National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

Empathetic Dialogue Generation using generation-based models

DIPLOMA THESIS

EMMANOUIL ZARANIS

Supervisor : Alexandros Potamianos
Associate Professor NTUA

Athens, November 2020

National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

Empathetic Dialogue Generation using generation-based models

DIPLOMA THESIS

EMMANOUIL ZARANIS

Supervisor : Alexandros Potamianos
Associate Professor NTUA

Approved by the examining committee on the November 24, 2020.

Alexandros Potamianos Associate Professor NTUA	Konstantinos Tzafestas Associate Professor NTUA	Athanasios Katsamanis Principal Researcher ATHENA RC

Athens, November 2020

Emmanouil Zaranis

Electrical and Computer Engineer

Dedicated to family and friends.

Περίληψη

Ανάμεσα στις ποικίλες προσεγγίσεις για την δημιουργία αυτόματων διαλογικών συστημάτων, τα διαλογικά συστήματα ανοιχτού πεδίου που βασίζονται σε μεθόδους παραγωγής γλώσσας (generation-based open-domain chatbots) φαίνεται να έχουν ιδιαίτερο ερευνητικό ενδιαφέρον. Σε αυτή την διπλωματική εργασία, στοχεύουμε να συνεισφέρουμε στα πλαίσια του να δημιουργηθεί ένας αυτόματος συνομιλητής ικανός να ανταποκριθεί κατάλληλα στον χρήστη, κατανοώντας όχι μόνο τι συζητιέται αλλά και ποικίλα συναισθήματα που κρύβονται στην συνομιλία, ανταποκρινόμενος τελικά με παρόμοιο συναισθηματικό τρόπο.

Αρχικά εισάγουμε τον αναγνώστη στον κλάδο των διαλογικών συστημάτων, παρέχοντας το κατάλληλο θεωρητικό υπόβαθρο στον τομέα της μηχανική μάθησης (Machine Learning), των βαθιών νευρωνικών δικτύων (deep neural networks) και της Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing). Στην συνέχεια, μελετάμε σε βάθος μοντέλα που βασίζονται σε μεθόδους παραγωγής γλώσσας (generation-based models) στον τομέα των διαλογικών συστημάτων. Πιο συγκεκριμένα μελετάμε τις παραδοσιακές αρχιτεκτονικές μοντέλων ακολουθίας σε ακολουθία (seq2seq) και τις επεκτάσεις τους με χρήση του μηχανισμού προσοχής (attention mechanism), καθώς και πιο σύγχρονες αρχιτεκτονικές βασιζόμενες στους transformers όπως τα μοντέλα Transformer Encoder Decoder, BERT, GPT-2, και T5. Αφού μελετήσουμε εξονυχιστκά τις αρχιτεκτονικές αυτές, παρουσιάζουμε τις πιο ευρέως χρησιμοποιούμενες μεθόδους αποκωδικοποιήσης (decoding methods) καθώς και μεθόδους αξιολόγησης των διαλογικών συστημάτων.

Ύστερα λοιπόν από την μελέτη της παραγωγής γλώσσας με χρήση generation-based μοντέλων, εμβαθύνουμε στην δημιουργία αυτόματων διαλογικών συστημάτων με χαρακτηριστικά ενσυναίσθησης (empathetic chatbots), χρησιμοποιώντας generation-based μοντέλα. Πιο συγκεκριμένα, εστιάζοντας στην εργασία Empathetic Dialogues που προτείνεται από την ερευνητική ομάδα του Facebook, μελετάμε τις ήδη υπάρχουσες προσεγγίσεις και πειραματιζόμαστε με διαφορετικές προσεγγίσεις για την βελτίωση των αποτελεσμάτων. Πιο συγκεκριμένα πειραματιζόμαστε με τις αρχιτεκτονικές BERT2BERT και BERT2GPT2, παράγοντας συγκρίσιμα αποτελέσματα με εκείνα που έχουν δημοσιευθεί από άλλους ερευνητές. Επιπλέον, πειραματιζόμαστε με τρεις προσεγγίσεις βασιζόμενες στο μοντέλο T5. Στην πρώτη προσέγγιση χρησιμοποιούμε το μοντέλο T5 ως έχει, βελτιστοποιώντας την εκπαίδευση του (finetuning) στο σύνολο δεδομένων Empathetic Dialogues. Στην δεύτερη και την τρίτη προσέγγιση, προεκτείνουμε το T5 μοντέλο με χρήση της εκμάθησης πολλαπλών εργασίων (multitask-learning). Τελικά, οι προτεινόμενες προσεγγίσεις μας, που βασίζονται στο μοντέλο T5, βελτιώνουν τα state-of-the-art αποτελέσματα με βάση την μετρική αξιολόγησης BLEU, επιτυγχάνοντας παράλληλα κοντινά αποτελέσματα με αυτά των state-of-the-art μοντέλων με βάση την μετρική perplexity. Τέλος, παρουσιάζουμε κάποια χαρακτηριστικά παραδείγματα διαλόγων, για μία πιο αντικειμενική αξιολόγηση, και αφού σχολιάσουμε τα αποτελέσματα, προτείνουμε μια σειρά από μελοντικές προεκτάσεις που θα μπορούσαν να συνεισφέρουν σημαντικά στον υπό μελέτη ερευνητικό τομέα.

Λέξεις κλειδιά

Ενσυναίσθηση, διαλογικά συστήματα, chatbots, βαθιά νευρωνικά δίκτυα, μηχανική μάθηση, επεξεργασία φυσικής γλώσσας, transformers, HRED, BERT, GPT2, T5, BERT2BERT, BERT2GPT2, seq2seq, εκμάθηση πολλαπλών εργασιών, μεταφορά μάθησης

Abstract

Among the various approaches for building conversational agents able to entertain humans, open domain generation-based chatbots is a significant field of research. However, beyond understanding what is being discussed, human communication requires awareness of how someone is feeling. Following this perspective, in this diploma thesis, we study dialog generation and specifically we focus on the challenging task of building empathetic conversational agents, which are able to understand any implied feelings and respond accordingly.

First, we provide the reader with a brief theoretical background on machine learning (ML), deep learning (DL) and Natural Language Processing (NLP). Then we study in depth generation-based models for dialog generation. More specifically, we analyze the traditional vanilla seq2seq architecture, the vanilla seq2seq with attention and the Hierarchical Recurrent Encoder Decoder (HRED) architecture. Afterwards, we study transformer-based models that can be used in dialogue generation such as the Transformer Encoder Decoder, the BERT, the GPT-2, and the T5 models. After presenting the theoretical background of those architectures, we analyze the most commonly used decoding methods in dialog generation providing typical examples for better understanding. Finally, we present the most common automatic and human evaluation metrics/methods used for ranking dialog systems.

From the perspective of creating conversational agents that are able to understand the implied feelings of a conversation and respond accordingly, we focus on the Empathetic Dialogues task, a task proposed by Facebook. After, a brief introduction to the task and related work, we conduct several experiments and discuss the results. More specifically, at first, we analyze the datasets we used for the experiments (Empathetic Dialogues and ConvAI2) and then we present the baseline architectures used by other researchers on the task. Afterwards, we propose new ways for further improving the results of the task. More specifically, we experiment with the BERT2BERT and BERT2GPT2 architectures, achieving comparable results with already proposed models, but without reaching the state-of-the-art results. Furthermore, we experiment with three versions of the T5 model. In the first approach, we use the T5 model as is but fine-tune it on the Empathetic Dialogues dataset. In the second and the third approaches, we extend the T5 baseline architecture with multi-task learning. All of the T5-based approaches achieve state-of-the-art results in average BLEU score metric, while their performance as far as perplexity is concerned is close to the current state-of-the-art model. Moreover, after presenting the results of the experiments we provide various examples to demonstrate the performance of the proposed models more qualitatively. To further improve the proposed approach, we refer to promising future extensions and modifications that we suggest for future study.

Key words

Empathy, dialog systems, chatbots, deep learning, machine learning, natural language processing, dialogue generation, transformers, HRED, BERT, GPT2, T5, BERT2BERT, BERT2GPT2, seq2seq, multi-task learning, transfer learning

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof. Alexandros Potamianos, not only for his invaluable guidance during the conduction of my thesis but also for his motivation.

In addition, I would like to thank Athanasios Katsamanis for the guidance, the excellent cooperation we had and the time he spent on a daily basis.

Moreover, I would like to give my special thanks to Giorgos Paraskevopoulos for his invaluable support.

Last but not least, my appreciation also goes to my family and friends for their unselfish love and support during all these years. Specifically, I would like to thank them for instilling me with priceless virtues that will forever accompany me.

Emmanouil Zaranis,

Athens, November 24, 2020

Contents

List of Tables

List of Figures

Εκτεταμένη Περίληψη στα Ελληνικά

0.1 Εισαγωγή

Η ραγδαία εξέλιξη της μηχανικής μάθησης (machine learning) και πιο συγκεκριμένα των βαθιών νευρικών δικτύων (deep learning) έχει ευννοήσει σημαντικά την ανάπτυξη αυτόματων διαλογικών συστημάτων ικανών όχι μόνο να βοηθήσουν τους ανθρώπους σε συγκεκριμένες υπηρεσίες, όπως για παράδειγμα αγορά αεροπορικών εισητηρίων, κρατήσεων σε εστιατόρια, αγορές μέσω Ίντερνετ, παροχή εξυπηρέτησης πελατών κ.α, αλλά και να τους ψυχαγωγήσουν.

Ανάμεσα στις ποικίλες προσεγγίσεις για την δημιουργία αυτόματων διαλογικών συστημάτων, τα διαλογικά συστήματα ανοιχτού πεδίου που βασίζονται σε μεθόδους παραγωγής γλώσσας (generation-based open-domain chatbots) φαίνεται να έχουν ιδιαίτερο ερευνητικό ενδιαφέρον. Χάρη στην εξέλιξη των βαθιών νευρικών δικτύων, ποικίλα τέτοια συστήματα έχουν αναπτυχθεί, όντας ικανά να συνομιλήσουν με ανθρώπους παράγοντας οχί μόνο συντακτικά αλλά και νοηματικά ορθές προτάσεις. Ωστόσο, σε μία εποικοδομητική συζήτηση πέραν από την κατανόηση και την επίγνωση του θέματος συζήτησης, είναι σημαντικό κανείς να μπορεί να αντιληφθεί και τα συναισθήματα του συνομιλητή κατά τον διάλογο. Κάτι τέτοιο, ενώ είναι σχετικά απλό για τους ανθρώπους να καταλαβαίνουν και να αναγνωρίζουν τα συναισθήματα των συνομιλητών τους σε ένα διάλογο, για τα συστήματα τεχνητής νοημοσύνης (Artificial Intelligence systems) αποτελεί μια σημαντική πρόκληση.

Στα πλαίσια αυτής της διπλωματικής εργασίας, στοχεύουμε να συνεισφέρουμε στα πλαίσια του να δημιουρηθεί ένας αυτόματος συνομιλητής ικανός να ανταποκριθεί κατάλληλα στον χρήστη, κατανοώντας όχι μόνο τι συζητιέται αλλά και ποικίλα συναισθήματα που κρύβονται στην συνομιλία, ανταποκρινόμενος τελικά με παρόμοιο συναισθηματικό τρόπο. Πιο συγκεκριμένα, αρχικά εισάγουμε τον αναγνώστη στον κλάδο των διαλογικών συστημάτων και των γλωσσικών μοντέλων (language models). Στην συνέχεια, παρουσιάζουμε κάποιες βασικές έννοιες και μεθόδους της μηχανικής μάθησης και των βαθιών νευρικών δικτύων. Ύστερα, εστιάζουμε στο τομέα της Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing), μελετώντας μεθόδους δημιουργίας μαθηματικών αναπαραστάσεων των λέξεων και γενικότερα της γλώσσας (language representations) και προσπάθειες μοντελοποίησης (language modeling). Έπειτα αναλύουμε σε βάθος generation-based αρχιτεκτονικές βαθιών νευρωνικών δικτύων, παραδοσιακές αλλά και σύγχρονες, που χρησιμοποιούνται για την παραγωγή διαλόγου (dialog generation). Αφού μελετήσουμε εξονυχιστικά τις αρχιτεκτονικές αυτές, παρουσιάζουμε τις πιο ευρέως χρησιμοποιούμενες μεθόδους αποκωδικοποιήσης (decoding methods) καθώς και μεθόδους αξιολόγησης των διαλογικών συστημάτων.

Ύστερα λοιπόν από την μελέτη της παραγωγής γλώσσας με χρήση generation-based μοντέλων, εμβαθύνουμε στην δημιουργία αυτόματων διαλογικών συστημάτων με χαρακτηριστικά ενσυναίσθησης (empathetic chatbots), χρησιμοποιώντας generation-based μοντέλα. Πιο συγκεκριμένα, εστιάζοντας στην εργασία Empathetic Dialogues που προτείνεται από την ερευνητική ομάδα του Facebook, μελετάμε τις ήδη υπάρχουσες προσεγγίσεις και πειραματιζόμαστε με διαφορετικές προσεγγίσεις για την βελτίωση των αποτελεσμάτων. Τελικά, οι προτεινόμενες προσεγγίσεις μας, που βασίζονται στο μοντέλο T5, βελτιώνουν τα state-of-the-art αποτελέσματα με βάση την μετρική αξιολόγησης BLEU,

επιτυγχάνοντας παράλληλα κοντινά αποτελέσματα με αυτά των state-of-the-art μοντέλων με βάση την μετρική perplexity. Τέλος, παρουσιάζουμε κάποια χαρακτηριστικά παραδείγματα διαλόγων, για μία πιο αντικειμενική αξιολόγηση, και αφού σχολιάσουμε τα αποτελέσματα, προτείνουμε μια σειρά από μελοντικές προεκτάσεις που θα μπορούσαν να συνεισφέρουν σημαντικά στον υπό μελέτη ερευνητικό τομέα.

0.1.1 Διαλογικά Συστήματα

Τα αυτόματα διαλογικά συστήματα εμπίπτουν σε δύο γενικές κατηγορίες: τα προσανατολισμένα σε συγκεκριμένες εργασίες (task-oriented) και τα μη προσανατολισμένα σε εργασίες (non-task oriented). Τα διαλογικά συστήματα που ανήκουν στην πρώτη κατηγορία συστημάτων, έχουν σχεδιαστεί για συγκεκριμένες εργασίες και έχουν ρυθμιστεί έτσι ώστε να πραγματοποιούν σύντομες συνομιλίες με τον χρήστη. Κύριος στόχος τους είναι να βοηθήσουν τον χρήστη να ολοκληρώσει μια συγκεκριμένη εργασία, αντλώντας διάφορες πληροφορίες για τον στόχο προς επίτευξη και παράγοντας σχετικές απαντήσεις. Αυτοί οι πράκτορες μπορούν να εφαρμοστούν σε διάφορους τομείς, όπως κρατήσεις σε εστιατόρια, αεροπορικά ταξίδια, ψώνια ή και παραγγελίες φαγητού. Τέτοιους ψηφιακούς βοηθούς μπορεί κανείς να βρει σε κάθε κινητό τηλέφωνο ή οικιακούς ελεγκτές (π.χ. Siri, Cortana, Alexa, Google Now / Home κ.λπ.) των οποίων τα αυτόματα διαλογικά συστήματα μπορούν να δώσουν ταξιδιωτικές οδηγίες, να ελέγξουν οικιακές συσκευές, να βρουν εστιατόρια ή να βοηθήσουν να πραγματοποιηθούν τηλεφωνικές κλήσεις ακομά και να στείλουν κείμενα σε μορφή SMS. Το κλειδί για να θεωρήσει κανείς αυτά τα συστημάτα επιτυχημένα είναι να βοηθήσουν τον χρήστη να ολοκληρώσει την απαιτούμενη εργασία το συντομότερο δυνατό. Έτσι, η κύρια προσοχή δίνεται στην ολοκλήρωση εργασιών, έχοντας σύντομες συνομιλίες.

Από την άλλη πλευρά, οι πράκτορες που δεν είναι προσανατολισμένοι στην εκπλήρωση συγκεκριμένων εργασίων, έχουν σχεδιαστεί για εκτεταμένες συνομιλίες και με στόχο να μιμηθούν την αδόμητη συνομιλία της ανθρώπινης αλληλεπίδρασης μεταξύ ανθρώπων, αντί να επικεντρώνονται στην ολοκλήρωση εργασιών. Η έρευνα για συστήματα διαλόγου που δεν βασίζονται σε στόχους επιστρέφει στα μέσα της δεκαετίας του ’60. Ξεκίνησε με το διάσημο πρόγραμμα ELIZA [28], του Weizenbaum, ένα συστήμα βασιζόμενο μονάχα σε απλούς κανόνες κατάτμησης/ανάλυσης κειμένου. Το συγκεκριμένο σύστημα κατάφερε να μιμηθεί αρκετά πειστικά ένα ανθρωποκεντρικό ψυχοθεραπευτή με το να επαναδιατυπώνει διάφορες δηλώσεις και με το να θέτει ερωτήσεις. Στην ίδια ερευνητική κατεύθυνση κινήθηκαν στην συνέχεια και άλλοι ερευνητές, οι οποίοι χρησιμοποίησαν απλούς κανόνες ανάλυσης κειμένου για να κατασκευάσουν το σύστημα διαλόγου PARRY [29], το οποίο κατάφερε να μιμηθεί την παθολογική συμπεριφορά ενός παρανοϊκού ασθενούς στο βαθμό που οι κλινικοί γιατροί δεν μπορούσαν να το διακρίνουν από πραγματικούς ασθενείς. Στις μέρες μας, ως εξέλιξη αυτών των πρώτων συστημάτων, αναπτύχθηκαν διάφορα συστήματα που δεν εστιάζουν σε πρακτικούς σκοπούς, όπως τα προαναφερθέντα, αλλά εστιάζουν στην ψυχαγωγία όπως για παράδειγμα το σύστημα XiaoIce της Microsoft [30].

Τα αυτόματα διαλογικά συστήματα που δεν έχουν ως στόχο την εκπλήρωση συγκεκριμένων εργασιών (chatbots) μπορούν περαιτέρω να χωριστούν σε δύο κατηγορίες, ανάλογα με την αρχιτεκτονική τους: chatbots με βάση κανόνες (rule-based) και συλλογές δεδομένων (corpus-based). Τα rule-based chatbots βασίζονται σε κανόνες μετατροπής μοτίβων που χρησιμοποιούνται για την κωδικοποίηση της εισόδου και για την παραγωγή μιας απάντησης. Τα chatbots αυτής της κατηγορίας χρησιμοποιούν λέξεις-κλειδιά που σχετίζονται με μια κατάταξη, με συγκεκριμένες λέξεις να έχουν υψηλότερη κατάταξη και γενικότερες λέξεις να έχουν χαμηλή κατάταξη. Δεδομένης λοιπόν της εισόδου, τα chatbots βάσει κανόνα βρίσκουν τη λέξη με την υψηλότερη κατάταξη λέξεων-κλειδιών στην είσοδο και, στη συνέχεια, επιλέγουν τον κανόνα με την υψηλότερη κατάταξη που ταιριάζει καλύτερα στην είσοδο. Σχηματίζοντας έτσι ένα κωδικοποιημένο μοτίβο, εφαρμόζεται στην συνέχεια ένας κανόνας μετασχηματισμού στο μοτίβο αυτό για την παραγωγή μιας απάντησης. Το πρώτο chatbot που εφαρμόστηκε βασισμένο σε κανόνες ήταν το ELIZA. Σήμερα, ορισμένα μοντέρνα chatbots ακολουθούν επίσης την

αρχιτεκτονική που βασίζεται σε κανόνες, όπως το ALICE chatbot [31], το οποίο χρησιμοποιεί μια ενημερωμένη έκδοση της αρχιτεκτονικής του ELIZA.

