithm and seems to work better for clustering vectors of sentences. Filtering stop words before clustering did not improve the quality of clusters, probably because many utterances that we want to filter out contain a large number of stop words.

5.2 Data Analysis

5.2.1 Dataset

With 90 000 utterances in 13 000 dialogs, DailyDialog (Li et al., 2017c), our primary dataset, is comparable in size with the Cornell Movie-Dialogs Corpus (Danescu-Niculescu-Mizil and Lee, 2011), but contains real-world conversations. Using the IDENTITY approach, about 87% of utterances have 0 entropy (i.e. they do not appear with more than one target), 5% have an entropy of 1 (e.g. they appear twice, with different targets), remaining values rise sharply to 7. This distribution is similar for source and target utterances. Table 5.1 shows high entropy utterances.

Entropy is clearly proportional to utterance frequency (Figure 5.3), but has a wide range of values among utterances of equal frequency. For example, utterances with a frequency of 3 can have entropies ranging from 0 to $\log_2 3 \approx 1.58$, the latter of which would be over our filtering threshold of 1 (see Section 5.3.1 for details on selecting thresholds). Since high-entropy utterances are relatively short, we also examined the relationship between entropy and utterance length (Figure 5.4). Given the relationship between frequency and entropy, it comes as no surprise that longer utterances have lower entropy.

5.2.2 Clustering Results

Compared to IDENTITY, both SENT2VEC and AVG-EMBEDDING produce a much lower number of clusters with 0 entropy, but also a huge cluster with more than 5000 elements (the size of the second-largest cluster is below 500), which we didn't filter since it clearly doesn't group utterances with similar meaning. Generally,

Utterance	Frequency	Entropy
yes .	173	7.06
thank you .	141	6.57
why ?	104	6.33
here you are .	99	6.10
ok .	75	6.00
what do you mean ?	77	5.97
may i help you ?	72	5.96
can i help you ?	80	5.93
really ?	74	5.91
sure .	66	5.66
what can i do for you ?	51	5.63
why not ?	61	5.42
what ?	48	5.27
what happened ?	44	5.18
anything else ?	43	5.17
thank you very much .	72	5.14
what is it ?	41	5.06
i see .	42	5.05
no .	42	5.04
thanks .	50	5.03

Table 5.1: Top 20 source utterances (from DailyDialog) sorted by entropy. The entropy was calculated with IDENTITY.

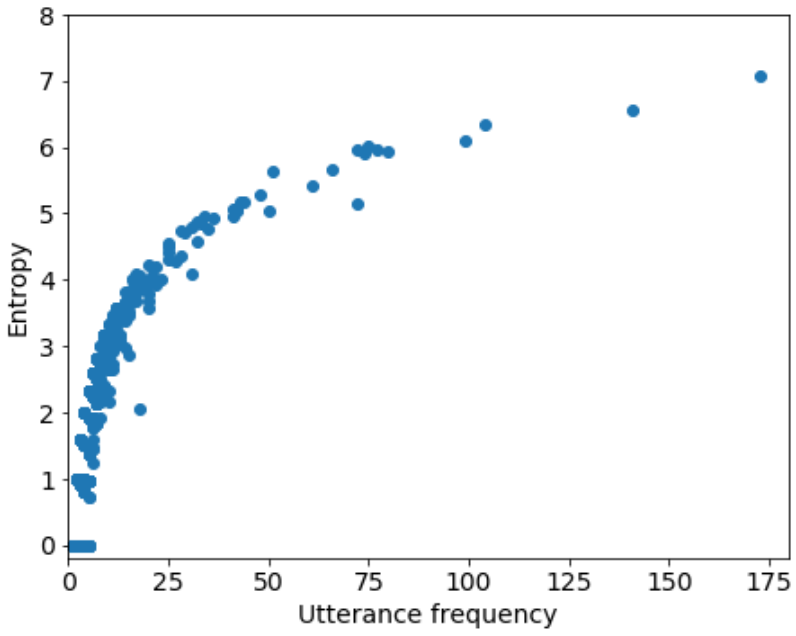

Figure 5.3: Entropy of source utterances (computed with IDENTITY) with respect to utterance frequency.

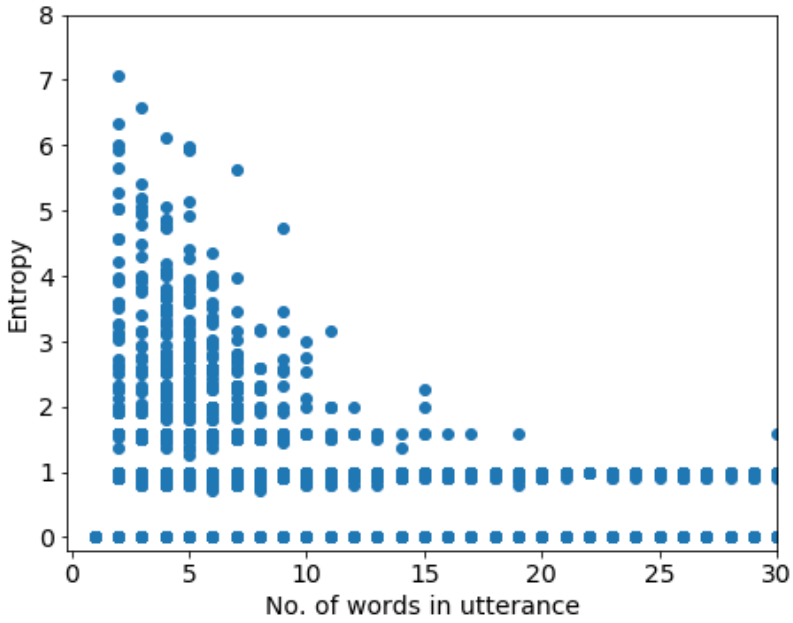

Figure 5.4: Entropy of source utterances (computed with IDENTITY) with respect to utterance length.

```
hi  an  .  how are you  ?
hi  craig  !  how are you  ?
hi  how are you  .  is alice there  ?
hi  !  how are you doing  ?
hi  francis  morning  !  how are you doing today  ?
hi  peter  !  how are you  ?
hi  randy  .  what are you doing right now  ?
hi  jane  .  how are you doing this morning  ?
hi  nancy  .  how are you doing  ?
hi  how are you doing  ?
hi  nancy . how are you doing  ?
hi steve  .  this is mike  .  what are you doing  ?
hi  how are you  ?
hi b  .  how are you  ?
hi  alex  .  how are you doing  ?
hi  !  how are you going  ?
hi mike  how are you doing  ?
hi  .  how can i help you  ?
hi  jack  !  how are you doing  ?
hi  carlos  .  what are you doing this afternoon  ?
hi  victor  .  how are you  ?
oh  yes  .  hi  how are you  ?
hi  tom  .  how have you been  ?
hi  bob  !  how are you doing  ?
hi  alice . how are you  ?
hi  brad  .  how are you today  ?
```