Από την άλλη πλευρά, τα chatbots που βασίζονται σε corpus, αντί να χρησιμοποιούν χειροποίητους κανόνες, χρησιμοποιούν μια πληθώρα διαλόγων μεταξύ ανθρώππων για να δώσουν μια απάντηση. Αυτά τα chatbots χωρίζονται περαιτέρω σε δύο κατηγορίες: τα retrieval-based και τα generation-based. Τα συστήματα που ανήκουν στην πρώτη κατηγορία επιλέγουν καθοριστικά από ένα σταθερό σύνολο πιθανών απαντήσεων. Πιο συγκεκριμένα, αυτά τα chatbots αντιστοιχούν το ιστορικό διαλόγου και τις εξωτερικές γνώσεις (π.χ. μια βάση δεδομένων, η οποία μπορεί να ερωτηθεί από το σύστημα) σε μια ενέργεια απόκρισης. Συστήματα που αναζητούν μέσω μιας βάσης δεδομένων διαλόγων και επιλέγουν απαντήσεις βασιζόμενα στην ομοιότητα των ερωτοαπαντήσεων [32, 33], ανήκουν σε αυτήν την κατηγορία. Σε αντίθεση με τα συστήματα ανάκτησης, τα generation-based συστήματα προσπαθούν να παράγουν απαντήσεις δημιουργόντας μια πιθανοτική κατανομή έναντι των πιθανών απαντήσεων. Αυτά τα συστήματα δημιουργούν απαντήσεις λέξη-προς-λέξη, δειγματοληπτώντας λέξεις με βάση μια κατανομή πιθανότητας στο λεξιλόγιο που χρησιμοποιείται. Τα generation-based συστήματα μπορούν επίσης να συνδυάσουν εξωτερικές γνώσεις από βάσεις δεδομένων για να παράγουν καλύτερες απαντήσεις. Ένα από τα πιο δημοφιλή generation-based διαλογικά συστήματα είναι το Meena [34], το οποίο προτάθηκε από την Google και εκπαιδεύτηκε σε συζητήσεις ανοιχτού πεδίου προερχόμενες από τα μέσα κοινωνικής δικτύωσης. Πρόσφατα επίσης, το FacebookAI, δημιούργησε το BlenderBot [35], το μεγαλύτερο διαλογικό σύστημα ανοιχτού τομέα που έχει κατασκευαστεί μέχρι στιγμής. Αυτό το διαλογικό σύστημα συνδυάζει αρχιτεκτονικές ανάκτησης όσο και generation-based, έχοντας συνολικά 9.4 δισεκατομμύρια παραμέτρους. Είναι ένα διαλογικό σύστημα με πολλαπλές ικανότητες ενσυναίσθησης, προσωπικότητας και γνώσης, το οποίο σύμφωνα με αξιολογήσεις που βασίζονται στην ανθρώπινη κρίση φαίνεται να έλκει το ενδιαφέρον των χρηστών. Ωστόσο, σε αυτή τη διπλωματική εργασία μελετάμε μόνο generation-based συστήματα χωρίς τη χρήση εξωτερικών γνώσεων (μεθόδων ανάκτησης).

0.1.2 Μεταφορά Μάθησης (Transfer Learning) & Γλωσσικά Μοντέλα (Language Models)

Στον τομέα της επεξεργασίας φυσικής γλώσσας (NLP), τα βαθιά νευρικά δίκτυα έχουν βελτιώσει την απόδοση των μοντέλων σε πολλές εργασίες. Ωστόσο, η εκπαίδευση ενός μοντέλου από το μηδέν απαιτεί πληθώρα επισημειωμένων δεδομένων (labeled data). Για παράδειγμα, η εκπαίδευση ενός συστήματος διαλόγου απαιτεί εκατομμύρια δεδομένα, με συγκεκριμένες επισημειωμένες απαντήσεις (targets). Ωστόσο, σε πολλές πρακτικές εφαρμογές, υπάρχουν λίγα διαθέσιμα επισημειωμένα δεδομένα για την εποπτεία της εκπαίδευσης του μοντέλου. Σε αυτές τις περιπτώσεις, η χρήση της μεταφοράς μάθησης (transfer learning) προσφέρει μια εναλλακτική λύση. Τα μοντέλα προεκπαιδεύονται σε μια παρόμοια εργασία και, στη συνέχεια, προσαρμόζονται στην απαιτούμενη εργασία, χρησιμοποιώντας τις γνώσεις που αποκτήθηκαν από τη διαδικασία προεκπαίδευσης. Έτσι, προκειμένου να επιτύχουμε ικανοποιητικά αποτελέσματα κατά τη δημιουργία ενός αυτόματου συνομιλητή, προεκπαιδεύουμε ένα μοντέλο ως γλωσσικό μοντέλο. Με αυτόν τον τρόπο, το μοντέλο που χρησιμοποιείται είναι σε θέση να δημιουργήσει υψηλού επιπέδου αναπαραστάσεις της γλώσσας, χρησιμοποιώντας αυτήν τη γνώση κατά τη διάρκεια της βελτιστοποίησης (fine-tuning).

Στη μοντελοποίηση γλωσσών, το μοντέλο υπολογίζει μια πιθανότητα κατανομής σε μια ακολουθία λέξεων, δημιουργώντας μια μοναδική αναπαράσταση για κάθε λέξη που προέκυψε, βάσει ενός προηγούμενου περιεχομένου. Με άλλα λόγια, ένα γλωσσικό μοντέλο καλείται να υπολογίσει την πιθανότητα εμφάνισης ενός αριθμού λέξεων σε μια συγκεκριμένη ακολουθία. Πιο συγκεκριμένα, ένα γλωσσικό μοντέλο δεδομένης μιας ακολουθίας λέξεων ως είσοδο, προσπαθεί να προβλέψει την επόμενη λέξη. Με αυτόν τον τρόπο λοιπόν, το μοντέλο μπορεί να δημιουργήσει αναπαραστάσεις λέξεων με βάση το περιεχόμενο/ιστορικό. Ως αποτέλεσμα, δεν απαιτεί επισημειωμένα εκπαιδευτικά δεδομένα, που είναι δύσκολο να βρεθούν. Το απλό κείμενο, ωστόσο, διατίθεται σε μεγάλες ποσότητες για κάθε

πιθανή εργασία. Έτσι, τα γλωσσικά μοντέλα μπορούν να εκπαιδευτούν σε μια πληθώρα δεδομένων που είναι διαθέσιμα δωρεάν. Λόγω της ικανότητας των γλωσσικών μοντέλων να εξάγουν γενικές αναπαραστάσεις λέξεων με βάση το περιεχόμενο/ιστορικό, χρησιμοποιούνται πλέον ευρέως σε πολλούς τομείς. Ένας τέτοιος τομέας είναι τα διαλογικά συστήματα.

0.1.3 Εκμάθηση Πολλαπλών Εργασιών (Multi-task Learning)

Η εκμάθηση πολλαπλών εργασιών είναι μια τεχνική μάθησης στην οποία θέλουμε να εκπαιδεύσουμε ένα μοντέλο σε πολλές σχετικές εργασίες ταυτόχρονα. Με άλλα λόγια, θέλουμε το μοντέλο να κάνει προβλέψεις για κάθε μία από τις εργασίες ταυτόχρονα. Ο στόχος αυτού είναι να εκμεταλλευτούμε τις πληροφορίες σε ένα από τα προβλήματα, ώστε να μπορούμε να βελτιώσουμε την απόδοση στα υπόλοιπα προβλήματα. Στην μάθηση με χρήση βαθιών νευρωνικών δικτύων, η βασική ιδέα της εφαρμογής εκμάθησης πολλαπλών εργασιών είναι να έχουμε διαφορετικά δίκτυα που αποτελούν μέρος της ίδιας δομής και να έχουν μερικές κοινές παραμέτρους. Έτσι, το κοινόχρηστο μέρος επηρεάζεται από όλα τα προβλήματα, ενώ τα μη κοινόχρηστα μέρη επηρεάζονται από τα δεδομένα εκπαίδευσης κάθε εργασίας ανεξάρτητα.

0.2 Αυτόματη Παραγωγή Διαλόγου με χρήση Generation-based Μοντέλων

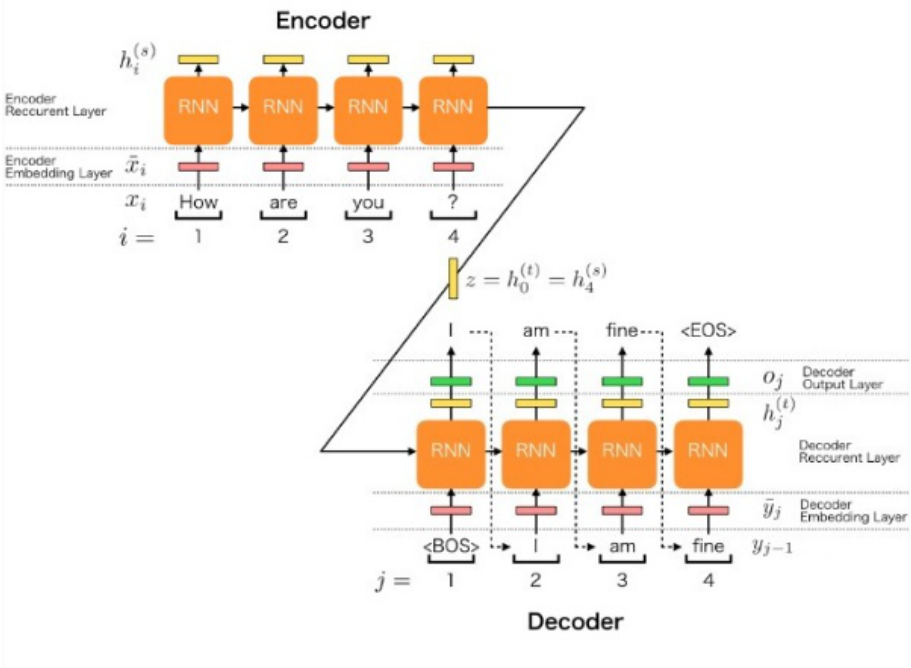
Η βασική ιδέα πίσω από τα generation-based μοντέλα είναι η παραγωγή προτάσεων λέξη προς λέξη, βασιζόμενα σε ένα προηγούμενο περιεχόμενο/ιστορικό που στην πιο απλή περίπτωση μπορεί να αποτελείται από την προηγούμενη στροφή (turn) του διαλόγου. Διάφορες generation-based αρχιτεκτονικές έχουν προταθεί για διαλογικά συστήματα. Παρακάτω θα μελετήσουμε κάποιες βασικές παραδοσιακές αρχιτεκτονικές (vanilla seq2seq και vanilla seq2seq με attention) και στην συνέχεια θα εστιάσουμε και σε πιο συγχρονες (Transformer Encoder Decoder, BERT, GPT2 και T5). Επιπλέον, θα μελετήσουμε τις βασικές μεθόδους αποκωδικοποίησης που χρησιμοποιούνται κατά την παραγωγή απαντήσεων, καθώς επίσης και τις κύριες μετρικές που χρησιμοποιούνται σήμερα για την αξιολόγηση των διαλογικών συστημάτων.

0.2.1 Vanilla seq2seq

Μια από τις απλούστερες αρχιτεκτονικές που μπορεί να χρησιμοποιηθεί σε διαλογικά συστήματα είναι αυτή των μοντέλων ακολουθίας σε ακολουθία (seq2seq). Το απλούστερο μοντέλο αυτής της κατηγορίας είναι το Vanilla seq2seq. Η ιδέα του μοντέλου είναι η αντιστοίχιση μιας ακολουθίας εισόδου μεταβλητού μήκους σε μια ακολουθία εξόδου επίσης μεταβλητού μήκους. Για να επιτευχθεί η αντιστοίχιση αυτή χρησιμοποιείται ένα μοντέλο κωδικοποιητή (encoder) για την απόκτηση μιας σταθερού μήκους διανυσματικής αναπαράστασης της ακολουθίας εισόδου, που αντιπροσωπεύει τις κωδικοποιημένες πληροφορίες. Επιπλέον, χρησιμοποιείται και ένα μοντέλο αποκωδικοποιητή (decoder) για την εξαγωγή της ακολουθίας εξόδου, αποκωδικοποιώντας τη σταθερού μήκους διανυσματική αναπαράσταση που έχει προκύψει από τον κωδικοποιητή και δημιουργώντας έτσι μια απάντηση λέξη προς λέξη. Η περιγραφόμενη αρχιτεκτονική απεικονίζεται στην Εικόνα 0.1.

0.2.2 Vanilla seq2seq με attention

Το κύριο μειονέκτημα του μοντέλου Vanilla seq2seq έγγειται στο γεγονός οτι είναι σχεδόν ανίκανο να αντιπροσωπεύει μακροπρόθεσμες ακολουθίες με τη χρήση μιας απλής διανυσματικής αναπαράστασης σταθερού μήκους. Προκειμένου λοιπόν να μοντελοποιηθούν καλύτερα μακροπρόθεσμες ακολουθίες, ο μηχανισμός προσοχής (attention mechanism) εφαρμόστηκε στον τομέα του NLP. Ο μηχανισμός προσοχής, αντί να βασίζεται μόνο στην κρυφή κατάσταση (hidden state) του αποκωδικοποιητή, αναγκάζει το μοντέλο να μάθει να εστιάζει (να παρακολουθεί) σε συγκεκριμένα μέρη της ακολουθίας εισόδου κατά την αποκωδικοποίηση, διαφορετικά κάθε φορά που παράγει μία νέα λέξη. Στην Εικόνα

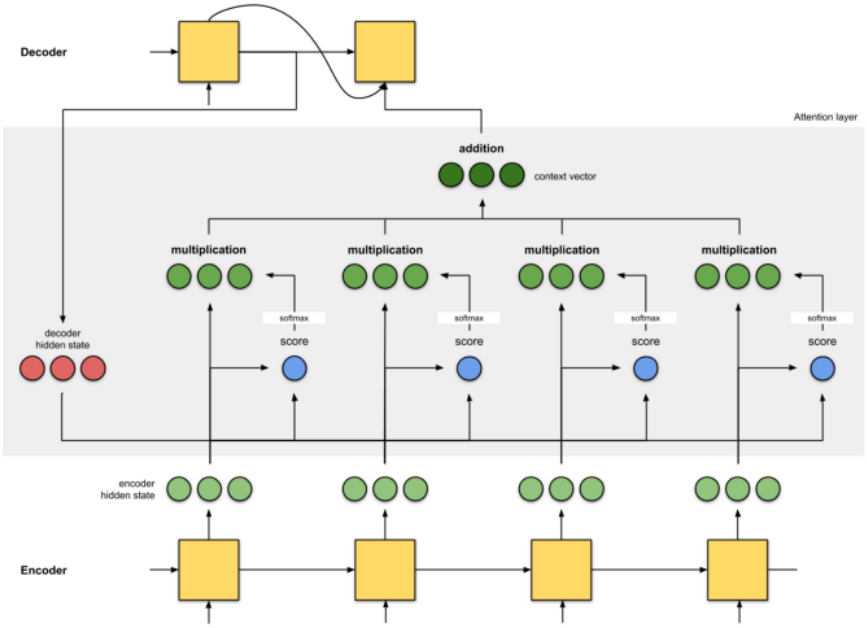


Σχήμα 0.1: Η αρχιτεκτονική Vanilla seq2seq, αποτελούμενη από τον κωδικοποιητή και τον αποκωδικοποιητή. Πηγή: [1]

0.2 απεικονίζεται ο μηχανισμός προσοχής που υλοποιήθηκε από τον Bahdanau [12]. Με βάση αυτόν τον μηχανισμό για κάθε λέξη υπολογίζουμε μία "βαθμολογία προσοχής" (attention score), η οποία καθορίζει κατά πόσο πρέπει να ληφθεί υπόψιν η συγκεκριμένη κρυφή κατάσταση του κωδικοποιητή στην έξοδο. Στην συνέχεια, με μία σειρά από κατάλληλες πράξεις περνάμε αυτή την πληροφία στην κρυφή κατάσταση του αποκωδικοποιητή και παράγουμε την επόμενη λέξη.

0.2.3 Transformer Encoder Decoder

Το 2017, προτάθηκε η αρχιτεκτονική του δικτύου Transformer [3], επιτυγχάνοντας όχι μόνο καλύτερη ποιότητα των παραγόμενων απαντήσεων, αλλά απαιτώντας πολύ λιγότερο χρόνο εκπαίδευσης. Το μοντέλο Transformer βασίζεται εξ ολοκλήρου στο μηχανισμό αυτοπροσοχής (self-attention) για τον υπολογισμό των αναπαραστάσεων της εισόδου και της εξόδου. Αποτελείται από μια στοίβα κωδικοποιητών και μία αποκωδικοποιητών. Η στοίβα του κωδικοποιητή αποτελείται από N πανομοιότυπα επίπεδα κωδικοποιητών. Κάθε ένα από αυτά αποτελείται από 2 υποεπίπεδα. Το πρώτο υποεπίπεδο είναι ένας μηχανισμός αυτοπροσοχής πολλαπλών κεφαλών (multi-head self-attention), ένω το δεύτερο είναι ενα πλήρως συνδεδεμένο δίκτυο τροφοδοσίας προς τα εμπρός. Μέσω του μηχανισμού αυτοπροσοχής ο κωδικοποιητής βρίσκει τις αλληλοσυχεστίσεις των λέξεων που αποτελούν την είσοδο, δίνοντας στην συνέχεια κατάλληλη έμφαση σε κάθε μια από αυτές. Όσον αφορά την στοίβα του αποκωδικοποιητή αποτελείται και εκείνη από N πανομοιότυπα επίπεδα αποκωδικοποιητών. Εκτός από τα δύο υποεπίπεδα που συναντήσαμε στον κωδικοποιητή, κάθε στρώμα αποκωδικοποιητή έχει επίσης ένα τρίτο υποεπίπεδο, το οποίο εφαρμόζει αυτοπροσοχή πολλαπλών κεφαλών (multi-head self-attention) στις εξόδους της στοίβας κωδικοποιητή. Η αυτοπροσοχή σε κάθε υποεπίπεδο αποκωδικοποιητή έχει τροποποιηθεί με τέτοιο τρόπο ώστε να αποφεύγεται η προσοχή (παρακαλούθηση) σε επόμενες θέσεις, και να εστιάζει σε προηγούμενες. Η απεικόνιση του μοντέλου φαίνεται στην Εικόνα 0.3.



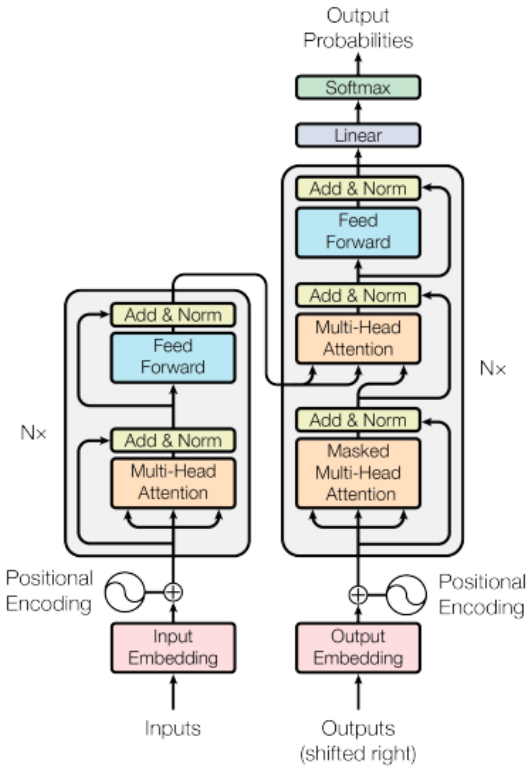
Σχήμα 0.2: Η αρχιτεκτονική Vanilla seq2seq ενισχυμένη με τον μηχανισμό προσοχής (attention). Πηγή: [2]

0.2.4 BERT

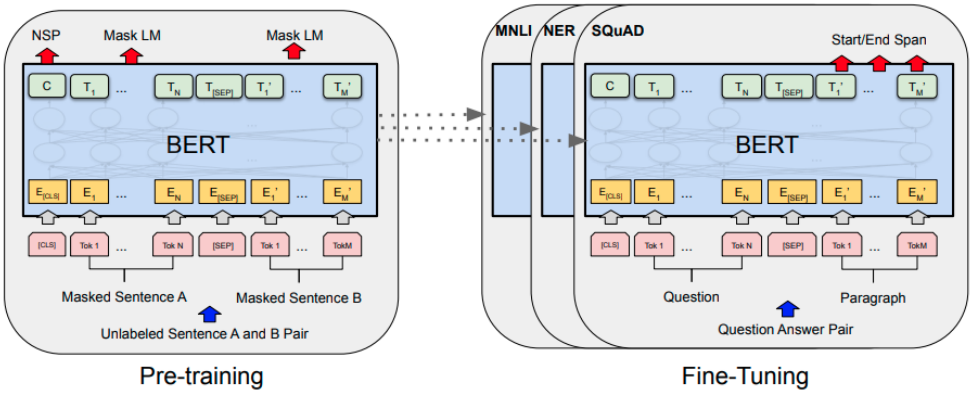
Το BERT μοντέλο βασίζεται αρχιτεκτονικά στην ιδέα του Transformer. Αποτελείται από μια στοίβα κωδικοποιητών χρησιμοποιώντας την δυνατότητα αυτοπροσοχής πολλών κεφαλών (multi-head self-attention) σε δύο κατευθύνσεις. Ένας από τους κύριους λόγους για την καλή απόδοση του BERT σε διαφορετικές εργασίες (tasks) είναι η προεκπαίδευσή του σε δύο μη εποπτευόμενες εργασίες. Με αυτόν τον τρόπο, το μοντέλο έχει τη δυνατότητα να κατανοεί τα μοτίβα της γλώσσας. Η πρώτη εργασία στην οποία το μοντέλο είναι προεκπαιδευμένο ονομάζεται "μοντελοποίηση γλώσσας με κενά" (masked language modeling - MLM). Σε αυτήν την εργασία, το 15% των λέξεων κάθε ακολουθίας καλύπτεται τυχαία και το μοντέλο προσπαθεί να προβλέψει τις λέξεις αυτές. Η δεύτερη εργασία ονομάζεται "πρόβλεψη επόμενης πρότασης" (Next Sentence Prediction - NSP). Πολλές σημαντικές μεταγενέστερες εργασίες, όπως η απάντηση ερωτήσεων (QA) και η "κατανόηση φυσικής γλώσσας" (Natural Language Inference - NLI) βασίζονται στην κατανόηση της σχέσης μεταξύ δύο προτάσεων, οι οποίες δεν καταγράφονται άμεσα από τη μοντελοποίηση γλωσσών. Κατά συνέπεια, προκειμένου να μπορεί το μοντέλο να κατανοεί τις σχέσεις των προτάσεων, είναι προ-εκπαιδευμένο να προβλέπει αν μία πρόταση (B) αποτελεί συνέχεια μίας άλλης πρότασης (A) σε ένα κείμενο. Για τη διαδικασία προεκπαίδευσης χρησιμοποιούνται κείμενα από το BooksCorpus και την Αγγλική Βικιπαίδεια. Η διαδικασία προεκπαίδευσης (pre-training) και βελτιστοποίησης εκπαίδευσης (fine-tuning) απεικονίζεται στην Εικόνα 0.4.

0.2.5 GPT2

Η αρχιτεκτονική του μοντέλου δεν είναι μια ιδιαίτερα νέα αρχιτεκτονική, καθώς είναι σαν και αυτή του αποκωδικοποιητή του Transformer (decoder-only transformer). Το GPT2 μοντέλο δηλαδή αποτελείται μόνο από μια στοίβα αποκωδικοποιητών. Κάθε αποκωδικοποιητής αποτελείται από δύο υποεπίπεδα, το επίπεδο πολλών κεφαλών αυτοπροσοχής (multi-head self-attention) και το επίπεδο με το πλήρες δίκτυο τροφοδοσίας προς τα εμπρός. Στο επίπεδο αυτοπροσοχής χρησιμοποιείται "masked self-attention", δηλαδή το μοντέλο εστιάζει σε λέξεις που βρίσκονται σε προηγούμενες θέσεις από την τρέχουσα. Το μοντέλο είναι εκπαιδευμένο ως γλωσσικό μοντέλο, στο σύνολο δεδομένων WebText,



Σχήμα 0.3: Η αρχιτεκτονική Transformer Encoder Decoder. Πηγή: [3]

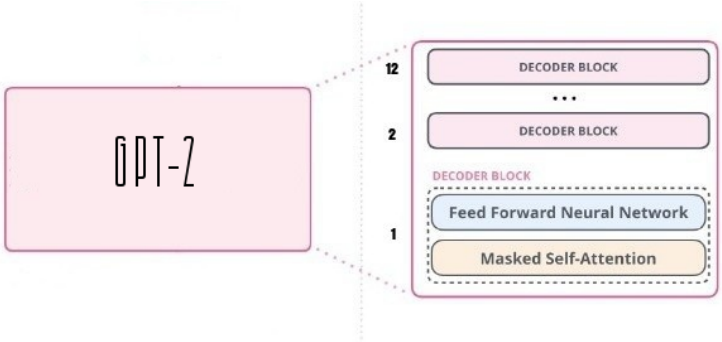


Σχήμα 0.4: Διαδικασίες pre-training και fine-tuning του μοντέλου BERT. Πηγή: [4]

προσπαθώντας να προβλέψει την επόμενη λέξη. Η απεικόνιση του μοντέλου φαίνεται στην Εικόνα 0.5.

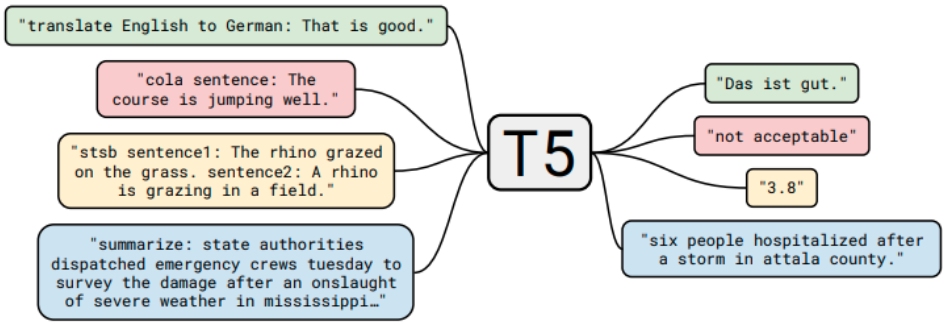
0.2.6 Text-to-Text Transfer Transformer (T5)

Το T5 μοντέλο ακολουθεί ακριβώς την ίδια αρχιτεκτονική του Transformer Encoder Decoder μοντέλου. Ως προς την εκπαίδευση, η βασική ιδέα πίσω από το μοντέλο T5 είναι η μετατροπή όλων των εργασιών NLP σε μια ενοποιημένη μορφή κειμένου σε κείμενο (text-to-text) όπου η είσοδος και η έξοδος είναι πάντα συμβολοσειρές κειμένου. Αυτή η προσέγγιση μορφής κειμένου σε κείμενο μας επιτρέπει να εφαρμόζουμε εύκολα το ίδιο μοντέλο, την ίδια διαδικασία εκπαίδευσης και την ίδια διαδικασία αποκωδικοποίησης σε κάθε εργασία που εξετάζεται, όπως η απάντηση ερωτήσεων (question answering), η συνοπτική παρουσίαση εγγράφων (document summarization), η ταξινόμηση συναι-



Σχήμα 0.5: Αναπαράσταση του GPT2 μοντέλου.

σθημάτων (sentiment classification) και οι εργασίες μηχανικής μετάφρασης (machine translation). Οι ερευνητές, για να εκπαιδεύσουν ένα μεμονωμένο μοντέλο για το διαφορετικό σύνολο εργασιών που περιγράφονται παραπάνω, πρόσθεσαν ένα πρόθεμα (prefix) στην αρχική ακολουθία εισόδου πριν την τροφοδοτήσουν στον κωδικοποιητή, προκειμένου να καθορίσουν ποια εργασία θα πρέπει το μοντέλο να εκτελέσει. Για παράδειγμα, προκειμένου το μοντέλο να μεταφράσει την πρόταση “That is good.” από τα Αγγλικά στα Γερμανικά, η πρόταση “translate English to German: That is good.” τροφοδοτείται ως είσοδος στο μοντέλο, και η πρόταση “Das ist gut.” χρησιμοποιείται ως στόχος (target). Το μοντέλο είναι προεκπαιδευμένο σε δεδομένα χωρίς ετικέτες, προκειμένου να γενικευθεί η γνώση που θα είναι χρήσιμη κατά τη διάρκεια της βελτιστοποίησης της εκπαίδευσης (fine-tuning). Χρησιμοποιήθηκε το “Colossal Clean Crawled Corpus (C4)” σύνολο δεδομένων. Η τεχνική του “masked language modeling”, που χρησιμοποιήθηκε και στο BERT, εφραμόστηκε προκειμένου το μοντέλο να μάθει να προβλέπει τις κρυμμένες λέξεις. Όσον αφορά τη διαδικασία βελτιστοποίησης (fine-tuning), οι ερευνητές πειραματίστηκαν με διαφορετικές προσεγγίσεις, όπως η εφαρμογή fine-tuning σε κάθε εργασία ξεχωριστά, η εκμάθηση πολλαπλών εργασιών (multi-task learning), η εκπαίδευση πολλαπλών εργασιών αφήνοντας μία εργασία (leave-one-out multi-task learning) κ.α. Ο μηχανισμός λειτουργίας του μοντέλου για διαφορετικές εργασίες φαίνεται στην Εικόνα 0.6.



Σχήμα 0.6: Απεικόνιση του μηχανισμού λειτουργίας του μοντέλου T5 σε διαφορετικές εργασίες. Πηγή: [5]

0.2.7 Μέθοδοι Αποκωδικοποίησης

Εκτός από τις βελτιωμένες αρχιτεκτονικές των Transformers και τα πάρα πολλά μη εποπτευόμενα δεδομένα εκπαίδευσης, οι μέθοδοι αποκωδικοποίησης παίζουν επίσης σημαντικό ρόλο στη δημιουργία συνεκτικών και άπταιστων απαντήσεων. Η μέθοδος αποκωδικοποίησης είναι μια στρατηγική που εφαρμόζεται στον αποκωδικοποιητή, σύμφωνα με την οποία επιλέγουμε από το λεξιλόγιο του μοντέλου που χρησιμοποιείται, ποια λέξη θα παραχθεί. Οι βασικότερες μέθοδοι αποκωδικοποίησης που χρησιμοποιούνται σήμερα είναι:

- **Άπληστη αποκωδικοποίηση (greedy decoding):** Η άπληστη αποκωδικοποίηση είναι η απλούστερη μέθοδος αποκωδικοποίησης που μπορεί να χρησιμοποιηθεί για τη δημιουργία προτάσεων. Σύμφωνα με αυτήν τη μέθοδο, δεδομένων των προηγούμενων λέξεων που δημιουργήθηκαν, σε κάθε χρονικό βήμα, επιλέγουμε τη λέξη με την υψηλότερη πιθανότητα εμφάνισης.

- **Ακτινωτή αποκωδικοποίηση (beam search):** Η ακτινωτή αποκωδικοποίηση αναλύει περισσότερες διαδρομές σε κάθε χρονικό βήμα (αντί να επιλέγει απλά την λέξη με την μεγαλύτερη πιθανότητα εμφάνισης). Έτσι, σε κάθε χρονικό βήμα, διατηρεί τις περισσότερες πιθανότητες υποθέσεων και τελικά επιλέγει την υπόθεση που έχει τη συνολική υψηλότερη πιθανότητα. Ο αριθμός των υποθέσεων που διατηρούνται σε κάθε χρονικό βήμα είναι μια παράμετρος που καθορίζεται από τον ερευνητή, που ονομάζεται αριθμός “beams”.

- **Top-k δειγματοληψία (Top-k sampling):** Κατά τη δειγματοληψία top-k, οι k πιθανότερες επόμενες λέξεις φιλτράρονται και η μάζα πιθανότητας ανακατανέμεται μόνο σε αυτές τις λέξεις k. Ωστόσο, αυτό μπορεί να είναι προβληματικό καθώς ορισμένες λέξεις μπορεί να ληφθούν ως δείγμα από μια πολύ εστασμένη κατανομή, ενώ άλλες από μια πολύ πιο επίπεδη κατανομή.

- **Top-p δειγματοληψία (Top-p (nucleus) sampling):** Στη δειγματοληψία top-p επιλέγουμε λέξεις από το μικρότερο δυνατό σύνολο λέξεων, των οποίων η συσσωρευτική πιθανότητα υπερβαίνει ένα όριο πιθανότητας p. Αυτή η μέθοδος αποκωδικοποίησης επιτρέπει δυναμική επιλογή λέξεων.

- **Top-k Top-p δειγματοληψία (Top-k Top-p Sampling):** Σε αυτή την περίπτωση συνδυάζονται οι δύο προηγούμενες μέθοδοι. Στο πρώτο βήμα εφαρμόζουμε top-k φιλτράρισμα και στην συνέχεια top-p δειγματοληψία.

0.2.8 Μέθοδοι Αξιολόγησης

Οι μέθοδοι αξιολόγησης των απαντήσεων ενός διαλογικού συστήματος μπορούν να διακριθούν σε αυτόματες και σε αυτές που γίνονται με βάση την ανθρώπινη κρίση. Οι πιο συνηθισμένες μετρικές που χρησιμοποιούνται και ανήκουν στην πρώτη κατηγορία είναι:

- Perplexity
- BLEU
- Distinct-n

ενώ αυτές που ανήκουν στην δεύτερη είναι:

- Σύγκριση κατά ζεύγη
- Αξιολόγηση συνάφειας
- Αξιολόγηση συνοχής

Οι παραπάνω μέθοδοι περιγράφονται αναλυτικά στην ενότητα 4.11 στο πλήρες κείμενο (αγγλικό κείμενο).

0.3 Διαλογικά Συστημάτα Ενσυναίσθησης με χρήση Generation-based Μοντέλων

Τα σύγχρονα διαλογικά συστήματα επιτυγχάνουν εντυπωσιακά αποτελέσματα, όντας ικανά να επικοινωνούν με τον χρήστη, διατηρώντας το ενδιαφέρον του. Ωστόσο πέρα από την κατανόηση του τι συζητείται, είναι σημαντικό ένα διαλογικό σύστημα να αντιλαμβάνεται τι αισθάνεται ο χρήστης.

Ενώ κάτι τέτοιο συμβαίνει ασυνείδητα κατά τον διάλογο που αναπτύσσεται μεταξύ ανθρώπων, για τα διαλογικά συστήματα αποτελεί σημαντική πρόκληση. Βασιζόμενοι λοιπόν στην πρόκληση του να δημιουργήσουμε διαλογικά συστήματα ικανά να κατανοούν και να αναγνωρίζουν τα συναισθήματα που κρύβονται πίσω από μια συζήτηση, καθώς και να παράγουν απαντήσεις κατάλληλου συναισθηματικού περιεχομένου, δείχνοντας ενσυναίσθηση, μελετάμε την εργασία με θέμα "Διάλογοι με Ενσυναίσθηση", που προτάθηκε από την Facebook. Στα πλαίσια λοιπόν αυτής της μελέτης, αρχικά παρουσιάζουμε τα σύνολα δεδομένων (datasets) τα οποία χρησιμοποιήσαμε, μελετάμε αρχιτεκτονικές βασισμένες σε generation-based μοντέλα που έχουν προταθεί από άλλους ερευνητές και προτείνουμε δικές μας ιδέες. Στην συνέχεια παρουσιάζουμε τα αποτελέσματα των πειραμάτων μας συγκρίνοντας τα αποτελέσματα.

0.3.1 Σύνολα Δεδομένων (Datasets)

Στα πλαίσια αυτής της διπλωματικής χρησιμοποιήσαμε το Empathetic Dialogues dataset και το ConvAI2 dataset, τα οποία περιγράφουμε εν συντομία παρακάτω.

- **Empathetic Dialogues Dataset:** Το Empathetic Dialogues Dataset αποτελεί το κύριο dataset που χρησιμοποιήσαμε για βελτιστοποίηση (fine-tuning) των μοντέλων μας. Αποτελείται από συζητήσεις ανοιχτού περιεχομένου (open-domain) μεταξύ δύο συνομιλητών. Κάθε συζήτηση βασίζεται σε μία κατάσταση στην οποία έχει βρεθεί ο ένας εκ των δύο συνομιλητών και σχετίζεται με ένα συγκεκριμένο συναίσθημα. Τα βασικά στατιστικά χαρακτηριστικά του dataset φαίνονται στον πίνακα 0.1.

Πίνακας 0.1: Στατιστικά του Empathetic Dialogue dataset

	Train	*Valid.*	*Test*
Αριθμός διαλόγων	19433	2770	2547
Αριθμός στροφών (turns)	84324	12078	10973
Μέσος αριθμός στροφών διαλόγων	4.31	4.36	4.31

Για πλήρη κατανόηση παρουσιάζουμε ένα παραδείγμα από το σύνολο εκπαίδευσης στην Εικόνα 0.7.

Label: Content
Situation: Speaker felt this when...
"eating my favorite meal makes me happy."
Conversation:
Speaker: i am at my best when i have my favorite meal.
Listener: nice
Speaker: i love enchiladas
Listener: really?
Speaker: yes. enchiladas for the win!

Σχήμα 0.7: Παραδείγμα διαλόγου από το σύνολο εκπαίδευσης του Empathetic Dialogues dataset.

- **ConvAI2 Dataset:** Το ConvAI2 Dataset αποτελεί το δεύτερο dataset που χρησιμοποιήσαμε για προεκπαίδευση (pretraining) των μοντέλων. Αποτελείται και αυτό από συζητήσεις ανοιχτού περιεχομένου (open-domain) μεταξύ δύο συνομιλητών, όπου κάθε ένας έχει ένα συγκεκριμένο προφίλ προσωπικότητας. Τα βασικά στατιστικά χαρακτηριστικά του dataset φαίνονται στον πίνακα 0.2.

Πίνακας 0.2: Στατιστκά του ConvAI2 dataset

	$Train$	$Valid.$	$Test(Hidden)$
Αριθμός διαλόγων	17,878	1000	1015
Αριθμός στροφών (turns)	131438	7801	6634
Αριθμός προσωπικοτήτων	1155	100	100

Ομοίως με πριν, για βαθύτερη κατανόηση παρουσιάζουμε ένα παράδειγμα στον πίνακα 0.2.

Πίνακας 0.3: Παράδειγμα διαλόγου του ConvAI2

Persona 1	**Persona 2**
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

0.3.2 Βασικές Αρχιτεκτονικές (Baseline Architectures)

Σε αυτό το μέρος αναφέρουμε τις βασικές αρχιτεκτονικές που χρησιμοποιήθηκαν από άλλους ερευνητές (πίνακας 0.4). Για περισσότερες πληροφορίες παρακαλούμε τον αναγνώστη να ανατρέξει στην πλήρη διπλωματική εργασία (αγγλικό κείμενο).

0.3.3 Προτεινόμενες Αρχιτεκτονικές (Proposed Architectures)

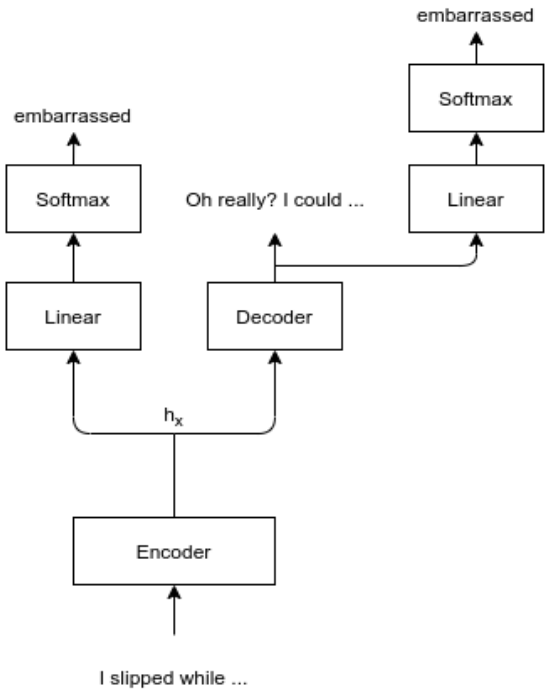
Σε αυτό το μέρος παρουσιάζουμε τα generation-based μοντέλα που χρησιμοποιήσαμε. Αρχικά πειραματιστήκαμε με τα μοντέλα BERT2BERT και BERT2GPT2, και στην συνέχεια με 3 προσεγγίσεις βασισμένες στο μοντέλο T5. Πιο αναλυτικά:

- **BERT2BERT:** Το μοντέλο BERT2BERT ακολουθεί την seq2seq αρχιτεκτονική. Αποτελείται από έναν BERT encoder και έναν BERT decoder.
- **BERT2GPT2:** Το μοντέλο BERT2BERT ακολουθεί και αυτό την seq2seq αρχιτεκτονική. Αποτελείται από έναν BERT encoder και έναν GPT2 decoder.
- **T5-baseline:** Σε αυτή την προσέγγιση χρησιμοποιήσαμε το T5 μοντέλο ως έχει.