Figure 5.5: A cluster produced by SENT2VEC.

clusters were formed of similar utterances with the occasional exception of longer outlier utterances clustered together (instead of creating a separate cluster for each outlier), which can be attributed to the nature of the clustering algorithm. Overall, SENT2VEC appeared to produce better clusters than AVG-EMBEDDING, as reflected in the evaluation in Section 5.3.

```
Center: coffee  ?  i don  t honestly like that kind of stuff  .
Entropy: 5.885753989955374
Size: 138
Elements:
here you are  .
here you are  .  have a nice stay here  .
here they are  .
you are kidding  .
of course  .  here you are  .
here you are  madam . all these are sixteens  .
we are here  .
here we are  .  this is wangfujing street  .
here you are  .  you left the medicine here  .
certainly  here you are  .
of course . here you are  .
sure  here you are  .
here you are . you can try them on  .
here you are . it s very attractive  .
here we are  .
surely of course . here you are  .
of course  here you are  .
you are late  .
thank you  .  here you are  .
here you are  madam  .  all these are sixteens  .
```

Figure 5.6: A high-entropy cluster produced by SENT2VEC.

We experimented with different bandwidth values[3] for the Mean Shift algorithm to produce clusters with as many elements as possible while also keeping the elements semantically similar. In an example cluster (Figure 5.5) we can see that the clustering was able to group several variants of *How are you?*, in particular, those with different names. In general, we noticed that both in the case of IDENTITY and the clustering methods, utterances labelled with the highest entropy are indeed those generic sources and replies which we hoped to eliminate. See Figure 5.6 and Figure 5.7 for two high entropy clusters.

5.3 Experiments

5.3.1 Model and Parameters

We use the official implementation[4] of the Transformer (see Table 5.3 for a report of hyperparameters). The vocabulary for DailyDialog was limited to the most frequent 16 384 words, and train / validation / test splits contained 71 517 / 9 027 / 9 318 examples, respectively.

Clustering and Filtering. For AVG-EMBEDDING fastText[5] embeddings were used. The bandwidth of Mean Shift was set to 0.7 and 3.5 for AVG-EMBEDDING and SENT2VEC, which produced 40 135 and 23 616 clusters, respectively. Entropy

[3]Bandwidth is like a radius in the latent space of utterance representations (Fukunaga and Hostetler, 1975).

[4]https://github.com/tensorflow/tensor2tensor

[5]https://fasttext.cc/

```
Center: come on  you can at least try a little  besides your cigarette  .
Entropy: 4.959251313559618
Size: 140
Elements:
thank you very much for your kindness  .
yes  please  .  thank you very much  .
sure . thank you very much  .
thank you very much  .  it  s very kind of you  .
okay  .  thank you very much  .
thank you very much  .
you are so kind  !  thank you very much  .
yes . thank you very much  .
thank you very much  .  see you tomorrow afternoon  .
i love flowers you know  .  thank you very much  .
yes  thank you very much  .
i see  .  thank you very much  .
thank you very much  .  take the pills  .
well  thank you very much  .
thank you very much  .  are you here alone  ?
here it is  .  and thank you very much  .
i understand  .  thank you very much  !
oh  thank you very much  .
fine  thank you very much  .
ok  thank you very much  .
oh  thank you so much  .
i  ll bring the card  .  thank you very much  .
all right  .  thank you very much  .
ok  i see  .  thank you very much  .
quite well  thank you  .
i know  .  thank you very much  .
i love flowers  you know  .  thank you very much  .
very well  thank you  .
thank you very much  .  byebye  .
oh  well thank you very much  .
thank goodness  .  it is still there  .  thank you very much  .
thank you so much  .
thank you very much  !
thank you very much  doctor  .
okay  sir  here you are  .  thank you very much  .
yes  thank you so much  .
fantastic  .  thank you very much  .
thank you very much  mr green  .
well  thank you  .
```

Figure 5.7: A high-entropy cluster produced by SENT2VEC.

thresholds and amount of data filtered can be found in Table 5.2. Generally we set the threshold so that filtered data amount is similar to the DailyDialog IDENTITY scenario. We also set a threshold for the maximum average utterance length (15 and 20 for AVG-EMBEDDING and SENT2VEC) in clusters that we considered for filtering, excluding outliers from the filtering process (see Section 5.2.2).

Training and Decoding. Word embeddings of size 512 were randomly initialized, batch size was set to 2048 tokens, and we used the Adam optimizer (Kingma and Ba, 2014). We experimented with various beam sizes (Graves, 2012), but greedy decoding performed better according to all metrics, also observed previously (Asghar et al., 2017, Shao et al., 2017, Tandon et al., 2017).

Dataset	Type	Threshold	SOURCE	TARGET	BOTH
	IDENTITY	1	5.64%	6.98%	12.2%
DailyDialog	AVG-EMBEDDING	3.5	5.39%	7.06%	12.0%
	SENT2VEC	3.5	6.53%	8.45%	14.3%
Cornell	IDENTITY	4	-	7.39%	14.1%
Twitter	IDENTITY	0.5	-	1.82%	9.96%

Table 5.2: Entropy threshold and amount of data filtered for all datasets in the 3 filtering scenarios.

Name	Value
Hidden size	512
Number of hidden layers	6
Label smoothing	0.1
Filter size	2048
Number of attention heads	8
Layer dropout	0.2
Relu dropout	0.1
Attention dropout	0.1
Learning rate	0.2
Learning rate warmup steps	8000

Table 5.3: Transformer hyperparameters.