Πίνακας 0.4: Βασικές Αρχιτεκτονικές (Baseline Architectures)

Όνομα Μοντέλου	Προεκπαίδευση	Χαρακτηριστικά	Βιβλ.
(1) Vaswani Full Transformer	Reddit Dataset	transformer	[26, 25]
(2) Multitask Transformer	Reddit Dataset	transformer, multitasking	[26, 25]
(3) Prepend-k	Reddit Dataset	transformer, prepending	[26, 25]
(4) Ensemble of Encoders	Reddit Dataset	transformer, ensembling	[26, 25]
(5) CAiRE	Book Corpus, PersonaChat	GPT, multitasking	[27]
(6) GPT2-baseline	WebText	GPT2	[36]
(7) GPT2-prepend	WebText	GPT2, prepending	[36]
(8) DodecaDialogue MT	Reddit, Twitter	transformer, multitasking	[37]
(9) DodecaDialogue MT+FT	Reddit, Twitter	transformer, multitasking, finetuning	[37]
(10) BST Generative	Reddit Dataset	transformer, multitasking	[35]

- **T5-multitask1:** Σε αυτή την προσέγγιση προεκτείναμε το T5-baseline, προσθέτοντας έναν ταξινομητή πάνω από τον encoder. Έτσι με την χρήση της εκμάθησης πολλαπλών εργασιών (multi-task learning) προσπαθήσαμε να βελτιώσουμε την αναγνώριση συναισθημάτος στον διάλογο και παράλληλα να ληφθεί αύτη υπόψιν κατά την αποκωδικοποίηση.
- **T5-multitask2:** Σε αυτή την προσέγγιση προεκτείναμε το T5-multitask1, προσθέτοντας έναν ακόμη ταξινομητή πάνω από τον decoder. Κατά την εκπαίδευση χρησιμοποιήσαμε τις ίδιες ετικέτες συναισθήματος για τους δύο ταξινομητές με σκοπό την καλύτερη μοντελοποίηση της ενσυναίσθησης. Αυτή η προσέγγιση θεωρητικά δεν έχει μόνο στόχο να παράξει μία απάντηση αλλά και να "τιμωρήσει" το μοντέλο στην περίπτωση που το συναίσθημα της απάντησης δεν είναι το επιθυμητό. Η αρχιτεκτονική του μοντέλου φαίνεται στην Εικόνα 0.8.



Σχήμα 0.8: Αναπαράσταση της αρχιτεκτονικής του μοντέλου T5-multitask2.

0.3.4 Πειράματα και Αποτελέσματα

Σε αυτή την υποενότητα, παρουσιάζουμε τα αποτελέσματα των μοντέλων που χρησιμοποιήθηκαν από άλλους ερευνητές, καθώς επίσης και αυτά που προέκυψαν από τις αρχιτεκτονικές που προτείναμε. Χρησιμοποιούμε τις μετρικές Perplexity (PPL) και BLEU score για αξιολόγηση. Αναφέρουμε εδώ, ότι η ένδειξη "(P+ED)" σε κάποια από τα μοντέλα μας δηλώνει ότι το μοντέλο προεκπαιδεύτηκε πρώτα στο ConvAI2 Dataset και στην συνέχεια βελτιστοποήθηκε στο Empathetic Dialogues, ενώ η ένδειξη "(ED)" δηλώνει ότι δεν έγινε κάποια επιπλέον προεκπαίδευση. Τα αποτελέσματα φαίνονται στον πίνακα 0.5.

Πίνακας 0.5: Πίνακας Αποτελεσμάτων

Μοντέλο	PPL	AVG BLEU
Vaswani Full Transformer [26, 25]	21.24	6.27
Multitask Transformer [26, 25]	24.07	5.42
EmoPrepend-1 [26, 25]	24.30	4.36
TopicPrepend-1 [26, 25]	25.40	4.17
Ensem-DM [26, 25]	19.05	6.83
Ensem-DM+ [26, 25]	19.10	6.77
CAiRE [27]	13.32	7.03
GPT2-baseline [36]	18.32*	7.71*
GPT2-prepend [36]	19.49*	7.78*
BST Generative [35]	11.48*	-
DodecaDialogue MT+FT [37]	**11.4**	8.1
DodecaDialogue MT [37]	11.5	8.4
Ours-Vaswani Full Transformer (ED)	33.46	-
Ours-Vaswani Full Transformer (P+ED)	28.64	-
BERT2BERT (ED)	20.77	5.53
BERT2BERT (P+ED)	19.54	6.78
BERT2GPT2 (ED)	17.93	7.22
BERT2GPT2 (P+ED)	21.48	7.19
T5 (ED)	12.40	9.31
T5 (P+ED)	12.51	**9.68**
T5-multitask1	12.58	9.28
T5-multitask2	12.96	9.13

Το * δηλώνει ότι τα αποτελέσματα μετρώνται στο validation σύνολο

Παρατηρώντας τα αποτελέσματα συμπεραίνουμε σε συντομία τα παρακάτω:

- Παρατηρούμε ότι το μοντέλο DocedaDialogue MT+FT παρουσιάζει state-of-the-art αποτελέσματα όσον αφορά το Perplexity.

- Παρατηρούμε επίσης ότι όσον αφορά τα μοντέλα Ours-Vaswani Full Transformer (ED) και Ours-Vaswani Full Transformer (P+ED) (τα οποια υλοποιήσαμε εμείς) παράγουν αποτελέσματα τα οποία απέχουν αρκετά από εκείνα των υπόλοιπων μοντέλων. Ο λόγος για τον οποίον αποτυγχάνουν αυτά τα δύο μοντέλα να εκπαιδευτούν σωστά είναι ο περιορισμένος αριθμός των δεδομένων.

- Στην περίπτωση των BERT2BERT και BERT2GPT2 μοντέλων, παρατηρούμε οτι τα παραγώμενα αποτελέσματα είναι συγκρίσιμα με αυτά των αρχικών προσεγγίσεων [26, 25]. Συγκεκριμένα το BERT2GPT2 παράγει καλύτερα αποτελέσματα και στις δύο μετρικές. Ωστόσο όμως

τα δύο αυτά μοντέλα δεν παράγουν καλύτερα αποτελέσματα από το CAiRE μοντέλο και τα state-of-the-art.

- Όσον αφορά τις προσεγγίσεις μας που βασίζονται στο μοντέλο T5, παρατηρούμε ότι όλες παράγουν state-of-the-art αποτελέσματα όσον αφορά την μετρική BLEU. Πιο συγκεκριμένα, το μοντέλο T5 (P+ED) βελτιώνει την state-of-the-art μετρική BLEU κατά 19.5%, ενώ παράλληλα προσεγγίζει την state-of-the-art μετρική Perplexity (την οποία πετυχαίνει το μοντέλο DodecaDialogue MT+FT) με διαφορά 9.7%.
- Τέλος, αξίζει να επισημάνουμε ότι τα T5-multitask1 και T5-multitask2 φαίνεται να μην βελτιώνουν την ποιότητα των απαντήσεων, σύμφωνα με τις αναφερόμενες μετρικές.

0.4 Συνεισφορές & Μελλοντικές Προεκτάσεις

Σε αυτήν τη διπλωματική εργασία, μελετήσαμε σε βάθος τη δουλειά που έχει γίνει στον τομέα της δημιουργίας διαλογικών συστημάτων με ενσυναίσθηση, χρησιμοποιώντας generation-based μοντέλα. Επιπλέον, προτείναμε δικές μας ιδέες για περαιτέρω βελτίωση αυτών των συστημάτων. Πιο συγκεκριμένα, αρχικά, αναλύσαμε τις παραδοσιακές αρχιτεκτονικές (vanilla seq2seq, vanilla seq2seq με attention) που χρησιμοποιούνται για την παραγωγή διαλόγου, και στην συνέχεια μελετήσαμε τα state-of-the-art μοντέλα (Transformer Encoder Decoder, BERT, GPT2 και T5), παρέχοντας λοιπόν στον αναγνώστη το κατάλληλο θεωρητικό υπόβαθρο. Στην συνέχεια επικεντρωθήκαμε στο πρόβλημα της παραγωγής διαλόγων με ενσυναίσθηση, ένα πρόβλημα που εισήγαγε το Facebook για τη δημιουργία διαλογικών συστημάτων ενσυναίσθησης. Στα πλαίσια αυτής της εργασίας μελετήσαμε τα μοντέλα που έχουν προταθεί από άλλους ερευνητές και στην συνέχεια προτείναμε δικές μας προσεγγίσεις. Τελικά, οι προτεινόμενες προσεγγίσεις μας, που βασίζονται στο μοντέλο T5, βελτιώνουν τα state-of-the-art αποτελέσματα με βάση την μετρική αξιολόγησης BLEU, επιτυγχάνοντας παράλληλα κοντινά αποτελέσματα με αυτά των state-of-the-art μοντέλων με βάση την μετρική perplexity.

Τέλος, προκειμένου να επεκτήνουμε και να βελτιώσουμε την έρευνα που έγινε σε αυτή την διπλωματική εργασία προτείνουμε κάποιες μελλοντικές προεκτάσεις. Αυτές εν συντομία είναι:

- Προτείνουμε να εκπαιδεύσουμε τα T5-based μοντέλα σε διαφορετικά tasks διαλόγου με χρήση εκμάθησης πολλαπλών εργασιών (multi-task learning), προκειμένου να αποκτήσουν περισσότερες δεξιότητες.
- Προτείνουμε επίσης την χρήση μετρικών ομοιότητας μεταξύ των δύο διανυσματικών αναπαραστάσεων που εμπεριέχουν την συναισθηματική πληροφορία (στο μοντέλο T5-multitask2). Με αυτήν την προσέγγιση μπορούμε να μοντελοποιήσουμε καλύτερα την ενσυναίσθηση.
- Επίσης η ενίσχυση αυτής της εργασίας με human evaluation μετρικές θα βοηθούσε πολύ ώστε να κρίνουμε τα αποτελέσματα πιο αντικειμενικά.
- Και τέλος, προτείνουμε να μελετηθούν περισσότερο τα μοντέλα σε χαμηλότερο επίπεδο, παρατηρώντας ποια μέρη αυτών δουλεύουν καλά και ποιά όχι (ablation study).

Chapter 1

Introduction

1.1 Motivation

Language is considered to be one of the hallmarks of human civilization. The ability to communicate has led to the development of a remarkable culture that may not exist again. Since dialogue is regarded as a fundamental element of human civilization, developing systems that are capable of understanding human language and communicating with humans is one of the most challenging tasks towards Artificial Intelligence (AI). This has become an active area of research in the domains of AI and Natural Language Processing (NLP), in particular.

With the advances of Machine Learning (ML), and particularly Deep Learning (DL), various dialog systems have been developed not only able to help humans complete tasks in several domains such as flight booking, restaurant reservations, online shopping, and customer care but also to entertain them. However, the complexity of human language makes the task of dialogue generation really challenging.

Among the various approaches in building conversational agents able to entertain humans, open domain generation-based chatbots is a significant field of research, as those agents seem to be the most human-like being able to discuss almost everything. Thankfully to the advance of Deep Learning various generation-based conversational agents are implemented being able to communicate with humans, producing syntactically and grammatically coherent responses. However, beyond understanding what is being discussed, human communication requires awareness of how someone is feeling. While it is straightforward for humans to recognize and acknowledge others' feelings in a conversation, this is a significant challenge for AI systems. Humans feel and express different types of emotions depending on the situation of the conversation. Natural communication is frequently prompted by people sharing their feelings or circumstances. We should also consider that emotions also play an important role during a conversation, as they help the conversational partner to better understand the situation of the speaker, developing a confidential relationship between the speaker and the listener. ELIZA [28], one of the first chatbots developed, focused most of its attention on asking its conversational partners why they were feeling a certain way. Without going any further, listening to somebody saying "I was fired from work yesterday." and giving a response like "Why were you fired?" is contextually relevant but is not enough, as the response is emotionally empty. However, providing a response like "Oh! I am so sorry to hear that. Why were you fired?", is more human-like and relevant to the emotional state of the user. Consequently, one of the most significant challenges for a dialogue agent is to appropriately respond to a conversation partner that is describing personal experiences, by understanding and acknowledging any implied feelings — a skill we refer to as empathetic responding.

Following this perspective, Facebook introduced the Empathetic Dialogues task [25], a task aiming to create dialog systems that are able to react in an empathetic way. To this end, we aim to contribute towards making a conversational agent able to appropriately respond to a conversation partner, by understanding the implied feelings and responding in an empathetic engaging way. In our research, we experiment with state-of-the-art deep learning generation-based models using the dataset introduced in Empathetic Dialogues task, and we present our approaches to further improve upon empathetic

dialog systems.

1.2 Dialogue Systems

The development of natural language is one of the most important characteristics of human civilization. Humans developed natural language in order to be able to coexist, communicate, and evolve as a social human being. From the moment of birth until the end of our life, we use natural language to communicate. So, the dialog is rightly considered to be the most fundamental element of language, as it is a significant part of our daily life. All of us use this kind of language, whether we are talking with our families, or hanging out with friends, or even participating in business meetings.

The rapid evolution of technology, promoted the creation of systems that are able to communicate with humans. Human-computer interaction (HCI) is a technology that allows communication between users and computers using natural language. Automated conversational agents are systems (machines) that have been designed to communicate with humans, imitating human behavior during the conversation. Those systems are widely applied in several domains such as many service industries to help schedule meetings, make restaurant reservations, shop online, and support customer care, and so on. However, conversational agents are not only used to offer services but to entertain humans too.

Conversational agents fall into two general categories, task-oriented and non-task-oriented agents. Task-oriented dialog agents also known as goal-based agents, are designed for particular tasks and set up to have short conversations with the user. The main goal is to help the user to complete a specific task, by getting information from the user and producing relevant responses. Various services that we use in our daily life, such as booking, traveling, shopping, or even ordering food applications are supported by those conversational agents. Without going any further, both domestic controllers and cellphone assistants (Siri, Alexa, Google Now/Home, etc.) make use of dialog systems in order to communicate with the user and complete the asked tasks, such as making restaurant reservations, giving traveling directions and making phone calls [38]. The key to the success of those agents is to help the user to complete the required task as soon as possible. So, the main focus is given to task completion, having short conversations, reducing the time of conversation to the minimum.

On the other hand, non-task oriented agents are designed for extended conversations and are set up to mimic the unstructured conversation of human-human interaction, rather than focusing on task completion. Research on non-goal-driven dialogue systems goes back to the mid-60s. It began, perhaps, with Weizenbaum's famous program ELIZA, a system based only on simple text parsing rules that managed to convincingly mimic a Rogerian psychotherapist by persistently rephrasing statements or asking questions [28]. A few years later the PARRY chatbot [29] was created, using simple text parsing rules. PARRY managed to mimic the pathological behavior of a paranoid patient to the extent that clinicians could not distinguish it from real patients.However, afterwards various systems have been developed not focusing on practical purposes such as the ELIZA agent, but having an entertainment value such as Microsoft's XiaoIce system [30].

Non-task-oriented dialog systems (chatbots) can further be separated into two categories, according to their architecture: rule-based and corpus-based chatbots. Rule-based chatbots are based on pattern-transform rules that are used to encode the input sentence and to produce a response. Rule-based chatbots use keywords that are associated with a rank, with specific words being more highly ranked, and more general words being low ranked. Given the input sentence, rule-based chatbots find the word with the highest keyword rank in the input sentence and then choose the highest-ranked rule that best matches the input sentence. Afterwards a transform rule is applied to the pattern rule to produce a response. The very first rule-based chatbot that was implemented was the ELIZA [28]. Nowadays,

some modern chatbots also follow the rule-based architecture such as the ALICE chatbot [31], which uses an updated version of ELIZA's architecture.

On the other hand, corpus-based chatbots, instead of using hand-built rules, mine human-human conversations to produce a response. Those chatbots are further divided into two categories: retrieval-based and generation-based chatbots. Systems that belong in the first category select deterministically from a fixed set of possible responses. More specifically, those chatbots map the dialogue history and external knowledge (e.g. a database, which can be queried by the system) to a response action. Systems that search through a database of dialogues and pick responses with the most similar context, belong to this category [32, 33]. In contrast to retrieval systems, generation-based systems attempt to generate responses by keeping a posterior distribution over possible responses. Those systems generate responses word-by-word, by sampling words from the probability distribution over the vocabulary used. Generation-based systems usually combine external knowledge from databases to produce more informative responses, however, in this diploma thesis we study generation-based systems without the use of external knowledge. One of the most famous generation-based open-domain chatbots is Meena [34] proposed by Google, which was trained on open-domain conversations from social media. Recently, FacebookAI also built the BlenderBot [35], the largest-ever open-domain chatbot. This chatbot combines both retrieval and generation based architectures and has 9.4 billion parameters totally. It is a human-like chatbot with multiple conversational skills including empathy, personality and knowledge, and outperforms other chatbots in terms of engagement level according to human evaluations.

1.3 Language Models

In the field of natural language processing (NLP), deep neural networks have improved models' performance in many tasks. However, training a model from scratch requires a plethora of labeled data. For instance, training a dialog system requires millions of data, with specific answers as labels (targets). However, in many practical applications, there are a few data available with tags to do supervised training of the model. In these cases, the use of transfer learning (described in Section 2.5) offers an alternative solution. The models are pre-trained in a similar task, and then are fine-tuned in the required task, by utilizing the knowledge acquired from the pre-training process. So, in order to achieve satisfactory results when building a conversational agent, we pre-train a model as a language model. In this way, the model used is able to create high-level representations of the language, utilizing this knowledge afterwards during fine-tuning.

In language modeling, the model calculates a probabilistic distribution over a sequence of words, creating a unique representation for each word occurred, based on a previous context. In other words, a language model is called to calculate the likelihood of occurrence of a number of words in a particular sequence. More specifically, a language model given a sequence of words as input tries to predict the next word. In this way, the model is capable of creating word representations based on context. As a result, it does not require training data with labels, which are hard to find. Plain text, however, is available in large quantities for each possible task. So, language models can be trained in a plethora of data that are available for free. Due to the ability of language models to export general word representations based on content, are now widely used in many domains. Such a domain is dialog systems. We further analyze the use of language models in Section 3.4.

1.4 Emotion Recognition

As already mentioned, a challenging task in building dialog systems is to understand the feelings of the user, as apart from understanding what is being discussed it is crucial to acknowledge how someone is feeling. Emotions play an important role during a conversation, as they help the conversational partner to better understand the situation of the speaker, developing a confidential relationship between

the speaker and the listener. From this perspective, is considered crucial to briefly study emotion recognition.

In NLP, emotion recognition is the process of recognising distinct emotions that are implied in the written word. The analysis of the implied feelings of the user can be considered as an extension of the sentiment analysis task. Sentiment analysis in its simplest form aims to detect positive, negative, or neutral feelings from the text, while emotion detection and recognition aim to detect and recognize types of feelings through the expression of texts, such as anger, disgust, fear, happiness, sadness, and surprise.

Thousands of articles have been written on the methods and applications of automated emotion recognition. Therefore, this is a significant field of NLP, which seems to be extremely useful, in many domains such as marketing, advertising [39, 40], automated question-answering [41], and especially in dialog systems [42, 43, 44, 45, 46]. We further study emotion recognition in Section 3.5.

1.5 Goals & Contributions

In this diploma thesis, we aim to contribute to the direction of research methods that make a conversational agent able to appropriately respond to a conversation partner, by understanding the implied feelings and responding in an empathetic engaging way. As we already mentioned, except that natural communication is frequently prompted by people sharing their feelings or circumstances, understanding any implied emotions during a conversation is beneficial as the conversational partner embraces the speaker's feelings and focuses his interest on the dialog. In this way, he is able to develop a more confidential relationship with the speaker. From this perspective, we consider crucial to enhance conversational agents with empathy (and generally with emotions) in order to be more human-like.