5.3.2 DailyDialog Results

We compute metrics on the unfiltered test set to show that filtered trainings perform better even on utterances that would have been filtered from the training data. TRF, the baseline `transformer` model trained on unfiltered data is compared to the 9 trainings on filtered data. In light of the issues mentioned in Chapter 3 we evaluate both at the validation loss minimum and at an overfitted point when the metrics start saturating (150 epochs). Metrics notations: Response length ($|U|$); per-word and per-utterance unigram (H^u_w, H^u_u) and bigram (H^b_w, H^b_u) entropy; unigram and bigram-level KL divergence (D^u_{kl}, D^b_{kl}); distinct-1 and distinct-2 (d1, d2); embedding metrics *average* (AVG), *extrema* (EXT), and *greedy* (GRE); coherence (COH); the 4 BLEU metrics (b1, b2, b3, b4).

Evaluating at the minimum validation loss (Table 5.4) clearly shows that models trained on data filtered by IDENTITY and SENT2VEC are better than the baseline. IDENTITY performs best among the three methods, surpassing the baseline on all but the *distinct-1* metric. SENT2VEC is a close second, getting higher values on fewer metrics than IDENTITY, but mostly improving on the baseline. Finally, AVG-EMBEDDING is inferior to the baseline, as it didn't produce clusters as meaningful as SENT2VEC, and thus produced a lower quality training set. It seems like filtering high entropy targets (both in the case of IDENTITY and SENT2VEC) is more beneficial than filtering sources, and BOTH falls mostly in the middle as expected, since it combines the two filtering types. By removing example responses that are boring

		$\lvert U\rvert$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	AVG	EXT	GRE	COH	d1	d2	b1	b2	b3	b4
trf		8.6	7.30	12.2	63.6	93	.330	.85	.540	.497	.552	.538	**.0290**	.149	.142	.135	.130	.119
ID	B	9.8	7.44	12.3	71.9	105	.315	.77	.559	**.506**	*.555*	.572	.0247	.138	.157	.151	.147	.136
	T	*10.9*	**7.67**	**12.7**	**83.2**	**121**	**.286**	**.72**	**.570**	**.507**	.554	**.584**	.0266	**.150**	**.161**	**.159**	**.156**	**.146**
	S	9.4	7.19	11.9	66.4	98	.462	1.08	.540	.495	.553	.538	.0262	.130	.139	.133	.128	.117
AE	B	7.9	7.25	12.0	57.7	83	.447	1.05	.524	.486	.548	.524	.0283	.132	.128	.121	.115	.105
	T	8.6	7.26	12.1	61.4	90	.425	1.12	.526	.492	.548	.529	.0236	.115	.133	.127	.121	.111
	S	*9.0*	7.21	11.9	*65.1*	*95*	.496	1.16	.536	.490	.548	.538	.0232	.109	.134	.130	.126	.116
SC	B	10.0	7.40	12.3	72.6	108	.383	.97	.544	.497	.549	.550	.0257	.131	.145	.142	.138	.128
	T	**11.2**	*7.49*	*12.4*	**82.2**	**122**	.391	.97	*.565*	*.500*	.552	*.572*	.0250	.132	*.153*	*.153*	*.152*	*.142*
	S	**11.1**	7.15	11.9	74.4	114	.534	1.27	.546	*.501*	**.560**	.544	.0213	.102	.144	.139	.135	.125

Table 5.4: Metrics computed at the minimum of the validation loss on the unfiltered test set (DailyDialog). TRF refers to `transformer`, **ID** to IDENTITY, **AE** to AVG-EMBEDDING, and **SC** to SENT2VEC. SOURCE-side, TARGET-side, and filtering BOTH sides are denoted by initials. Best results are highlighted with bold and best results separately for each entropy computing method are in italic (and those within a 95% confidence interval).

and generic from the dataset the model learns to improve response quality. Finding such utterances is useful for many purposes, but earlier it has been done mainly manually (Li et al., 2016d, Shen et al., 2017), whereas we provide an automatic, unsupervised method of detecting them based on entropy.

Every value is higher after 150 epochs of training than at the validation loss minimum (Table 5.5). The most striking change is in the unigram KL divergence, which is now an order of magnitude lower. IDENTITY still performs best, falling behind the baseline on only the two *distinct* metrics. Interestingly this time BOTH filtering was better than the TARGET filtering. SENT2VEC still mostly improves the baseline and AVG-EMBEDDING now also performs better or at least as good as the baseline on most metrics. In some cases the best performing model gets quite close to the ground truth performance. On metrics that evaluate utterances without context (i.e. entropy, divergence, *distinct*), randomly selected responses achieve similar values as the ground truth, which is expected. However, on embedding metrics, coherence, and BLEU, random responses are significantly worse than those of any model evaluated.

Computing the unigram and bigram KL divergence with a uniform distribution instead of the model yields a value of 4.35 and 1.87, respectively. Thus, all models learned a much better distribution, suggesting that this is indeed a useful metric. We believe the main reason that clustering methods perform worse than IDENTITY is that clustering adds some noise to the filtering process. Conducting a good clustering of sentence vectors is a hard task. This could be remedied by filtering only utterances instead of whole clusters, thus combining IDENTITY and the clustering methods. In this scenario, the entropy of individual utterances is computed based on the clustered

		$\|U\|$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	AVG	EXT	GRE	COH	d1	d2	b1	b2	b3	b4
trf		11.5	7.98	**13.4**	95	142	.0360	.182	.655	**.607**	**.640**	.567	**.0465**	**.297**	**.333**	.333	.328	.315
ID	B	**13.1**	**8.08**	**13.6**	**107**	**162**	.0473	.210	**.668**	**.608**	**.638**	**.598**	.0410	.275	**.334**	**.340**	**.339**	**.328**
	T	12.2	8.04	**13.6**	100	150	**.0335**	**.181**	**.665**	**.610**	**.640**	.589	.0438	.289	**.338**	**.341**	**.339**	**.328**
	S	12.3	7.99	**13.5**	101	153	.0406	.187	.662	**.610**	**.641**	.578	.0444	.286	**.339**	**.342**	**.338**	**.326**
AE	B	11.9	7.98	**13.5**	98	147	.0395	.197	.649	.600	.628	.574	.0434	.286	.318	.321	.318	.306
	T	*12.5*	7.99	**13.5**	*102*	*155*	.0436	.204	.656	.602	.634	*.580*	.0423	.279	.324	.327	.325	.313
	S	12.1	7.93	**13.4**	99	148	.0368	.186	.658	.605	**.636**	*.578*	.0425	.278	.325	.328	.324	.311
SC	B	12.8	**8.07**	**13.6**	105	159	.0461	.209	.655	.600	.629	*.583*	.0435	.282	.322	.328	.327	.316
	T	**13.0**	**8.06**	**13.6**	**107**	**162**	.0477	.215	.657	.602	.632	*.585*	.0425	.279	.324	.330	.329	.318
	S	12.1	7.96	**13.4**	100	150	*.0353*	.183	.657	**.606**	**.638**	.576	.0443	.286	.331	.333	.329	.317
RT		13.5	8.40	14.2	116	177	.0300	.151	.531	.452	.481	.530	.0577	.379	.090	.121	.130	.125
GT		14.1	8.39	13.9	122	165	0	0	1	1	1	.602	.0488	.362	1	1	1	1