Empathetic Dialogues [25] is a task proposed by Facebook, which aims to create dialog systems that are able to react in an empathetic way. Empathetic responding not only benefits open-domain chatbots but also task-oriented conversational agents, as many researchers have noticed that reacting in an empathetic way or generally displaying a caring attitude, is associated with better results in many tasks [47, 48, 49]. Following this work, we aim to contribute to the direction of research methods that make a conversational agent able to respond in an empathetic way.

In this research, after studying in depth the work done in the field of creating empathetic conversational agents, we experiment with state-of-the-art deep learning generation-based models for empathetic dialogue generation and propose our new ways to further improve upon these systems. Focusing on the Empathetic Dialogues task, we experiment with different approaches to improve the state-of-the-art results. More specifically, we experiment with BERT2BERT and BERT2GPT2 models achieving comparable results with those of the baselines. We also present our approaches based on the T5 model, improving the state-of-the-art results concerning the BLEU score by 19.5%, while their performance as far as perplexity is concerned, is close to the current state-of-the-art model, having a difference of 9.7%.

1.6 Thesis Organisation

This diploma thesis is structured as follows:

In Chapter 2 we provide technical background knowledge in machine learning and deep learning. More specifically, at first, we introduce the reader to the basic concepts of machine learning and we study some conventional machine learning models used in classification tasks. Afterwards, we introduce the basic concepts of deep learning and present the most common recurrent neural networks

(RNNs, LSTMs and GRUs), along with an introduction to attention mechanisms. Finally, the concepts of transfer and multi-task learning are introduced, also presenting the motivation of their use.

In Chapter 3, the natural language processing background needed for this thesis is presented. After briefly presenting popular natural language processing tasks and applications, we analyze the most common language representation methods (frequency-based, Word2Vec, Glove and contextualized embeddings). Then, we present language modeling with the use of n-gram models and neural networks. Finally, a brief description of emotion recognition using NLP is given.

In Chapter 4, we provide a theoretical background for understanding in depth dialogue generation using generation-based models. At first we analyse some basic and state-of-the-art models in depth, and then we focus on the decoding methods used for generating responses and on the most common evaluation metrics used for evaluating dialog systems.

In Chapter 5, we study in depth dialogue generation with empathy. After a brief introduction to the task, we present the recent work done and the datasets we used on the experiments we conducted. We analyse in depth baseline architectures proposed in recent studies and then we introduce proposed architectures to be applied to the task. Finally, we train the proposed models on the aforementioned datasets and present the corresponding results.

Finally, Chapter 6 presents our conclusions where we summarize our findings and provide an outlook into the future.

Chapter 2

Technical Background

In this Chapter we provide the reader with technical background knowledge in machine learning and deep learning. In Section 2.1 we present the basic methods of machine learning and in Section 2.2 we introduce traditional machine learning models used in classification tasks. In Section 2.3 an introduction in deep learning is done, presenting artificial neural networks (ANNs), activation functions, the backpropagation method and optimization strategies. Afterwards, in Section 2.4 we study recurrent neural networks and attention mechanisms. Finally, in Sections 2.5, 2.6 brief descriptions of transfer learning and multi-task learning methods are given accordingly.

2.1 Introduction to Machine Learning

In the 1950s, we saw the first computer game program claiming to be able to beat the checkers' world champion. Around the same time, in 1957, Frank Rosenblatt invented the "Perceptron" which was a very simple linear classifier [50]. At that time, it was a real breakthrough. Then, we see several years of stagnation of the neural network field due to its difficulties in solving certain problems. In the 1990s, machine learning methods started to gain in popularity thanks to the ability to extract interesting results by learning from a plethora of data.
Machine learning (ML) is a sub-field of artificial intelligence (AI). According to Arthur Samuel, machine learning enables computers to learn from data without being explicitly programmed [51]. The goal of machine learning generally is to build models that are able to understand unstructured data and make predictions or decisions. The primary aim is to allow the computers to learn this process automatically, without human intervention or assistance and adjust actions accordingly. We should mention here that machine learning differs from traditional computational approaches, as in traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or solve a problem, while in machine learning computers are not explicitly programmed.
Nowadays, most technologies have benefited from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology which converts images to text, is also using machine learning. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars also rely on machine learning for navigation. Machine learning algorithms are also used in applications such as email filtering and detection of network intruders, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory, and application domains to the field of machine learning. Data mining is a field of study within machine learning and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.
In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Three of the most widely adopted machine learning methods are:

- supervised learning which provides the algorithm with labeled data in order to be trained using

the input and the labeled data

- unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within the input data
- semi-supervised learning which provides the algorithm with mixed labeled and not labeled data

Let's explore those methods in more detail.

2.1.1 Supervised Learning

In supervised learning, an AI system is provided with samples that are labeled, which means that each sample is tagged with a correct label. In other words, there are input variables X and an output variable Y . The goal is to build an algorithm that learns the mapping function f from the input to the output [52].

$$Y = f(X) \tag{2.1}$$

So, the primary aim is to approximate the mapping function so well that when we have a new input sample, we can predict the output variables for that sample. The process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. Known the correct answers (entitled labels), the algorithm iteratively makes predictions on the training data and is corrected by the teacher according to a calculated error. Learning stops when the algorithm achieves an acceptable level of performance.
Tasks in supervised learning are distinguished into classification tasks and regression tasks. Regression is the procedure of mapping an input sample to a continuous output value, such as an integer or a floating-point value. On the other hand, classification is the task of identifying to which of a set of categories or classes an unseen observation belongs, given a training procedure of input samples that belong to the set of these classes. The simplest case of a classification task is to have two possible categories, which is known as binary classification.
An example of binary classification is shown in Figure 2.1 for filtering emails as "Spam" or "Not spam".

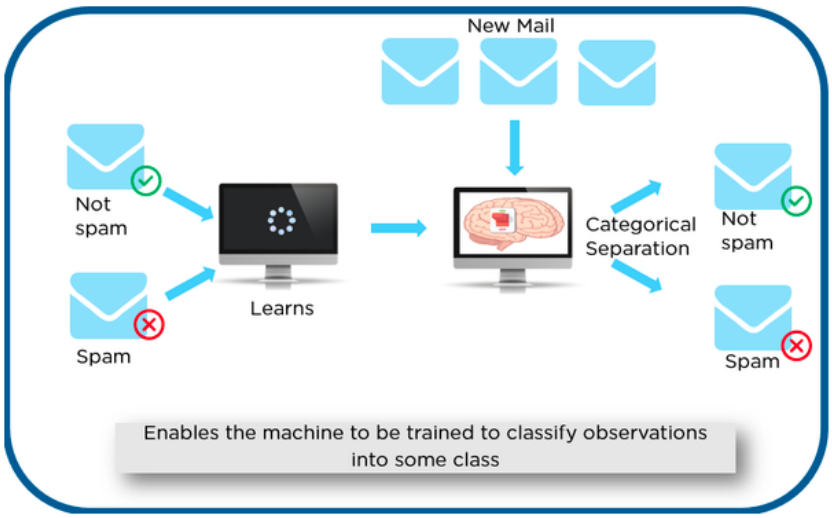


Figure 2.1: Example of binary classification. Image source: [6]

We initially take some data and mark them as "Spam" or "Not Spam". Those labeled data are used to train the model. Once the model is trained, we can test it by feeding it with a new email and check if the model is able to predict the right output.

2.1.2 Unsupervised Learning

In unsupervised learning, data are unlabeled. In other words, the training set consists of a set of input vectors X without any corresponding target values. So the learning algorithm is left to find commonalities among its input data and find patterns where the target values are either not observable or infeasible to obtain. [53]
As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable. The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.
Clustering is a large subclass of unsupervised tasks. The goal is to group observations together in such a way that members of a common group are similar to each other and different from members of other groups. However, it is often difficult to know how many clusters should exist or how they should look. Association is also a common unsupervised task. An association rule learning problem is where we want to discover rules that describe large portions of our data, such as people that buy X also tend to buy Y.

2.1.3 Semi-supervised Learning

Semi-supervised learning algorithms represent a middle ground between supervised and unsupervised algorithms. These algorithms operate on data that have a few labels but are mostly unlabeled. [54]. Traditionally, one would either choose the supervised route and operate only on the data with labels, vastly reducing the size of the dataset, otherwise, one would choose the unsupervised route and discard the labels while keeping the rest of the dataset for something like clustering. On the contrary, a semi-supervised machine-learning algorithm would partially train a model using a small set of labeled data. Then the partially trained model would label the unlabeled data (creation of "pseudo-labeled" data). Finally, meging the labeled and pseudo-labeled data, a semi-supervised approach would combine both the descriptive and predictive aspects of supervised and unsupervised learning.

2.2 Traditional Machine Learning Models

As already mentioned in Section 2.1.1, and according to the equation 2.1 the primary aim of each supervised learning algorithm is to learn the mapping function f from the input X to the output Y. The more accurate the mapping is, the better the learning. The question is what kind of function we can use. In the simplest case, the function f is assumed to be a linear function of the following form:

$$f(x) = xW + b \tag{2.2}$$

where $x \in \mathbb{R}^{d_{in}}$,$W \in \mathbb{R}^{d_{in} \times d_{out}}$ and $b \in \mathbb{R}^{d_{out}}$. So, assuming that the function f is a linear function our goal now is to find the best set of parameters $\theta = \{W, b\}$ in order to have an accurate mapping.

2.2.1 Loss Function

Let's assume that the input x is fed to the model. The output of the model is the prediction made, denoted with $\hat{\mathbf{y}}$. In order to quantify the error between the predicted output $\hat{\mathbf{y}}$ and the true label $\mathbf{y}$, the notion of the loss function is introduced. Formally, the loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ assigns a numerical score (a scalar) to a predicted output $\hat{\mathbf{y}}$ given the true output $\mathbf{y}$. The less the numerical score is, the better the prediction made. So, the parameters of the learned function are determined by minimizing the loss $\mathcal{L}$ over the training examples.

Given a labeled training set $(\mathbf{x}_{1:N}, \mathbf{y}_{1:N})$, the per-instance loss function L and a parameterized function $f(\mathbf{x}, \theta)$, we define the total loss over the training set with respect to the parameters θ as follows:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} L\left(f\left(\mathbf{x}_i; \theta\right), \mathbf{y_i}\right) \tag{2.3}$$

We notice that the total loss is the average sum of the losses over all training examples. The goal is now to determine the optimal parameters $\hat{\theta}$ that minimize the total loss $\mathcal{L}$:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} L\left(f\left(\mathbf{x}_i; \theta\right), \mathbf{y}_i\right) \tag{2.4}$$

Now we will define the notion of entropy and then we will study the most common loss functions used in classification tasks.

Suppose we want to communicate a set of n events from a particular probability distribution p of dataset X. Information entropy is the minimum average encoding size of information in dataset X to communicate the events [55]. More formally it is described as follows:

$$H(p) = \sum_{x} p(x) \log \frac{1}{p(x)} \tag{2.5}$$

We might also recall that information quantifies the number of bits required to encode and transmit an event. Lower probability events have more information, higher probability events have less information. If entropy is high, we have a big amount of information and also many events with low probabilities. So, entropy apart from the amount of information can also be seen as a measure of uncertainty.

Cross-entropy is defined as the minimum average of encoding size of communicating an event from one distribution to another. More formally it is described as:

$$H_p(q) = \sum_{x} q(x) \log\left(\frac{1}{p(x)}\right) \tag{2.6}$$

Binary entropy is the case with only two categories. According to Bernoulli process with probability p, if we let X be a random variable that can take the values 0 and 1, the binary-entropy is defined as:

$$H(X) = \text{plog}\frac{1}{p} + (1-p)\log\frac{1}{1-p} = -p\log p - (1-p)\log(1-p) \tag{2.7}$$

Binary cross-entropy loss: Binary cross-entropy loss [56] is used in binary classification. Let's assume two target classes labeled with the values 0 and 1. We use the sigmoid activation function (described in equation 2.22)on the classifier in order keep the output prediction $\hat{y}$ in the range $[0, 1]$. The output $\hat{y}$ is interpreted as the conditional probability $\hat{y} = P(y = 1|x)$. Thus, the rule used to determine if a new sample is classified to the first or to the second class is the following:

$$\text{prediction } = \begin{cases} 0, & \text{if } \hat{y} < 0.5 \\ 1, & \text{if } \hat{y} \geq 0.5 \end{cases} \tag{2.8}$$

So, the model is trained to maximize the log conditional probability $P(y = 1|x)$ or equally to minimize the binary cross-entropy loss:

$$L_{\text{binary}}\left(\hat{y}, y\right) = -y\log\hat{y} - (1-y)\log(1-\hat{y}) \tag{2.9}$$

Negative log-likelihood loss: Negative log-likelihood loss [56] is also known as categorical cross-entropy loss. It can be used for multi-class classification problems (when a probabilistic interpretation of scores is desired). Let $\mathbf{y} = \mathbf{y}_1, ..., \mathbf{y}_n$ be a vector representing the true multinomial distribution over the labels $\mathbf{1}, ..., \mathbf{n}$, and let $\hat{\mathbf{y}} = \hat{\mathbf{y}_1}, ..., \hat{\mathbf{y}_n}$ be the linear classifier's output, which was transformed by the softmax function and represents the class membership conditional distribution $\hat{\mathbf{y}}_i = P(y = 1 \mid \mathbf{x})$). The categorical cross-entropy loss for the i-th sample is described as:

$$L\left(\hat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i}\right) = -\mathbf{y}_i \log\left(\hat{\mathbf{y}}_i\right) \tag{2.10}$$

In order to find the optimal parameters for the model, we want to maximize the likelihood or equivalently to minimize the average negative log-likelihood over all the training samples. So, the objective function takes the following form:

$$\mathcal{L}_{\text{cross-entropy}}\left(\hat{\mathbf{y}}, \mathbf{y}\right) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_{[i]} \log\left(\hat{\mathbf{y}}_{[i]}\right) \tag{2.11}$$

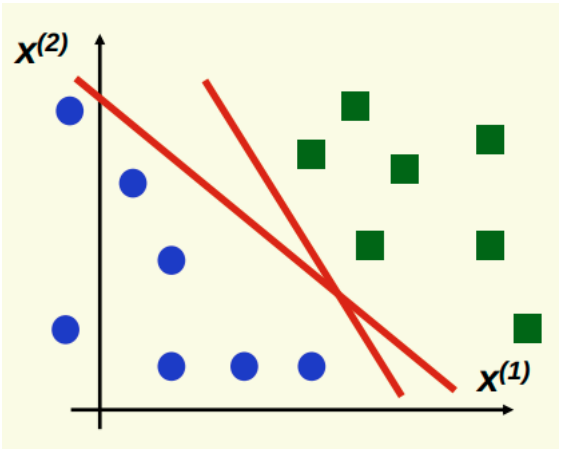
We should mention here that negative log-likelihood loss is significant for training the neural networks and especially in our case for training generative models. The value of the loss allows computing the error of the neural network and with the use of a parameter optimizer, the gradient $\nabla_\theta L(\hat{\mathbf{y}}, \mathbf{y})$ is computed to find a local minimum. Then the parameters (weights) of the model are optimized using the backpropagation algorithm [57]. We should note that the loss is not computed on each iteration over the whole dataset, but over a mini subset of samples called a batch.

2.2.2 Support Vector Machines (SVMs)

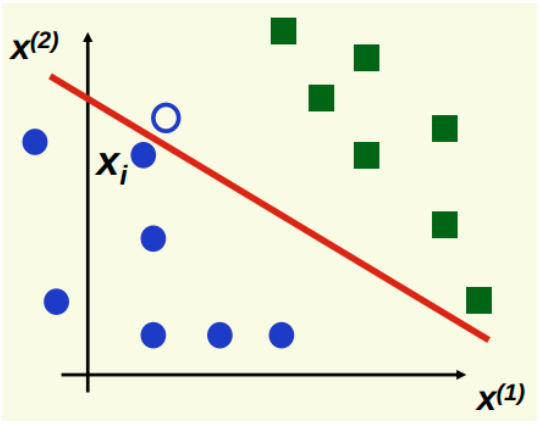
[58] Consider we have a two-class classification problem, where the two classes are linearly separable and we want to find a separating hyperplane. More formally, suppose that the training dataset comprises of N input vectors $x_1, x_2, ..., x_N$, where $x_i \in \mathbb{R}^d$ and the corresponding labels $y_1, y_2, ..., y_N$, where $y_i \in \{-1, 1\}$. Assuming that those N input samples are linearly separable we want to find a separating hyperplane of the following form:

$$f(x) = w^T x + b \tag{2.12}$$

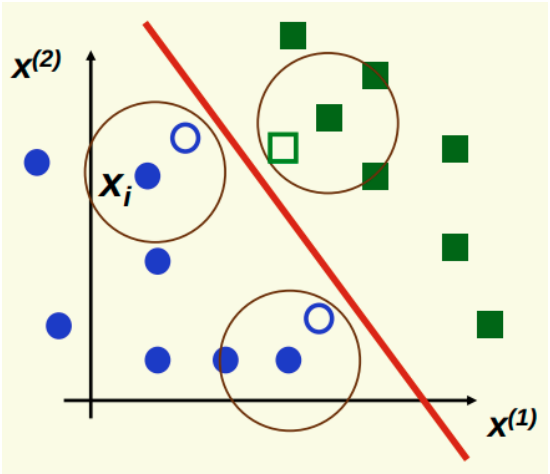
where $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$ and $b \in \mathbb{R}^d$. By definition there exists at least one choice of the parameters w and b such that a function of the form 2.12 satisfies $f(x_i) > 0$ for points having $y_i = 1$ and $f(x_i) < 0$ for points having $y_i = -1$, so that $y_i \cdot f(x_i) > 0$ for all training points.
As shown in Figure 2.2a we can notice that there are infinite possible hyperplanes that linearly separate the two classes. The goal of the SVM algorithm is to choose the optimal hyperplane, which is the hyperplane that has better performance on classifying unseen samples. By choosing a hyperplane that is close to a sample x_i, if we see a new sample close to x_i it is likely to be misclassified, having poor generalization as shown in Figure 2.2b. On the other hand, by choosing hyperplane as far as possible from any sample, new samples close to the old samples are likely to be classified correctly, having a good generalization as shown in Figure 2.2c. So, in order to have good generalization, we must find the hyperplane for which the minimum distance between the two classes (margin) is as wide as possible [59]. In other words, by maximizing the margin we can find the optimal hyperplane as shown in Figure 2.2d.
The samples that are closest to the separating hyperplane are called support vectors and completely define the optimal hyperplane (of course, we do not know which samples are support vectors without finding the optimal hyperplane).



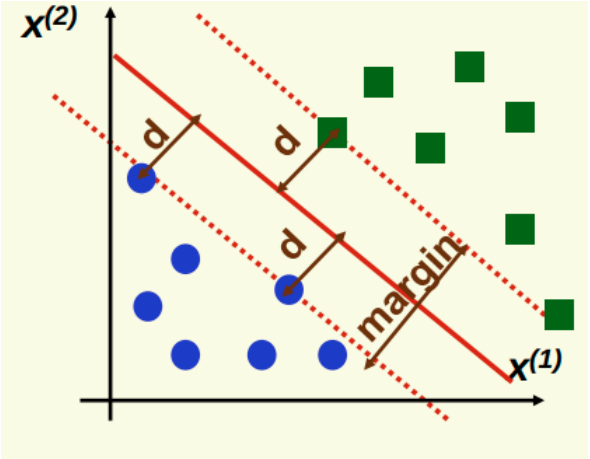
(a) Possible separation hyperplanes



(b) Choosing a hyperplane close to x_i (poor generalization).



(c) Choosing a hyperplane as far as possible from any sample (good generalization).



(d) Maximizing the margin and choosing the optimal hyperplane.