Table 5.5: Metrics computed on the unfiltered test set (DailyDialog) after 150 epochs of training. TRF refers to `transformer`, **ID** to IDENTITY, **AE** to AVG-EMBEDDING, and **SC** to SENT2VEC. SOURCE-side, TARGET-side, and filtering BOTH sides are denoted by initials. Best results are highlighted with bold and best results separately for each entropy computing method are in italic (and those within a 95% confidence interval). **GT** refers to ground truth responses and **RT** refers to randomly selected responses from the training set.

data. The intuition behind this approach would be that the noise in the clusters based on which we compute entropy is less harmful than the noise in clusters which we consider for filtering. Finally, Table 5.8 and Table 5.9 shows responses from the baseline and the best performing model to randomly selected inputs from the test set (which we made sure are not present in the training set) to show that training on filtered data does not degrade response quality.

5.3.3 Cornell and Twitter Results

To further solidify our claims we tested the two best performing variants of IDENTITY (BOTH and TARGET) on the Cornell Movie-Dialogs Corpus and on a subset of 220k examples from the Twitter corpus[6]. Entropy thresholds were selected to be similar to the DailyDialog experiments (Table 5.2). Evaluation results at the validation loss minimum on the Cornell corpus and the Twitter dataset are presented in Table 5.6 and Table 5.7, respectively. On these noisier datasets our simple IDENTITY method still managed to improve over the baseline, but the impact is not as pronounced and in contrast to DailyDialog, BOTH and TARGET perform best on nearly the same number of metrics. On these noisier datasets the clustering methods might

[6] `https://github.com/Marsan-Ma/chat_corpus/`

work better, this is left for future work. Compared to DailyDialog some important distinctions also underline that these datasets are of lesser quality. The COHERENCE metric is worse on the ground truth responses than on model responses (Table 5.6), and some embedding metrics and BLEU scores are better on randomly selected responses than on model responses (Table 5.7).

		$\lvert U\rvert$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	AVG	EXT	GRE	COH	d1	d2	b1	b2	b3	b4
trf		8.1	6.55	10.4	54	75	2.29	3.40	**.667**	.451	.635	**.671**	.0005	.001	**.108**	.120	.120	.112
ID	B	7.4	**6.67**	**10.8**	50	69	**1.96**	**2.91**	.627	**.455**	.633	.637	**.0021**	**.008**	.106	.113	.111	.103
	T	**12.0**	6.44	10.4	**74**	**106**	2.53	3.79	.646	**.456**	**.637**	.651	.0010	.003	**.108**	**.123**	**.125**	**.118**
RT		13.4	8.26	14.2	113	170	.03	.12	.623	.386	.601	.622	.0460	.320	.079	.102	.109	.105
GT		13.1	8.18	13.8	110	149	0	0	1	1	1	.655	.0400	.310	1	1	1	1

Table 5.6: Metrics on the unfiltered test set (Cornell) at the validation loss minimum. TRF refers to `transformer`, **ID** to IDENTITY. TARGET-side, and filtering BOTH sides are denoted by initials. Best results are highlighted with bold. **GT** refers to ground truth responses and **RT** refers to randomly selected responses from the training set.

		$\lvert U\rvert$	H_w^u	H_w^b	H_u^u	H_u^b	D_{kl}^u	D_{kl}^b	AVG	EXT	GRE	COH	d1	d2	b1	b2	b3	b4
trf		20.6	6.89	**11.4**	121	177	**2.28**	**3.40**	.643	.395	.591	.659	**.0021**	**.006**	.0519	.0666	.0715	.0693
ID	B	20.3	**6.95**	**11.4**	119	171	2.36	**3.41**	**.657**	.394	.595	**.673**	.0012	.003	**.0563**	**.0736**	.0795	.0774
	T	**29.0**	6.48	10.7	**157**	**226**	2.68	3.69	.644	**.403**	**.602**	.660	.0014	.005	**.0550**	**.0740**	**.0819**	**.0810**
RT		14.0	9.81	15.9	136	171	.05	.19	.681	.334	.543	.695	.0850	.540	.0444	.0751	.0852	.0840
GT		14.0	9.78	15.8	135	167	0	0	1	1	1	.734	.0810	.530	1	1	1	1

Table 5.7: Metrics on the unfiltered test set (Twitter) at the validation loss minimum. TRF refers to `transformer`, **ID** to IDENTITY. TARGET-side, and filtering BOTH sides are denoted by initials. Best results are highlighted with bold. **GT** refers to ground truth responses and **RT** refers to randomly selected responses from the training set.

Input	**Unfiltered**	**Unfiltered-Overfitted**	**identity target filtered**	**identity target filtered-overfitted**
some high speed chase scene i think .	i 'm not sure .	but many things are went wrong !	i 'm so sorry .	enough !
how long is this bus ride ?	about twelve hours .	it 's about forty-five minutes to an hour .	it 's about 20 minutes .	it 's only about fortyfive minutes to an hour .
is that yellow bus it ?	no it 's not far . it 's a bit cold .	yeah . there s a new one in america .	no it 's a yellow one .	no it 's not very expensive these days .
hi mary ! what do you like to do in your spare time ?	i like playing basketball .	well i like to play the piano .	i like playing basketball .	i like playing chess .
thirty u . s . dollars . . . and it 's only been used for six months .	that 's fine .	i see . is it a long time to come to the station ?	how many months did you get there ?	what 're two weeks ?
five hundred dollars altogether .	here you are .	here you are .	here is your bill .	here you are . thank you byebye .
look at this bridge . we will soon be crossing the thames .	yes i agree .	oh yeah .	how far is it ?	well there 's a spanish teacher . so i 'll often go back to the spanish museum .
where do they go ?	they 're on the bottom floor .	i have no idea .	they re in the cabinet next to the library .	go to them department please .
let me pay for it .	ok . here you are .	cash credit card or debit card ?	ok . here is the money .	cash credit card or debit card ?
he never does and in fact he seldom communicates with him .	he is a hard job . he can also learn a lot from the government .	that 's right . he has no children and his wife experience too .	he is a very kind of man .	yeah . he 's a very easy teacher .
so you must have had a busy and enjoyable day yesterday .	yes i am . i really enjoyed it .	yes . i am very sorry i can t make it .	i will . thanks for coming .	yes . let 's go .
yes . the reservation under mr . watson .	ok .	could you spell that please ?	i 'm afraid there 's no seat in the middle of the table .	ok . would you please fill out the table ?
there are so many to choose from . i did n't know you could do all of this using a card .	i see .	what 's the best part ?	i 'm not sure . i 'll go to the doctor and get my card right away .	you mean you do n't have to tell me the truth ?

Table 5.8: Responses to randomly selected test inputs which we made sure were not in the training data (DailyDialog). *Unfiltered* is the model trained on unfiltered data, and IDENTITY TARGET is the model trained on IDENTITY, TARGET filtered data. *Overfitted* means that the respective model is evaluated at an overfitted point.

Input	**Unfiltered**	**Unfiltered-Overfitted**	**identity target filtered**	**identity target filtered-overfitted**
you 're flattering me .	no .	no really .	no but you can t .	no really . i can only hear a little accent .
three bags and a suitcase . this is my luggage to check .	ok . i will take one .	what 's this ? essential oil ? wow ! this is a small bag .	i 'll put you in the bag .	ok . is this your luggage properly labelled sir ?
i like orange better .	what 's your favorite flavor ?	how about drinks ?	i 'll tell you what you 're doing .	could i have some water ?
i like to watch barbie sticking her tongue out .	what 's her figure ?	oh she 's a pretty woman .	you are not supposed to be serious .	oh shut up ! she 's not what you are thinking about .
ok . any time on friday will be ok with me .	ok . see you then .	i think so .	i 'll be expecting you then .	great !
no wonder you can control your voice so well . you are a professional singer .	i am not a singer .	i would n't mind a position that would be great .	i am a singer . i think i can be a singer in the lead .	i would n't like to be a singer but i love to sing that dream .
can you tell that i m excited ?	i m afraid i m not a child .	i m a fashion designer . next month i have a fashion show .	i m not sure . what do you want to know ?	i m sure that i ll cook the same for you .
would you like to have a shampoo sir ?	yes i 'd like a glass of iced tea .	no thank you .	no i 'd rather have it with me .	yes with the lux dandruff shampoo . it helps a lot to get rid of my dandruff .
what else would i prepare sir ?	you can bring your beef with your chicken and a bottle of wine .	oh good ! can you make some recommendation for me ?	well the interviewers are not interested in the position .	let me see . everything looks fine .
what made you think that ?	i like it a lot .	i love that shirt on you .	i think it 's great . i think i ve learned a lot from different countries .	i will care for it .
i can tell you what bus to catch but you have to walk a little bit .	i do n't know .	tell me how to get a ticket and a student bus station .	i 'm sorry but i do n't have to wait .	you 're going to have a car .

Table 5.9: Responses to randomly selected test inputs which we made sure were not in the training data (DailyDialog). *Unfiltered* is the model trained on unfiltered data, and IDENTITY TARGET is the model trained on IDENTITY, TARGET filtered data. *Overfitted* means that the respective model is evaluated at an overfitted point.

Chapter 6

Future Work

In the future, we plan to explore several additional ideas related to the work presented in Chapter 4 and Chapter 5. We wish to extend the Gutenberg Dialogue Dataset to multiple languages and improve the quality of the English dataset. A classifier could be trained to decide whether two consecutive utterances are part of the same dialogue (looking at non-conversational context as well). Positive and negative examples could be generated with a very low/high dialogue gap, or by manual annotation. Speaker related errors could be diminished using speaker identification. More elaborate techniques could be used to extract persona information from book context surrounding dialogues. These and other priors could be used to build more knowledgable dialogue models. Extending to other languages involves delimitation analysis, implementation of heuristics, and error analysis, requiring a minimal amount of language knowledge. We welcome contributions from the community, as our open-source modular pipeline minimizes the required effort for new languages. We also wish to compare out dataset with other datasets in a multi-turn setting using GPT-2 (Radford et al., 2019).

As mentioned in Section 5.3.2, we want to extend our clustering experiments combining the ideas behind IDENTITY and the clustering methods to make them more robust to noise. We wish to conduct clustering experiments on noisier datasets and try other sentence representations (Devlin et al., 2019). We also plan to combine our method with coherence-based filtering (Xu et al., 2018b). Furthermore, we intend to perform a direct quantitative evaluation of our method based on human evaluation. Finally, we believe our method is general enough that it could also be applied to datasets in other similar NLP tasks, such as machine translation, which could open another interesting line of future research.

An important part of conversations is timing and the involvement of both speakers. Current chatbot models are trained in such ways that they will only emit one response instantly after they receive the user's utterance. To make chatbots more human-like we propose an additional term to the loss function, which is based on the time delay between an utterance and a reply. Essentially the chatbot has to generate a reply and also guess how much time should it wait before emitting the reply. Twitter-style datasets usually have such annotations so the implementation of this feature should be straight-forward.

The time-delay can be represented as a vector of probabilities over time frame categories. For example, the model could have 10% confidence that the reply should be delayed by 0 to 10 seconds and 90% confidence that it should be delayed by 10 to 20 seconds. Such a representation is effective because a simple softmax can be used to get the probabilities of time frame categories and backpropagation can be extended by comparing the predicted time-delay vector with the one-hot ground-truth timing vector. Not only would this addition help achieve a more human-like delay (which could also be tunable) in the chatbot's responses, but it would allow a conversational model to periodically generate new utterances by itself, based on the conversation history. Regardless of whether the user has inputted an utterance or not, the conversation history can be fed into the model after the chatbot emitted an utterance, and a new utterance can be generated if the model thinks that it is necessary to further the conversation. With the time delay mechanism, this process can go on indefinitely, since the generation of the new utterance will only be considered after the time delay for the current utterance has passed. This feature would make the chatbot even more human-like and engaging, by furthering the conversation without user interaction.