Figure 2.2: Different hyperplane choices. Image source: [59]

If $f(x)$ separates the data, the geometric distance between a point x_i and a hyperplane $f(x) = 0$ is:

$$z = \frac{|f(x_i)|}{\|w\|} \tag{2.13}$$

Furthermore, we are only interested in solutions for which all data points are correctly classified, so that $y_i \cdot f(x_i) > 0$ for all i. Then, the distance between a point x_i and the optimal hyperplane is given by:

$$\frac{y_i f\left(x_i\right)}{\|w\|} = \frac{y_i\left(w^T x_i + b\right)}{\|w\|} \tag{2.14}$$

We can set $f(x) = +1$ and $f(x) = -1$ for the closest points of each class being on the boundaries, in order to maximize the margin. Considering now that x_i is an example closest to the boundary, its distance from the optimal hyperplane becomes:

$$\frac{y_i f\left(x_i\right)}{\|w\|} = \frac{1}{\|w\|} \tag{2.15}$$

So now the problem is to maximize the margin:

$$m = \frac{2}{\|w\|} \tag{2.16}$$

with the following constraints:

$$w^T x_i + b \geq 1, \text{ for each } y_i = +1 \tag{2.17}$$

$$w^T x_i + b \leq -1, \text{ for each } y_i = -1 \tag{2.18}$$

The above problem is reducible to determining the parameters w and b, for minimizing $\frac{1}{2}\|w\|^2$. So our problem now becomes:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad &y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, ..., N \end{aligned} \tag{2.19}$$

We should note here that $\frac{1}{2}\|w\|^2$ is a quadratic function, thus there is a global minimum. The problem of equation 2.19 is solved using Lagrange multipliers.

2.2.3 Logistic Regression

Logistic Regression (LR) is used to solve linear classification problems. The difference between other simple classifiers is that it calculates the probability of an input sample $x \in \mathbb{R}^d$ belonging to a class, by giving as output a value in the range [0,1]. [52, 60, 61]
Suppose that we study a binary classification problem with classes $y = \{0, 1\}$. The activation of the LR classifier is determined by applying a sigmoid function (described in equation 2.22) over the fitted line in order to get the final classification decision.

So the activation of LR for a given input vector x becomes:

$$P(y = 1|x) = h_w(x) = \sigma\left(w^T x\right) = \frac{1}{1 + e^{-w^T x}} \tag{2.20}$$

If the output is greater than 0.5 then the sample x is classified to the first class with a probability of $P(y = 1|x)$, otherwise to the second with a probability of $P(y = 0|x) = 1 - P(y = 1|x)$.

Supposing that we have a training set and we want to train a LR classifier in order to do classification between those two classes, we train the model using the binary cross entropy loss mentioned in Section 2.2.1, which takes the following form:

$$J(w) = -[y \log(P(y = 1 \mid x)) + (1 - y) \log(1 - P(y = 1 \mid x))] \tag{2.21}$$

2.3 Introduction to Deep Learning

Deep learning is essentially a branch of machine learning which tries to model data using complex architectures with non-linear transformations. [62, 63] Two significant differences between deep learning and machine learning methods are the size of the models [64] used and the part of feature extraction from data. In traditional machine learning, the algorithm is given a set of relevant features to analyze. However, in deep learning, the algorithm decides for itself what features are relevant, by processing a plethora of data. The elementary bricks of deep learning are the neural networks, which are combined to form large ("deep") models. Deep learning techniques have enabled significant progress in the field of computer vision, natural language processing, and speech recognition as they are used in a wide variety of applications seen nowadays such as self-driving cars, voice search, and voice-activated assistants, automatic machine translation systems, automatic brain cancer detection systems, etc.

2.3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models that are based on the abstract representation of human neurons. The goal of modeling biological neural systems gave a rise to the area of ANNs. So, by imitating neurons, the basic computational unit of the human nervous system, the deep learning community achieved significant results in many tasks. For that reason, a brief high-level description of the biological neuron human system is given in the following part.

The basic computational unit of the brain is the neuron [65]. Our nervous system consists of billions of neurons. The analogy between the biological neuron and its mathematical model is illustrated in Figure 2.3.

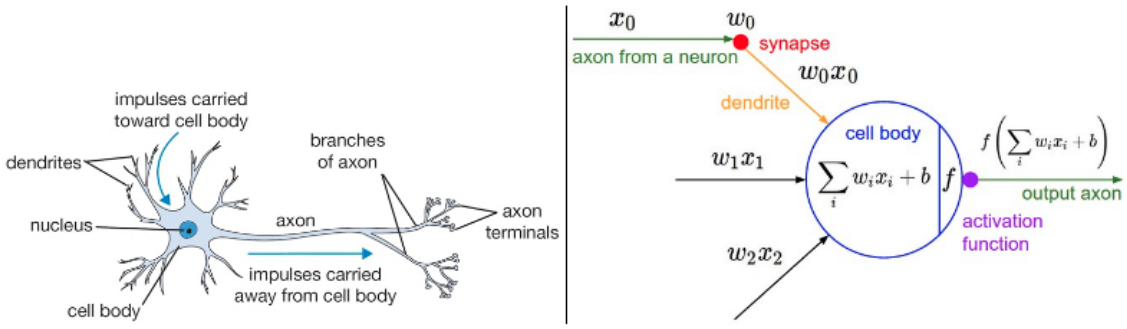


Figure 2.3: An illustration of a biological neuron (left) and its mathematical notation (right). Image Source: [7]

A typical biological neuron consists of a cell body (soma), dendrites, and a single axon. The soma is usually compact. The axon and dendrites are filaments that extrude from it. Dendrites typically branch profusely and extend a few hundred micrometers from the soma. [66] The axon leaves the soma at a swelling called the axon hillock, and travels for as far as 1 meter in humans. It branches but usually maintains a constant diameter. At the farthest tip of the axon's branches are axon terminals, where the neuron can transmit a signal across the synapse to another cell [67]. Each neuron receives input signals via the dendrites and the impulses are carried across the cell body. Then, output signals are produced along the single axon of the neuron and they are transmitted to other neurons across the synapses. Similarly, in the computational model of a neuron, the axons of the neurons are connected via synapses with other neurons. Then, each of the dendrites carries the signal to the cell body of the neuron. More specifically, the input signals that travel along the axons, denoted with x_i, interact multiplicatively with the synaptic strength of the synapse, denoted with w_i and pass through the dendrites to the cell body. The synaptic strengths (w_i) are learnable weights and control the strength of the influence of one neuron on another. Then, the signals carried from the dendrites to the cell body, denoted with $w_i x_i$, are summed and if the final sum is above a certain threshold the neuron can fire, sending a spike along its axon. The information transferred through the axon is highly related to the firing rate. Thus in the computational model, we model the firing rate with the use of an activation function, denoted with f. Some of the most common activation functions used are the sigmoid, the hyperbolic tangent and the rectified linear unit functions, which are described in Section 2.3.2.

In order to model complex data structures and create models that are able to learn non-linear functions and perform well on machine learning tasks, architectures that combine artificial neurons are designed. The simplest case of such architectures is the multi-layer perceptron (MLP), which consists of three layers (in its simplest form): the input layer, the hidden layer and the output layer. Each of the nodes of the hidden and output layer is a neuron that uses a non-linear activation function, exactly as we described it before. Multi-layer perceptrons differ from linear perceptrons as they can separate data that are non-linearly separable. A multi-layer perceptron that consists of more than one hidden

layers, is considered to be a deep neural network, while a multi-layer perceptron with one hidden layer is considered to be a simple neural network. The input layer of a deep neural network gets the input data and forwards them to the first hidden layer without any computation. Then the first hidden layer processes the data and creates a representation of them. The output of the first hidden layer is forwarded to the next hidden layers, constructing higher-level representations while moving to "higher" hidden layers. Finally, the last hidden layer forwards the final representation to the output layer, which makes a decision depending on the task. A comparison between a simple and a deep neural network is shown in Figure 2.4.

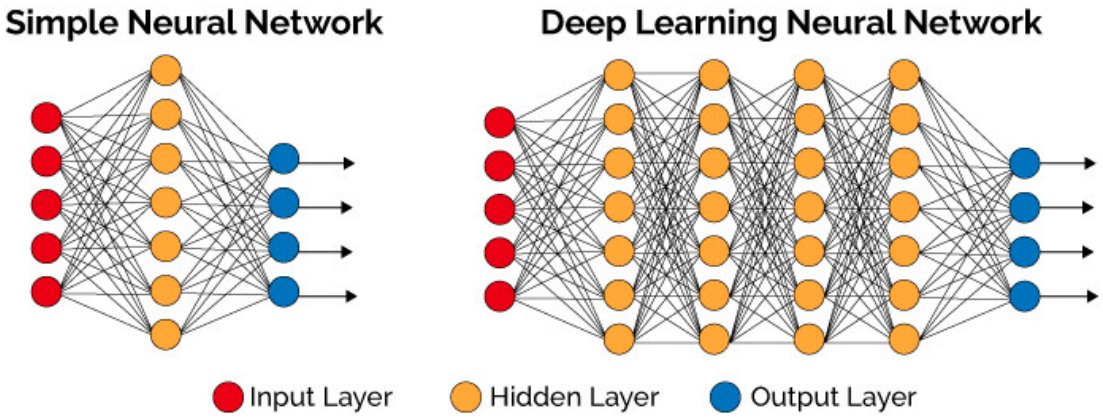


Figure 2.4: An illustration of a simple (left) and a deep (right) neural network. Image Source: [8]

2.3.2 Activation Functions

In this subsection we study the most commonly used non-linear activation functions and we analyse their advantages and disadvantages. The use of non-linear activation functions is significant in order to introduce non-linear real-world properties to artificial neural networks. If the activation function is not applied, the output signal becomes a simple linear function. Linear functions are only single-grade polynomials. A non-activated neural network will act as a linear regression with limited learning power. So, the use of non-linear activation functions is fundamental as we want our models to learn non-linear states and be able to distinguish non-linearly separable data.

Sigmoid function:
The sigmoid non-linearity is described by the following mathematical form:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.22}$$

The sigmoid function takes a real-valued number and outputs a real number in range $[0, 1]$. The sigmoid function is commonly used for classifiers as small changes in x are large in y, making the network to notice small changes in features. However, at either tail of 0 or 1, the derivative values are very small, converging to 0. In this way, the gradients "vanish" making the model unable to learn (more specifically the learning is minimal). The sigmoid function is graphically illustrated in Figure 2.5

Hyperbolic Tangent (tanh) function:
The hyperbolic tangent function (tanh) is described by the following mathematical form:

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.23}$$

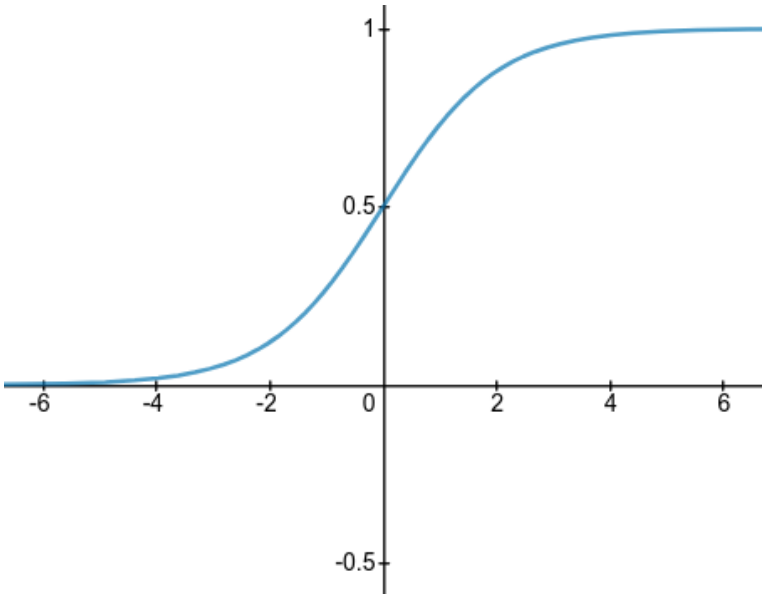


Figure 2.5: The sigmoid activation function.

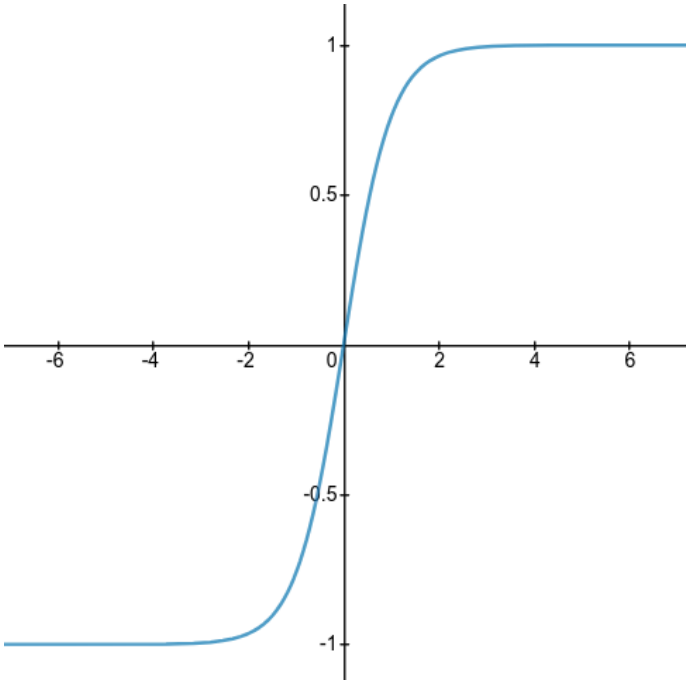


Figure 2.6: The tanh activation function.

The output values of the tanh function are in range $[-1, 1]$. The advantage over the sigmoid function is that its derivative is more steep, which means it can lead to higher value outputs. This means that it will be more efficient because it has a wider range for faster learning and grading. However, the same problem that the gradients converge to 0 at the tail of 0 and 1 still exists. A graphical illustration of the tanh function is shown in Figure 2.6.

Rectified Linear Unit (ReLU) function:
The ReLU activation function is described by the following mathematical form:

$$f(x) = x^+ = \max(0, x) \tag{2.24}$$

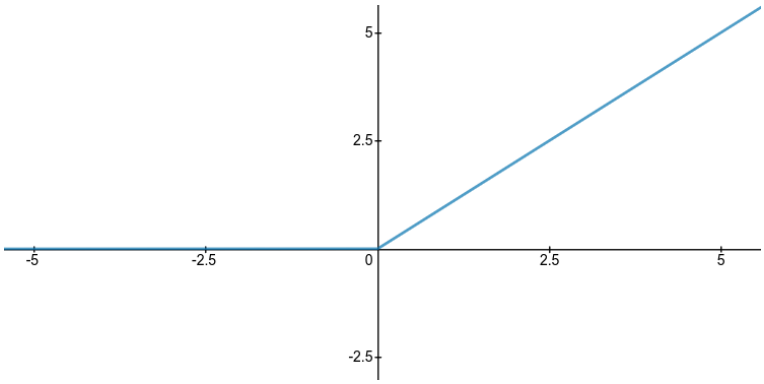


Figure 2.7: The Rectified Linear Unit (ReLU) activation function.

The ReLU activation function simply thresholds the input at zero. In contrast to sigmoid and tanh functions, it does not involve any computationally expensive operations, so it converges faster. We should also note here that the ReLU function avoids the vanishing gradient problem. Moreover, sigmoid and hyperbolic tangent functions cause almost all neurons to be activated in the same way, while the ReLU function is sparsely activated, making it more likely that neurons learn more meaningful aspects of the problem. However, that is not always the case, as neurons may get stuck by producing always 0 as output, if the input is negative. Finally, the problem of exploding gradients may occur, where weights are essentially "exploding", i.e., their values are rapidly increasing. A graphical illustration of the ReLU function is shown in Figure 2.7.

Gaussian Error Linear Unit (GELU) function:
The GELU activation function [68] is used in the most recent Transformers – Google's BERT [4] and OpenAI's GPT-2 [69]. It is described by the following mathematical form:

$$\text{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2}[1 + \text{erf}(x/\sqrt{2})] \tag{2.25}$$

where $\Phi(x) = P(X \leq x), X \sim \mathcal{N}(0,1)$ is the cumulative distribution function of the standard normal distribution. The GELU function can also be approximated as:

$$0.5x\left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right) \tag{2.26}$$

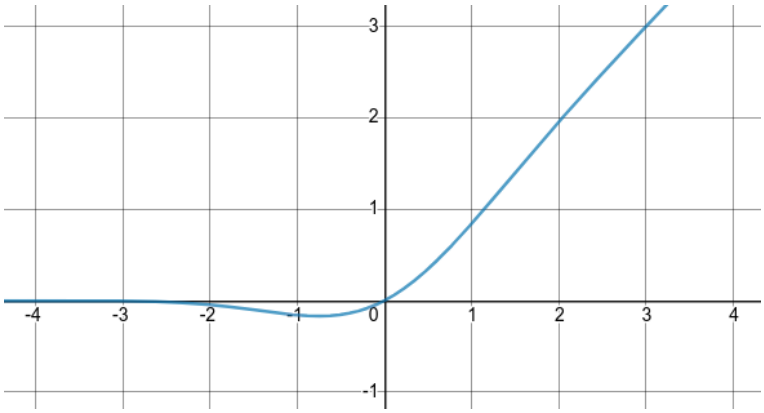


Figure 2.8: The Gaussian Error Linear Unit (GELU) activation function.

The GELU function has a negative coefficient, which shifts to a positive coefficient. So when x is greater than zero, the output will be x, except from when $x \in [0,1]$, where it slightly leans to a smaller

y-value. That formulation relates to stochastic regularizers because it is a modified expectation of adaptive dropout, providing neurons with more abstract probabilistic view. The GELU function also avoids the vanishing gradient problem and seems to be the state-of-the-art specifically in Transformer models (described in Chapter 4).

2.3.3 Backpropagation

Backpropagation is a widely used algorithm for training neural networks. During the training process, what we want to do is to find the optimal parameters (weights) for the model in order to minimize a given loss function. So, we have to compute the gradients. Even though the gradient computation for a neural network follows the chain rule [70], it can be very complex and error-prone. Fortunately, gradients can be efficiently computed using the backpropagation algorithm [71, 72]. Backpropagation methodically computes the derivatives of a complex expression using the chain-rule, while caching intermediary results. For updating the weights of the network after each computation of the loss function L, we use the partial derivatives $\frac{\partial L}{\partial w}$ of the loss function with respect to any learnable weight w of the network [70]. In this way, the model is being trained improving its performance.

2.3.4 Optimization

As already mentioned, during the training process of a model, we update the parameters of the model in order to minimize the loss. The procedure of optimally changing the weights of the model is called optimization. One of the most common categories of optimization algorithms is the gradient-based. The gradient at a certain point is the slope of the tangent to the function at that point, and it points to the direction of the steepest increase of the function. So, the weights of the model are updated in the opposite direction in order to minimize the loss. In the following part we analyse some of the most common gradient-based methods.

Gradient Descent (GD): The gradient descent algorithm computes the gradient of the loss function for the entire dataset with respect to the parameters θ of the model at each step (iteration) [70, 73]. Let $J(\theta)$ be the loss function and α small enough $\in \mathbb{R}$ the learning rate. The learnable parameters of the model are updated according to the following mathematical form:

$$\theta = \theta - \alpha \nabla_\theta J(\theta) \tag{2.27}$$

While the gradient descent algorithm is very simple to implement, it is not guaranteed to converge at a global minimum. It is possible to get stuck to a local minimum if the learning rate is not properly chosen. Moreover, if the dataset used is large, computing at each iteration the loss over the entire dataset may be computationally expensive and inefficient.

Stochastic Gradient Descent (SGD): The stochastic gradient descent algorithm [74] is a variant of gradient descent. In contrast to gradient descent it computes the gradient of the loss function over a subset of samples, and not for the whole dataset. So, it makes an estimation of the gradient using a sample, instead of computing the true gradient using all samples. Let $\mathbf{x}$ be the training examples and y the corresponding labels. The parameters are updated according to the following algorithm:

$$\theta = \theta - \alpha \nabla_\theta J\left(\theta; \mathbf{x}; \mathbf{y}\right) \tag{2.28}$$

Result: set of parameters θ
initialization of convergence criterion ϵ;
initialization of learning rate α;
initialization of parameters θ;
while $||\alpha \nabla_\theta J(\theta)|| > \epsilon$ **do**
    x, **y** = randomly chose n training samples;
    for *i=1,2,..n* **do**
        $\theta = \theta - \alpha \nabla_\theta J\left(\theta; \mathbf{x}^{(i)}; \mathbf{y}^{(i)}\right)$;
    end
end

Algorithm 1: SGD pseudo-alogrithm

This algorithm needs less time to converge to a minimum, which in large datasets is a significant aspect.