Another concept tied to temporal conditioning is real-time model updates. Basically, at each turn, the user's utterances can be backpropagated through the network, based on the conversation history. This would make the model update its weights and prior representations as the conversation goes on, thus remembering the dialogue. The technique could either replace or complement the need for taking into account long conversation histories with hierarchical models. It would be especially useful for real-world deployment of chatbots, where users expect a chatbot to be able to remember what they said 1000 utterances ago, which can not be achieved solely through a hierarchical model. Through these real-time updates of the parameters, the model is capable to encode and learn new information about the user, which acts as memory.

Chapter 7

Conclusion

We presented a thorough literature survey from the last 5 years. The three main aspects of dialogue modelling were discussed: datasets, models, and evaluation methods. We presented positives and negatives of approaches and techniques from the literature to these main aspects. We argued that current automatic evaluation metrics are not fit to accurately assess model performance. We showed experimentally how these metrics behave with respect to overfitting.

We presented the Gutenberg Dialogue Dataset consisting of 14.8M utterances in English. We described heuristics used in our dialogue extraction pipeline and conducted a detailed error analysis to uncover the causes of errors and asses data quality. In a pre-training comparison between Gutenberg and Opensubtitles we found that Gutenberg performs better on downstream datasets in both zero-shot and finetuning scenarios.

We proposed a simple unsupervised entropy-based approach that can be applied to any conversational dataset for filtering generic sources/targets that cause "confusion" during the training of open-domain dialogue models. We compared various setups in an extensive quantitative evaluation, and showed that the best approach is measuring the entropy of individual utterances and filtering pairs based on the entropy of target, but not source utterances.

Acknowledgements

I wish to thank Gábor Recski for his continued support since 2017. He was invaluable in my NLP initiation and dialogue modeling endeavours. I also wish to thank members of the HLT group for the many suggestions and advice related to some of the research presented in this thesis.

Bibliography

Akasaki, S. and Kaji, N. (2017). Chat detection in an intelligent assistant: Combining task-oriented and non-task-oriented spoken dialogue systems. *arXiv preprint arXiv:1705.00746*.

Arora, S., Liang, Y., and Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Asghar, N., Poupart, P., Jiang, X., and Li, H. (2017). Deep active learning for dialogue generation. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 78–83. Association for Computational Linguistics.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. .

Baheti, A., Ritter, A., Li, J., and Dolan, B. (2018). Generating more interesting responses in neural conversation models with distributional constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3970–3980. Association for Computational Linguistics.

Barone, A. V. M. and Sennrich, R. (2017). A parallel corpus of python functions and documentation strings for automated code documentation and code generation. *arXiv preprint arXiv:1707.02275*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bordes, A., Boureau, Y.-L., and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Britz, D. (2015). Recurrent neural network tutorial. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/. Accessed: 2017-10-09.

Carpenter, R. (2017). Cleverbot. http://www.cleverbot.com/. Accessed: 2017-10-04.

Chen, B. and Cherry, C. (2014). A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA. Association for Computational Linguistics.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Chiu, C.-C., Lawson, D., Luo, Y., Tucker, G., Swersky, K., Sutskever, I., and Jaitly, N. (2017). An online sequence-to-sequence model for noisy speech recognition. *arXiv preprint arXiv:1706.06428*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Choudhary, S., Srivastava, P., Ungar, L., and Sedoc, J. (2017). Domain aware neural dialog system. *arXiv preprint arXiv:1708.00897*.

Csáky, R., Purgai, P., and Recski, G. (2019). Improving neural conversational models with entropy-based data filtering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5650–5669, Florence, Italy. Association for Computational Linguistics.

Csaky, R. and Recski, G. (2017). Deep learning based chatbot models. In *National Scientific Students' Associations Conference*. National Scientific Students' Associations Conference.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Danescu-Niculescu-Mizil, C. and Lee, L. (2011). Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87. Association for Computational Linguistics.

Das, A., Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017). Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*.

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

Dinan, E., Logacheva, V., Malykh, V., Miller, A., Shuster, K., Urbanek, J., Kiela, D., Szlam, A., Serban, I., Lowe, R., Prabhumoye, S., Black, A. W., Rudnicky, A., Williams, J., Pineau, J., Burtsev, M., and Weston, J. (2019a). The second conversational intelligence challenge (convai2).

Dinan, E., Roller, S., Shuster, K., Fan, A., Auli, M., and Weston, J. (2019b). Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J. (2015). Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*.

Eric, M. and Manning, C. D. (2017). A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024*.

Facebook (2017). The babi project. `https://research.fb.com/downloads/babi/`. Accessed: 2017-10-13.

Fainberg, J., Krause, B., Dobre, M., Damonte, M., Kahembwe, E., Duma, D., Webber, B., and Fancellu, F. (2018). Talking to myself: self-dialogues as data for conversational agents. *arXiv preprint arXiv:1809.06641*.

Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Fang, H., Cheng, H., Sap, M., Clark, E., Holtzman, A., Choi, Y., Smith, N. A., and Ostendorf, M. (2018). Sounding board: A user-centric and content-driven social chatbot. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 96–100. Association for Computational Linguistics.

Feng, Y., Zhang, S., Zhang, A., Wang, D., and Abel, A. (2017). Memory-augmented neural machine translation. *arXiv preprint arXiv:1708.02005*.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40.

Gao, X., Lee, S., Zhang, Y., Brockett, C., Galley, M., Gao, J., and Dolan, B. (2019). Jointly optimizing diversity and relevance in neural response generation. *arXiv preprint arXiv:1902.11205*.

Ghazvininejad, M., Brockett, C., Chang, M.-W., Dolan, B., Gao, J., Yih, W.-t., and Galley, M. (2018). A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Goyal, K., Neubig, G., Dyer, C., and Berg-Kirkpatrick, T. (2017). A continuous relaxation of beam search for end-to-end training of neural sequence models. *arXiv preprint arXiv:1708.00111*.

Graves, A. (2012). Sequence transduction with recurrent neural networks. In *Representation Learning Workshop, ICML 2012*, Edinburgh, Scotland.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Gu, X., Cho, K., Ha, J.-W., and Kim, S. (2019). DialogWAE: Multimodal response generation with conditional wasserstein auto-encoder. In *International Conference on Learning Representations*.

Hancock, B., Bordes, A., Mazare, P.-E., and Weston, J. (2019). Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(23):146–162.

Havrylov, S. and Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *arXiv preprint arXiv:1705.11192*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Henderson, M. (2015). Machine learning for dialog state tracking: A review. In *Machine Learning in Spoken Language Processing Workshop*.

Henderson, M., Budzianowski, P., Casanueva, I., Coope, S., Gerz, D., Kumar, G., Mrkšić, N., Spithourakis, G., Su, P.-H., Vulić, I., and Wen, T.-H. (2019). A repository of conversational datasets. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 1–10, Florence, Italy. Association for Computational Linguistics.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Joshi, C. K., Mi, F., and Faltings, B. (2017). Personalization in goal-oriented dialog. *arXiv preprint arXiv:1706.07503*.

Kandasamy, K., Bachrach, Y., Tomioka, R., Tarlow, D., and Carter, D. (2017). Batch policy gradient methods for improving neural conversation models. In *International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konstas, I., Iyer, S., Yatskar, M., Choi, Y., and Zettlemoyer, L. (2017). Neural amr: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381*.

Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017). Natural language does not emerge'naturally'in multi-agent dialog. *arXiv preprint arXiv:1706.08502*.

Krause, B., Damonte, M., Dobre, M., Duma, D., Fainberg, J., Fancellu, F., Kahembwe, E., Cheng, J., and Webber, B. (2017). Edina: Building an open domain socialbot with self-dialogues. In *1st Proceedings of Alexa Prize (Alexa Prize 2017)*.

Krizhevsky, A. and Sutskever, I.and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS'2012*.

Kulikov, I., Miller, A. H., Cho, K., and Weston, J. (2018). Importance of a search strategy in neural dialogue modelling. *arXiv preprint arXiv:1811.00907*.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Lewis, M., Yarats, D., Dauphin, Y. N., Parikh, D., and Batra, D. (2017). Deal or no deal? end-to-end learning for negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453. Association for Computational Linguistics.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016a). A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT 2016*, pages 110–119. Association for Computational Linguistics.

Li, J., Galley, M., Brockett, C., Spithourakis, G. P., Gao, J., and Dolan, B. (2016b). A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 994–1003. Association for Computational Linguistics.

Li, J. and Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732. Association for Computational Linguistics.

Li, J., Miller, A. H., Chopra, S., Ranzato, M., and Weston, J. (2016c). Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823*.

Li, J., Monroe, W., and Jurafsky, D. (2017a). Data distillation for controlling specificity in dialogue generation. *arXiv preprint arXiv:1702.06703*.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016d). Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Li, J., Monroe, W., Shi, T., Ritter, A., and Jurafsky, D. (2017b). Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169. Association for Computational Linguistics.

Li, X., Mou, L., Yan, R., and Zhang, M. (2016e). Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. *arXiv preprint arXiv:1604.04358*.

Li, Y., Su, H., Shen, X., Li, W., Cao, Z., and Niu, S. (2017c). Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, pages 986–995. AFNLP.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Lipton, Z., Li, X., Gao, J., Li, L., Ahmed, F., and Deng, L. (2018). Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. Association for the Advancement of Artificial Intelligence.

Lison, P. and Bibauw, S. (2017). Not all dialogues are created equal: Instance weighting for neural conversational models. *arXiv preprint arXiv:1704.08966*.

Lison, P. and Tiedemann, J. (2016). Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*.

Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132. Association for Computational Linguistics.

Liu, H., Lin, T., Sun, H., Lin, W., Chang, C.-W., Zhong, T., and Rudnicky, A. (2017). Rubystar: A non-task-oriented mixture model dialog system. In *1st Proceedings of Alexa Prize (Alexa Prize 2017)*.

Lowe, R., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y., and Pineau, J. (2017). Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126. Association for Computational Linguistics.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Ludwig, O. (2017). End-to-end adversarial learning for generative conversational agents. *arXiv preprint arXiv:1711.10122*.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Marietto, M. d. G. B., de Aguiar, R. V., Barbosa, G. d. O., Botelho, W. T., Pimentel, E., França, R. d. S., and da Silva, V. L. (2013). Artificial intelligence markup language: A brief tutorial. *arXiv preprint arXiv:1307.3091*.

Microsoft (2017). Microsoft bot framework. `https://dev.botframework.com/`. Accessed: 2017-10-04.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Mo, K., Zhang, Y., Yang, Q., and Fung, P. (2017). Fine grained knowledge transfer for personalized task-oriented dialogue systems. *arXiv preprint arXiv:1711.04079*.

Moghe, N., Arora, S., Banerjee, S., and Khapra, M. M. (2018). Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332, Brussels, Belgium. Association for Computational Linguistics.

Mou, L., Song, Y., Yan, R., Li, G., Zhang, L., and Jin, Z. (2016). Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358. The COLING 2016 Organizing Committee.

Murray, K. and Chiang, D. (2018). Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Olabiyi, O., Salimov, A., Khazane, A., and Mueller, E. (2018). Multi-turn dialogue response generation in an adversarial learning framework. *arXiv preprint arXiv:1805.11752*.

Olah, C. (2015). Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed: 2017-10-08.

opensubtitles.org (2017). Opensubtitles. `https://www.opensubtitles.org/`. Accessed: 2017-10-08.

Pagliardini, M., Gupta, P., and Jaggi, M. (2018). Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540. Association for Computational Linguistics.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Park, Y., Cho, J., and Kim, G. (2018). A hierarchical latent structure for variational conversation modeling. In *Proceedings of NAACL-HLT 2018*, pages 1792–1801. Association for Computational Linguistics.

Park, Y., Patwardhan, S., Visweswariah, K., and Gates, S. C. (2008). An empirical analysis of word error rate and keyword error rate. In *INTERSPEECH*, pages 2070–2073.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. `https://github.com/openai/gpt-2`.

Ram, A., Prasad, R., Khatri, C., Venkatesh, A., Gabriel, R., Liu, Q., Nunn, J., Hedayatnia, B., Cheng, M., Nagar, A., et al. (2018). Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Ramachandran, P., Liu, P. J., and Le, Q. V. (2016). Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sedoc, J., Ippolito, D., Kirubarajan, A., Thirani, J., Ungar, L., and Callison-Burch, C. (2018). Chateval: A tool for the systematic evaluation of chatbots. In *Proceedings of the Workshop on Intelligent Interactive Systems and Language Generation (2IS&NLG)*, pages 42–44. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Serban, I. V., Klinger, T., Tesauro, G., Talamadupula, K., Zhou, B., Bengio, Y., and Courville, A. C. (2017a). Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., Kim, T., Pieper, M., Chandar, S., Ke, N. R., et al. (2017b). A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A. C., and Bengio, Y. (2017c). A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence.