As we already mentioned, the gradient-based algorithms described do not guarantee convergence. On the one hand, choosing a small learning rate slows convergence making it also possible to get stuck to local minimums. On the other hand, choosing a very large learning rate can hinder convergence and cause the loss function to fluctuate around the minimum or even diverge. In Figure 2.9 an illustration of both circumstances is shown.

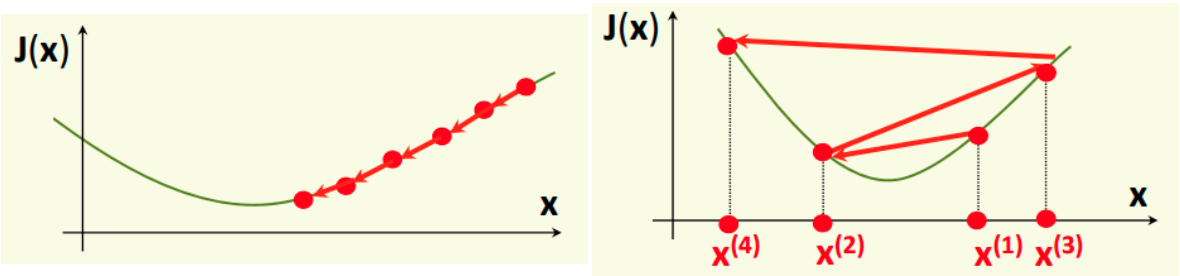


Figure 2.9: Choosing the learning rate is challenging as a too small value (left figure) may result in a long training process that could get stuck, whereas a too large value may result in learning a sub-optimal set of weights too fast or an unstable training process.

Adam: The Adam optimizer [75] differs from the previously described optimizers, where the learning rate remains the same for all weight updates. The Adam optimizer is an adaptive learning rate optimizer, which means that it keeps a learning rate for each parameter of the model, adapting it separately for each weight. Adam is well suited for problems that are large in parameters, is computationally efficient and has little memory requirement.

A number of alternate optimization algorithms have been introduced, similar to the Adam optimizer. Currently, Adam [75], Adagrad [76], Adadelta [77] and BertAdam [4] optimizers are the most widely used in deep neural models for NLP.

2.4 Deep Neural Networks

2.4.1 Recurrent Neural Networks

"Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence." [9]. Traditional neural networks can't do this and this is a major drawback. However, recurrent neural networks address this issue.

Recurrent neural networks (RNNs) are a type of artificial neural network designed to recognize patterns in sequences of data, such as numerical times series data emanating from sensors, stock markets and government agencies (but also including text, genomes, handwriting and the spoken word). What differentiates RNNs from other neural networks is that they take time and sequence into account. The core idea behind RNNs is to process information sequentially. Connections between units of an RNN form a directed graph along a sequence. They are called recurrent because they perform the same task for every element of a sequence, making the output dependent upon previous inputs. This allows it to exhibit temporal dynamic behavior for a time sequence and gives them the ability to have an "internal memory" which captures the information calculated so far. Intuitively, RNNs are able to remember important things about the input they received which enables them to make precise predictions for the data that comes next [64].

Vanilla RNNs: A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. A simple visualization of an RNN is displayed in Figure 2.10. Actually unrolling an RNN is just a way of better visualizing the behavior and the underlying mechanisms of this NN, but the core functionality is the same in both representations. At each time step, the input vector x_t is passed through the RNN. The RNN updates it's hidden state h_t (based on the input vector x_t and the RNN's hidden state of the previous time step h_{t-1}) , which forwards to the next successor and also gives as output the y_t. So, the RNN "remembers" the context of the input it has already seen while training.

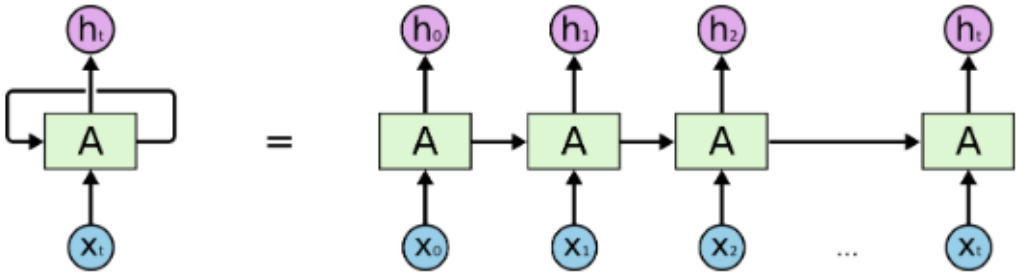


Figure 2.10: Unrolled Recurrent Neural Network. Image source: [9]

Formally, at each time step t, the equations that describe the function of the RNN are:

$$h_t = f_h\left(W_{hh}h_{t-1} + W_{hx}x_t + b_h\right) \tag{2.29}$$

$$y_t = f_y\left(W_{yh}h_t + b_y\right) \tag{2.30}$$

where h_t is the hidden state at time step t, x_t is the input vector at time step t, y_t is the output vector at time step t, b_h is the bias for h, b_y is the bias for y and f_x, f_h are the activation functions for x and h respectively. The are three separate matrices of weights: W_{hx} (input-to-hidden weights), W_{hh} (hidden-to-hidden), and W_{yh} (hidden-to-output).
Moreover, it is worth mentioning that we could stack piles of RNN layers on top of each other by simply connecting the activation of each cell at time step t, which is h_t, as an input to the next RNN layer for the same time step.

Bidirectional RNNs: As mentioned previously, the data passed through the RNN is processed sequentially and finally the last hidden state of the RNN captures the encoded information. It is also possible to acquire more information from the data, not only by processing the data forwardly but backwards too. A bidirectional RNN (BRNN) can be used for this purpose and it is illustrated in Figure 2.11.

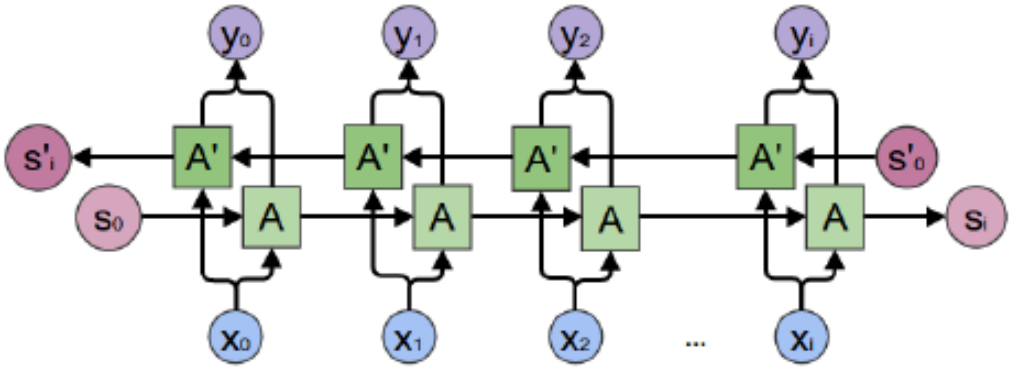


Figure 2.11: A Bidirectional Recurrent Neural Network. Practically, it's a left-to-right and a right-to-left RNN zipped together. Image source: [10]

So a bidirectional RNN consists of an RNN which processes the input sequence from the begging to the end (left to right), and another one which processes the input in reverse (right to left). Then, for each time step t, the hidden states are merged in order to have a final representation of the input sequence at each time step. More specifically, we compute the hidden state of the forward RNN $\overrightarrow{h_t}$, as well as the corresponding hidden state of the backward RNN $\overleftarrow{h_t}$ and then we concatenate them at each time step t. The hidden state at the last time step contains the information of the fully encoded sequence.

The problem of the long-term dependencies: One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous music frames, might inform the understanding of the present frame. Although the fact that in theory RNN are absolutely capable of handling "long-term dependencies", in practice they do not seem able to learn them [78, 79, 80, 81]. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the sky," we don't need any further context – it is pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information (illustrated in Figure 2.12). However, when trying to predict the last word in the text "I grew up in France… I speak fluent French.", recent information suggests that the word is the name of the language, but if we want to narrow down which language, we need the context of France, from further back. So there is a long-term dependence which the RNN can't learn. The reason of this failure, is the vanishing gradient problem (described in subsection 2.3.2) which Hochreiter identified first in his diploma thesis in 1991 [78].

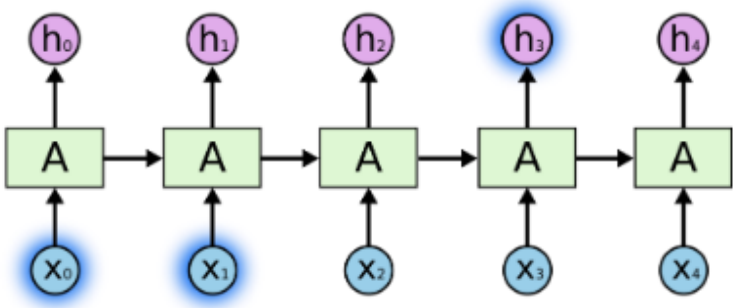


Figure 2.12: A short-term dependence being able to be learned from the RNN. Image source: [9]

Later, Long Short-Term Memory Networks (LSTMs) were proposed by Sepp Hochreiter [82],overcoming the problem of the vanishing gradient, and consequently capturing better long-term dependencies.

LSTMs: The Long Short-Term Memory networks partially overcome the problem mentioned above by preserving long-term dependencies using the cell state. [82, 83]

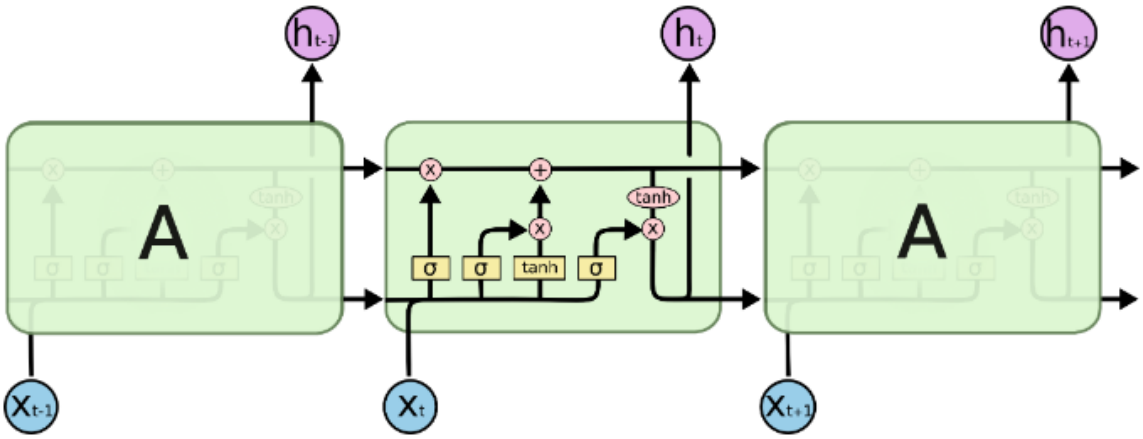


Figure 2.13: The inner architecture of the LSTM unit, containing four interacting layers. Image source:[9]

The core components of the LSTM architecture are the input,forget and output gates and of course the cell state. After a formal explanation, we will explain more those components.
Given a sequence $x_1, x_2, \ldots, x_t, \ldots, x_n$ of vectors of an input sequence of length n, for vector x_t, with inputs h_{t-1} and c_{t-1}, h_t and c_t are computed as follows:

$$f_t = \sigma\left(W_f[h_{t-1}; x_t] + b_f\right) \tag{2.31}$$

$$i_t = \sigma\left(W_i[h_{t-1}; x_t] + b_i\right) \tag{2.32}$$

$$o_t = \sigma\left(W_o[h_{t-1}; x_t] + b_o\right) \tag{2.33}$$

$$\tilde{c}_t = \tanh\left(W_c[h_{t-1}; x_t] + b_c\right) \tag{2.34}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.35}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{2.36}$$

Now we will intuitively explain the main components:

Input gate($\mathbf{i}_t$): The input state regulates the amount of information that will flow from the sigmoid activation of the previous time step h_{t-1} alongside with the information of the current input vector x_t when we are updating the current state weights. More formally, the current input vector x_t and the hidden state of the previous time step h_{t-1} are passed into a sigmoid activation function which forces the output values to range between 0 and 1 (0 means unimportant, 1 means important). Then those values are used for updating the current states.

Forget gate(f_t): The forget gate controls the information that would be kept or thrown away. The current input vector x_t is passed into a sigmoid function alongside with the hidden state of the previous time step h_{t-1}. The output values of the sigmoid function range between 0 and 1, which means forgetting or keeping the relevant information.

Cell state(c_t): The cell state is computed by adding the the information which has been controlled by the aforementioned gates(input and forget gate). The cell state of the previous time step c_{t-1} is pointwise multiplied by the "forget" vector f_t, controlling which information to forget and which not. The current input vector x_t and the previous hidden state h_{t-1} are also passed to the tanh function to squish values between -1 and 1 ($\tilde{c}_t$). Then, i_t is pointwise multiplied with the $\tilde{c}_t$. So the input gate filters the important information of the input vector and the previous hidden state to be kept. Finally we add those two terms and we have the cell state.

Output gate(o_t): The output gate decides what the new hidden state would be. The hidden state of the previous time step h_{t-1} and the current input vector are passed into a sigmoid function. Then, the cell state is passed to the tanh function. Finally, we multiply o_t with the output of the tanh function, in order to determine which information the new hidden state will carry.

GRUs: The researchers in [84] introduced the Gated recurrent unit (GRU). The GRU is like an LSTM with a forget gate, but has fewer parameters as it lacks the output gate. The LSTM is "strictly stronger" than GRU as claimed by Gail Weiss, Yoav Goldberg and Eran Yahav [85]. The inner architecture of the GRU unit is illustrated in Figure 2.14. Following the previous notation, the equations that describe

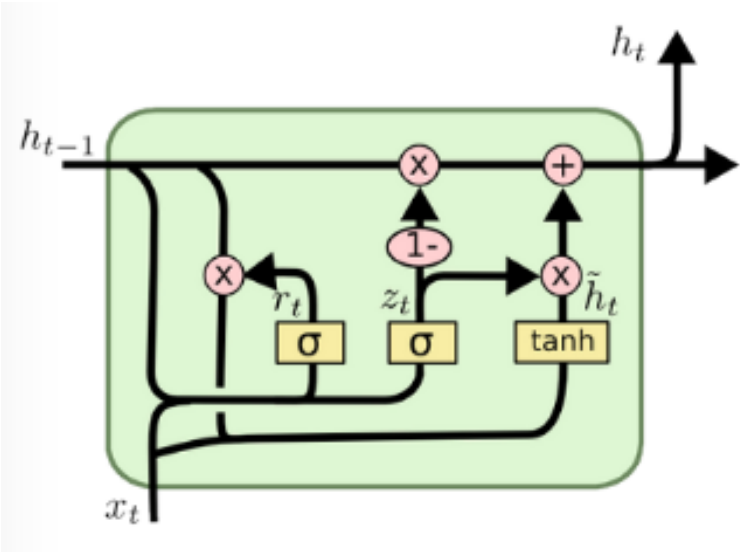


Figure 2.14: The inner architecture of the GRU unit. Image source: [11]

the function of the GRU are the following:

$$z_t = \sigma\left(W_z[h_{t-1}; x_t]\right) \tag{2.37}$$

$$r_t = \sigma\left(W_r[h_{t-1}; x_t]\right) \tag{2.38}$$

$$\tilde{h}_t = \tanh\left(W[r_t \odot h_{t-1}; x_t]\right) \tag{2.39}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.40}$$

where σ is the logistic sigmoid, $\sigma(x) \in [0, 1]$, $\odot$ represents the element-wise scalar product between vectors, and W_z, W_r, W are matrices to be learned.

2.4.2 Attention Mechanisms

Attention has been a fairly popular concept and a useful tool in the deep learning community in recent years.The idea of attention,is to some extent, motivated by how we pay visual attention to different regions of an image or correlate words in one sentence. For example, let's take a look in the picture of a Shiba Inu in Figure 2.15. Human visual attention allows us to focus on a certain region of an image with "high resolution"(i.e., focusing on the ear on the yellow box), while perceiving the surrounding image in "low resolution"(i.e., we do not pay attention to the snowy background or to the sweater). Given a small patch of an image, pixels in the rest provide clues of what should be displayed there. We expect to see a pointy ear in the yellow box, because we have seen a dog's nose, another pointy ear on the right, and Shiba's mystery eyes (stuff in the red boxes). However, the snowy background and the sweater would not be as helpful as the aforementioned doggy features. Similarly, we can explain the relationship between words in one sentence or close context. For example, when we see "eating", we expect to encounter a food word very soon.

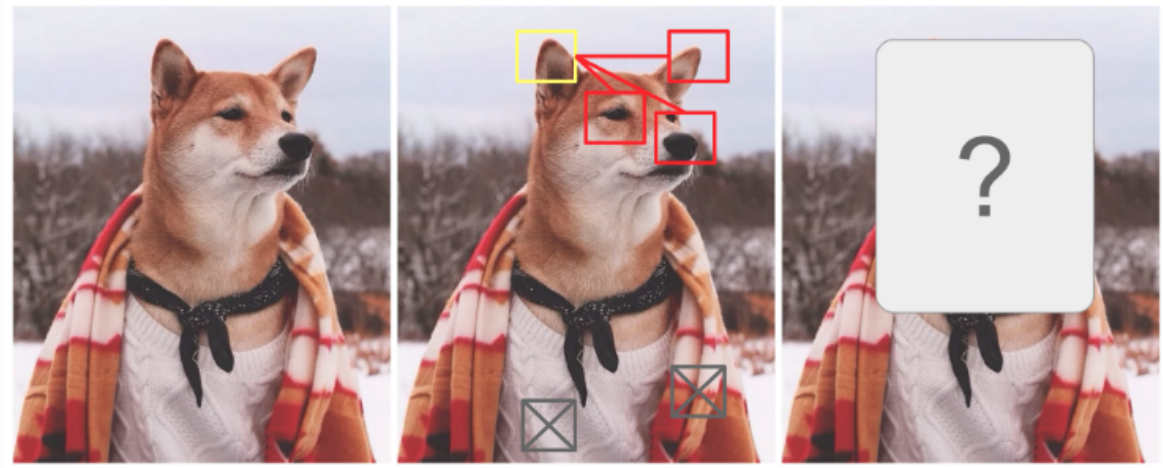


Figure 2.15: A Shiba Inu in a men's outfit. An example of visual attention. [86]

So, the attention mechanism in deep learning is based on the concept of directing our focus, and paying greater attention to certain factors when processing the data. Attention is most commonly used in sequence-to-sequence models [87], as the fixed-length context vector, created by sequence-to-sequence models, is incapable of representing long-term sequences. Once attention is introduced [12] the problem is solved, as instead of attempting to learn a single representation for each sentence, the model pays attention to specific input vectors of the input sequence, based on the attention weights. Using attention, we obtain a context vector $\mathbf{c_i}$, which consumes three pieces of information:

- Encoder's hidden state (encoder model described in Section 4.3)
- Decoder's hidden state (decoder model described in Section 4.3)
- alignment between source and target

Table 2.1: A summary table of several popular attention mechanisms

Name	*Score − function*	*Citation*
Content-base attention	score $\left(h_i^{(dec)}, h_j^{(enc)}\right) = \text{cosine}\left[h_i^{(dec)}, h_j^{(enc)}\right]$	[88]
Additive(*)	score $\left(h_i^{(dec)}, h_j^{(enc)}\right) = v_a^\top \tanh\left(W_a\left[h_i^{(dec)}; h_j^{(enc)}\right]\right)$	[12]
Location-base(**)	$\alpha_{i,j} = \text{softmax}\left(W_a h_i^{(dec)}\right)$	[14]
General	score $\left(h_i^{(dec)}, h_j^{(enc)}\right) = {h_i^{(dec)}}^\top W_a h_j^{(enc)}$	[14]
Dot-product	score $\left(h_i^{(dec)}, h_j^{(enc)}\right) = {h_i^{(dec)}}^\top h_j^{(enc)}$	[14]
Scaled-Dot-product (***)	score $\left(h_i^{(dec)}, h_j^{(enc)}\right) = \frac{{h_i^{(dec)}}^\top h_j^{(enc)}}{\sqrt{n}}$	[3]

(*) Referred to as "concat" in [14] and as "additive attention" in [3].
(**) This simplifies the softmax alignment to only depend on the target position.
(***) It adds a scaling factor $1/\sqrt{n}$ (n is the dimension of the source hidden state), motivated by the concern that when the input is large, the softmax function may have an extremely small gradient, making it hard to use for efficient learning.