Shalyminov, I., Dušek, O., and Lemon, O. (2018). Neural response ranking for social conversation: A data-efficient approach. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 1–8. Association for Computational Linguistics.

Shang, L., Lu, Z., and Li, H. (2015). Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Shannon, M. (2017). Optimizing expected word error rate via sampling for speech recognition. *arXiv preprint arXiv:1706.02776*.

Shao, Y., Gouws, S., Britz, D., Goldie, A., Strope, B., and Kurzweil, R. (2017). Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2210–2219. Association for Computational Linguistics.

Shen, X., Su, H., Li, W., and Klakow, D. (2018a). Nexus network: Connecting the preceding and the following in dialogue generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4316–4327. Association for Computational Linguistics.

Shen, X., Su, H., Li, Y., Li, W., Niu, S., Zhao, Y., Aizawa, A., and Long, G. (2017). A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 504–509. Association for Computational Linguistics.

Shen, X., Su, H., Niu, S., and Demberg, V. (2018b). Improving variational encoder-decoders in dialogue generation. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. Association for the Advancement of Artificial Intelligence.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.

Song, Y., Yan, R., Li, X., Zhao, D., and Zhang, M. (2016). Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Sriram, A., Jun, H., Satheesh, S., and Coates, A. (2017). Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112, Montreal, CA.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Tandon, S., Bauer, R., et al. (2017). A dual encoder sequence to sequence model for open-domain dialogue modeling. *arXiv preprint arXiv:1710.10520*.

Tao, C., Mou, L., Zhao, D., and Yan, R. (2018). Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. Association for the Advancement of Artificial Intelligence.

Tensorflow (2017). Sequence-to-sequence models. `https://www.tensorflow.org/tutorials/seq2seq`. Accessed: 2017-10-08.

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In Calzolari, N., editor, *LREC*, Istanbul, Turkey. European Language Resources Association (ELRA).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning*.

Wallace, R. S. (2009). The anatomy of alice. *Parsing the Turing Test*, pages 181–210.

Wang, Y., Liu, C., Huang, M., and Nie, L. (2018). Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 2193–2203. Association for Computational Linguistics.

Wei, B., Lu, S., Mou, L., Zhou, H., Poupart, P., Li, G., and Jin, Z. (2017). Why do neural dialog systems generate short and meaningless replies? a comparison between dialog and translation. *arXiv preprint arXiv:1712.02250*.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Weston, J., Dinan, E., and Miller, A. (2018). Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 87–92, Brussels, Belgium. Association for Computational Linguistics.

Williams, J. D., Asadi, K., and Zweig, G. (2017). Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wiseman, S. and Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

Wojciechowski, M. and Zakrzewicz, M. (2002). Dataset filtering techniques in constraint-based frequent pattern mining. In *Pattern detection and discovery*, pages 77–91. Springer.

Wolf, T. (2019). How to build a state-of-the-art conversational ai with transfer learning. `https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313`. Accessed: 2020-04-29.

Wolf, T., Sanh, V., Chaumond, J., and Delangue, C. (2018). Transfertransfo: A transfer learning approach for neural network based conversational agents. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*.

Worswick, S. (2017). Mitsuku. `http://www.mitsuku.com/`. Accessed: 2017-10-04.

Wu, B., Jiang, N., Gao, Z., Li, S., Rong, W., and Wang, B. (2018). Why do neural response generation models prefer universal replies? *arXiv preprint arXiv:1808.09187*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xing, C., Wu, W., Wu, Y., Liu, J., Huang, Y., Zhou, M., and Ma, W.-Y. (2017). Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. Association for the Advancement of Artificial Intelligence.

Xing, C., Wu, Y., Wu, W., Huang, Y., and Zhou, M. (2018). Hierarchical recurrent attention network for response generation. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. Association for the Advancement of Artificial Intelligence.

Xing, Y. and Fernández, R. (2018). Automatic evaluation of neural personality-based chatbots. In *Proceedings of The 11th International Natural Language Generation Conference*, pages 189–194. Association for Computational Linguistics.

Xu, C., Wu, W., and Wu, Y. (2018a). Towards explainable and controllable open domain dialogue generation with dialogue acts. *arXiv preprint arXiv:1807.07255*.

Xu, X., Dušek, O., Konstas, I., and Rieser, V. (2018b). Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3981–3991. Association for Computational Linguistics.

Yang, Y., Huang, L., and Ma, M. (2018). Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium. Association for Computational Linguistics.

Yao, K., Peng, B., Zweig, G., and Wong, K.-F. (2016). An attentional neural conversation model with improved specificity. *arXiv preprint arXiv:1606.01292*.

Yao, K., Zweig, G., and Peng, B. (2015). Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*.

Yin, Z., Chang, K.-h., and Zhang, R. (2017). Deepprobe: Information directed sequence understanding and chatbot design via recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2131–2139. ACM.

Yu, Z., Black, A. W., and Rudnicky, A. I. (2017). Learning conversational systems that interleave task and non-task content. *arXiv preprint arXiv:1703.00099*.

Zhang, H., Lan, Y., Guo, J., Xu, J., and Cheng, X. (2018a). Reinforcing coherence for sequence to sequence model in dialogue generation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 4567–4573.

Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., and Weston, J. (2018b). Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 2204–2213. Association for Computational Linguistics.

Zhang, Y., Galley, M., Gao, J., Gan, Z., Li, X., Brockett, C., and Dolan, B. (2018c). Generating informative and diverse conversational responses via adversarial information maximization. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*.

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2019). Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

Zhao, T., Lee, K., and Eskenazi, M. (2018). Unsupervised discrete sentence representation learning for interpretable neural dialog generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 1098–1107. Association for Computational Linguistics.

Zhao, T., Lu, A., Lee, K., and Eskenazi, M. (2017a). Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. *arXiv preprint arXiv:1706.08476*.

Zhao, T., Zhao, R., and Eskenazi, M. (2017b). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

Zhou, H., Huang, M., Zhang, T., Zhu, X., and Liu, B. (2018a). Emotional chatting machine: Emotional conversation generation with internal and external memory. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. Association for the Advancement of Artificial Intelligence.

Zhou, K., Prabhumoye, S., and Black, A. W. (2018b). A dataset for document grounded conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 708–713, Brussels, Belgium. Association for Computational Linguistics.

Zhu, W., Mo, K., Zhang, Y., Zhu, Z., Peng, X., and Yang, Q. (2017). Flexible end-to-end dialogue system for knowledge grounded conversation. *arXiv preprint arXiv:1709.04264*.