Table 2.2: A summary table of broad categories of attention types

Name	*Description*	*Citation*
Self attention	Relating different positions of the same input sequence.	[13]
Global/Soft attention	Attending to the entire input state space.	[89]
Local/Hard attention	Attending to part of input state space; i.e., a patch of the input image.	[89, 14]

Let's assume that the encoder network has hidden states $h_1^{(enc)}, h_2^{(enc)}, \ldots, h_n^{(enc)}$ and the decoder network has hidden state $h_i^{(dec)}$ at current time-step i. The context vector c_i is calculated at time-step i as an average of the encoder's states weighted with the attention scores α_i:

$$c_i = \sum_{j=1}^{n} \alpha_{i,j} h_j^{(enc)} \tag{2.41}$$

$$\alpha_{i,j} = \text{softmax}\left(score\left(h_i^{(dec)}, h_j^{(enc)}\right)\right) \tag{2.42}$$

where the softmax function is defined as:

$$\text{softmax}\left(z_i\right) = \frac{e^{z_i}}{\sum_{j=1} K e^{z_j}} \tag{2.43}$$

where i = 1,..,K and $z = (z_i,\ldots,z_K) \in \mathbb{R}^K$. The $score(h_i^{(dec)}, s_j^{(enc)})$ function calculates an unormalized alignment score defining how much each source hidden state must be considered for each output. So the weights $\alpha_{i,j}$ are calculated based on how well match the pair of input at position j and output at position i. A summary of attention mechanisms and the corresponding score functions is shown in Table 2.1. A summary of broader categories of attention mechanisms is shown in Table 2.2.
In the following we analyse some of the attention mechanisms shown in Tables 2.1,2.2.

Additive attention

The original (additive) attention mechanism proposed by [12] uses a feed-forward neural network with a single hidden layer to calculate the alignment scores. This network is jointly trained with the other parts of the model. The $score$ function is therefore in the following form given that $tanh$ is used as the non-linear activation function:

$$\text{score}\left(h_i^{(dec)}, h_j^{(enc)}\right) = v_a^\top \tanh\left(W_a\left[h_i^{(dec)}; h_j^{(enc)}\right]\right) \tag{2.44}$$

where v_a and W_a are both weight matrices to be learned in the alignment model. We can also extend this method by learning separate transformations for $h_i^{(dec)}$ and $h_j^{(enc)}$. Assuming the weight matrices W_1 and W_2 for $h_i^{(dec)}$ and $h_j^{(enc)}$ respectively, we can then sum the transformed matrices as shown below:

$$\text{score}\left(h_i^{(dec)}, h_j^{(enc)}\right) = v_a^\top \tanh(W_1 h_i^{(dec)} + W_2 h_j^{(enc)}) \tag{2.45}$$

The matrix of alignment scores is a nice byproduct to explicitly show the correlation between source and target words, as the one shown in Figure 2.16.

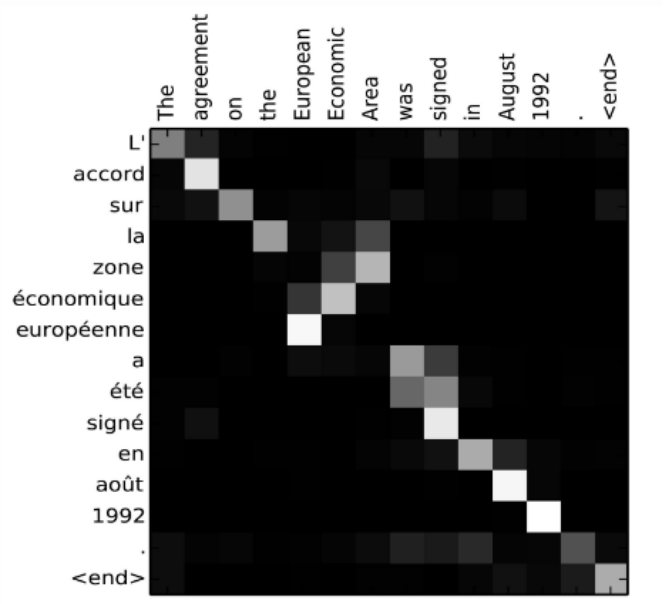


Figure 2.16: Alignment matrix of "L'accord sur l'Espace économique européen a été signé en août 1992" (French) and its English translation "The agreement on the European Economic Area was signed in August 1992". We notice that the attention mechanism when translating the French word "zone" pays attention to the English word "Area" in spite of the fact that the words are not aligned. Image source: [12]

Multiplicative attention

The researchers in [14] proposed multiplicative (known also as general) attention, which simplifies the attention method by calculating the score function as shown in the following equation:

$$\text{score}\left(h_i^{(dec)}, h_j^{(enc)}\right) = {h_i^{(dec)}}^\top W_a h_j^{(enc)} \tag{2.46}$$

In spite of the fact that additive and multiplicative attention have similar complexity, multiplicative attention is more widely used as it is faster and more space-efficient in practice as it can be implemented more efficiently using matrix multiplication. While in small dimensionalities of the decoder

states d_h, both methods perform equally well, in larger dimensions additive attention seems to lack efficiency [90]. One way to mitigate this is to scale $score(h_i^{(dec)}, h_j^{(enc)})$ by $1/\sqrt{d_h}$ [3].

Self-attention

Self-attention, also known as intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading [13], abstractive summarization [91], textual entailment [92] or image description generation [93]. So, by using self-attention without any additional information, we can still extract relevant aspects from the sentence by allowing it to attend to itself [94] and learn the correlation between the current words and the previous part of the sentence. Theoretically self-attention can adopt any score functions, by just replacing the target sequence with the same input sequence. More specifically, the unormalized alignment scores can be calculated for each hidden state h_i using the score function shown in the following equation:

$$\text{score}(h_i) = v_a^\top \tanh(W_a h_i) \tag{2.47}$$

So assuming that we have a matrix H for all hidden states given as $H = [h_1, h_2, \ldots, h_n]$ we can calculate the attention matrix a and the final representation c using the following equations:

$$a = \text{softmax}(v_a^\top \tanh(W_a H^\top)) \tag{2.48}$$

$$c = Ha^\top \tag{2.49}$$

The example in Figure 2.17, shows how the self-attention mechanism enables us to learn the correlation between the current words and the previous part of the sentence in the field of reading comprehension.

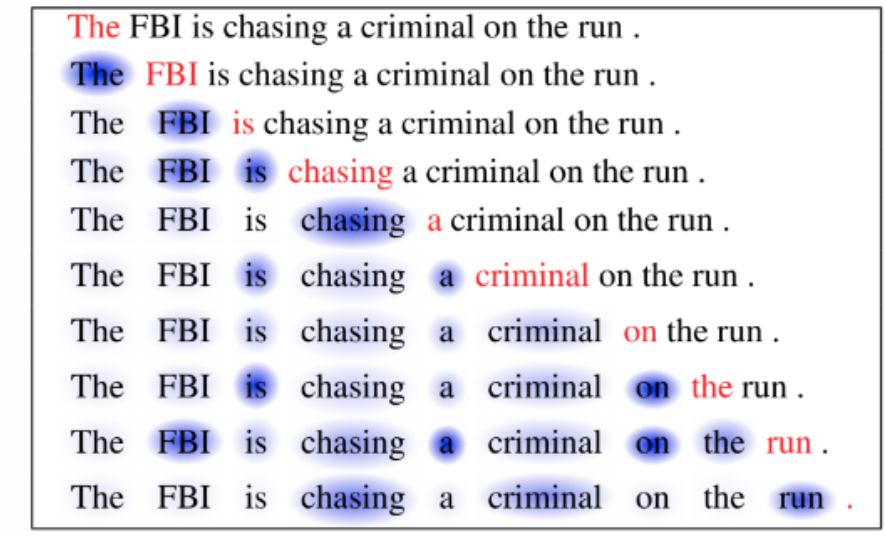


Figure 2.17: The current word is in red and the size of the blue shade indicates the activation level. Image source: [13]

Soft and Hard attention

Soft attention: In soft attention, the attention mechanism has access to the entire source. The alignment weights are learned and placed "softly" over all source positions. This type of attention makes the model smooth and differentiable. However, it is really expensive when the input is large.

Hard attention: In hard attention, the attention mechanism selects a specific window of the input to attend. So, the calculations needed during inference time are fewer, but the model is non-differentiable and requires more complicated techniques such as variance reduction or reinforcement learning to train.

Global and Local attention

[14, 95] Global and Local attention mechanisms differ in terms of whether the attention is placed over all source positions or only a few source positions. These two mechanisms were proposed by [14] expanding the encoder-decoder architecture referred in Section 4.3. Common to these attention mechanisms is the fact that both at each time step t during the decoding phase receive as input the decoder's hidden state $h_t^{(dec)}$ and the goal is to produce the context vector c_t that captures relevant source-side information to help predict the current target word y_t. More specifically, given the decoder's hidden state $h_t^{(dec)}$ and the context vector c_t, a simple concatenation layer is used to combine the information from both vectors and produce the new attention hidden state, as shown in the following equation:

$$\tilde{h}_t = \tanh\left(W_c\left[c_t; h_t^{(dec)}\right]\right) \tag{2.50}$$

The attention hidden state $\tilde{h}_t$ is then used for predicting the current word y_t. What differs in global and local attention is the computation of the context vector c_t.

Global attention: In global attention the context vector c_t is produced by paying attention to all source positions, so the model considers all the hidden states from the encoder $h_j^{(enc)}$. In this attention type, a variable-length alignment vector a_t, whose size equals the number of time steps on the source side, is derived by comparing the current decoder's hidden state $h_t^{(dec)}$ with each source (encoder's) hidden state $h_j^{(enc)}$:

$$\begin{aligned} a_t(j) &= \text{align}\left(h_t^{(dec)}, h_j^{(enc)}\right) \\ &= \frac{\exp\left(\text{score}\left(h_t^{(dec)}, h_j^{(enc)}\right)\right)}{\sum_{j'}\exp\left(\text{score}\left(h_t^{(dec)}, h_{j'}^{(enc)}\right)\right)} \end{aligned} \tag{2.51}$$

where the score function is described with one of the three different alternatives following:

$$\text{score}\left(h_t^{(dec)}, h_j^{(enc)}\right) = \begin{cases} {h_t^{(dec)}}^\top h_j^{(enc)} & \text{dot} \\ {h_t^{(dec)}}^\top W_a h_j^{(enc)} & \text{general} \\ v_a^\top \tanh\left(W_a\left[h_t^{(dec)}; h_j^{(enc)}\right]\right) & \text{concat} \end{cases} \tag{2.52}$$

The fact that the global attention mechanism has to attend to all words on the source side for each target word, is a main drawback, as this process is expensive and can potentially render it impractical to translate longer sequences, e.g., paragraphs or documents. An illustration of the global attention mechanism is shown in Figure 2.18.

Local attention: The local attention mechanism is an interesting blend between hard and soft attention. An improvement over the hard attention mechanism has been made in order to make the model differentiable. In local attention, the context vector c_t is produced by paying attention to a few source positions. By selectively focusing on a small window of context, the process has an advantage of avoiding the expensive computation incurred in the soft and global attention and also makes it easier to train the model than the hard attention approach as the model is differentiable. More specifically, the model first generates an aligned-position p_t for each target word at time step t. The context

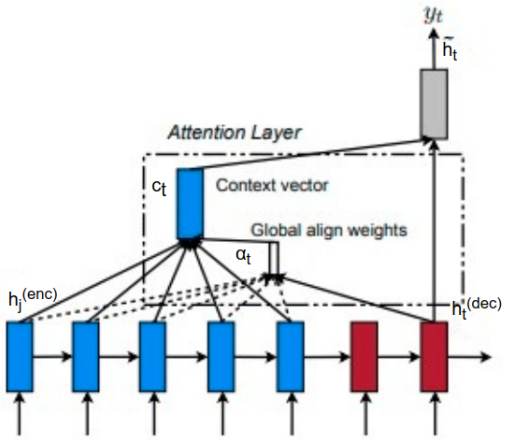


Figure 2.18: Global attentional model: at each time step t, the model infers a variable-length alignment weight vector a_t based on the current target state $h_t^{(dec)}$ and all source states $h_j^{(enc)}$. A global context vector c_t is then computed as the weighted average, according to a_t, over all the source state. Image source: [14]

vector c_t is then derived as a weighted average over the set of source (encoder's) hidden states within the window $[p_t - D, p_t + D]$, where D is empirically selected. The local alignment vector a_t is now fixed-dimensional, i.e., $\in \mathbb{R}^{2D+1}$. This type of attention has two variants, depending on the selection of the aligned-position p_t. In the first variant, the monotonic alignment (local-m), it is assumed that the source and target sequences are roughly monotonically aligned, by setting $p_t = t$. Then, the alignment vector a_t is calculated with equation 2.51. In the second variant, the predictive alignment (local-p), the model predicts the alignment positions as follows:

$$p_t = S \cdot \text{sigmoid}\left(v_p^\top \tanh\left(W_p h_t^{(dec)}\right)\right) \tag{2.53}$$

where W_p and v_p are model's matrices to be learned and S is the is the source sentence length. As a result of the sigmoid use, $p_t \in [0, S]$. To favor alignment points near p_t, a Gaussian distribution centered around p_t is used. Specifically, the alignment weights are now defined as:

$$a_t(j) = \text{align}\left(h_t^{(dec)}, h_j^{(enc)}\right) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right) \tag{2.54}$$

where the standard deviation is empirically set as $\sigma = \frac{D}{2}$ and the align() function is the same as in equation 2.51. We should also note that p_t is a real number, whereas s is an integer within the window centered at p_t. The local attention mechanism is very similar to the selective attention proposed by [96], which is used in image generation tasks. An illustration of the local attention mechanism is shown in Figure 2.19.

Key-Value-Query attention

This type of attention mechanism was introduced by [3] and is used in the Transformer model described in Section 4.6 (using self-attention). The attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Based on the Key-Value-Query attention we analyse Scaled-Dot-Product and Multi-Head attention mechanisms which were proposed in [3].

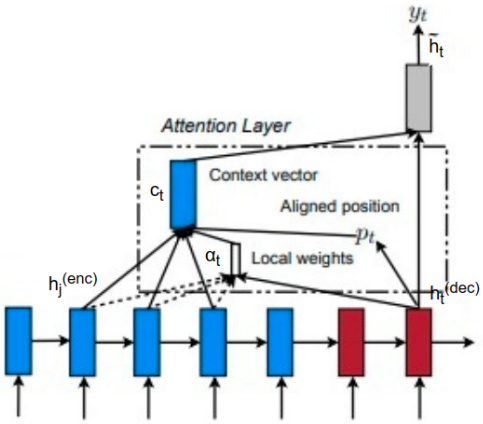


Figure 2.19: Local attention model: the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector c_t, a weighted average of the source hidden states in the window. The weights a_t are inferred from the current target state $h_t^{(dec)}$ and those source states $h_s^{(enc)}$ in the window. Image source: [14]

Scaled-Dot-Product attention

In Scaled-Dot-Product attention the input consists of queries and keys of dimension d_k and values of dimension d_v. The dot-products of each query with all keys is calculated and then each one is divided by $\sqrt{d_k}$ and finally a softmax function is applied to obtain the weight of each value. The output is computed as a weighted sum of the values and the respective weights. More specifically, all queries, keys and values are packed together into the matrixes Q,K,V respectively. So, the attention function computes the weights for all values simultaneously as follows:

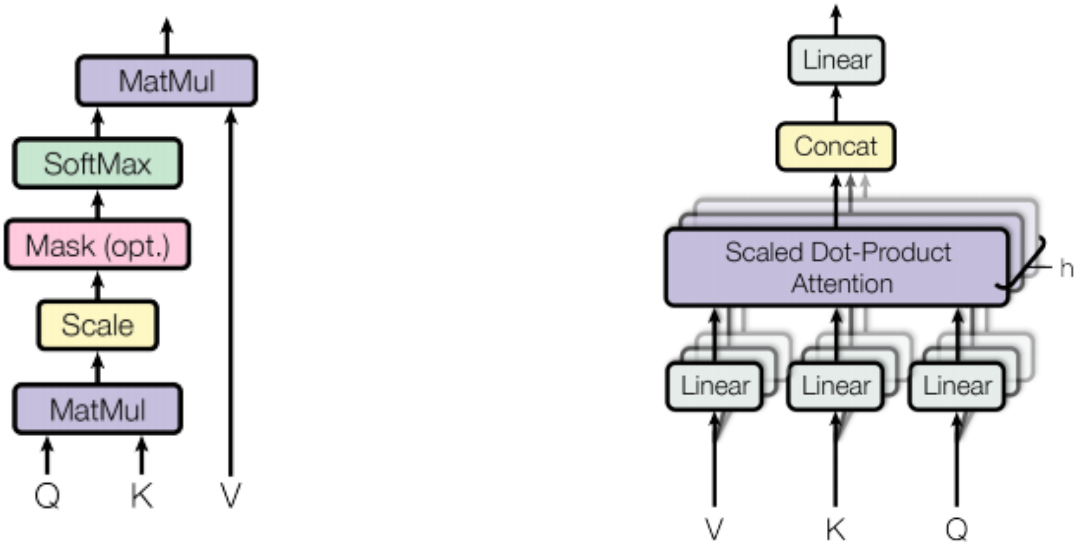
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.55}$$

Additive attention can also be used insted of dot-product (multiplicative) attention. As mentioned before, additive attention uses a feed-forward network with a single hidden layer. However, while the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code. While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot-product attention without scaling for larger values of d_k [97]. So using large values of d_k causes the dot products to grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. In order to counteract this effect, the dot-products are divided by $\sqrt{d_k}$. An illustration of scaled-dot-product attention is shown in Figure 2.20a

Multi-Head attention

The Multi-head attention mechanism extends the scaled-dot-product attention mechanism. Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, it linearly projects the queries, keys and values h times, with different learned linear projections to d_k, d_k and d_v dimensions, respectively. In this way, the model can jointly attend to information from different representation subspaces at different positions. The multi-head attention function is described as follows:

$$\begin{aligned} \text{MultiHead}\,(Q, K, V) &= \text{Concat}\,(\,\text{head}_1, \ldots,\, \text{head}_h)\, W^O \\ \text{where head}_i &= \text{Attention}\,\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.56}$$



(a) Scaled-Dot-Product attention using queries, keys and values.

(b) Multi-Head attention using queries, keys and values.

Figure 2.20: Scaled-Dot-Product and Multi-Head attention mechanisms. Image source: [3]

where the projection parameters are $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$. It's worth mentioning that in [3] the dimensions d_k and d_v are set as $d_k = d_v = d_{\text{model}}/h$ so that with the reduced dimension of each head, the total computational cost is similar to that of the single-head attention with full dimensionality. The multi-head attention mechanism is illustrated in Figure 2.20b.

2.5 Transfer Learning

As we have already mentioned, in many deep learning tasks we need a plethora of data in order to build a model that produces satisfying results. However, in many practical applications there are a few data available with tags to do a supervised training of the model as getting the amounts of data required for supervised models can become unfeasible due to time restrictions and computational limitations. It is also noticed that training a model on a small specific dataset may not be as effective. In these cases, the use of transfer learning offers an alternative solution.

The goal of transfer learning is to improve the performance of the model in a specific task (the target task), by utilizing knowledge acquired from training on the source task [98]. More formally, given a source domain D_S , a corresponding source task T_S , as well as a target domain D_T and a target task T_T , transfer learning's objective is to enable us to learn the target conditional probability distribution $P(Y_T|X_T)$ in D_T with the information gained from D_S and T_S , where X_T is the feature space and Y_T is the label space of task T and $D_S \neq D_T$ or $T_S \neq T_T$. Through the pre-training process on the source task, the model gains initial knowledge and learns to create high-level representations. So, when it is then fine-tuned on the target task, not only it learns faster, but it also reaches a higher performance level compared to the performance level achieved without transfer learning [99].

It is also worth mentioning the special case where the source task is an unsupervised task. This is an interesting scenario because we often have very large amounts of unlabeled training data, that can be used during pre-training boosting the model's performance by far. Such a scenario is also found in dialogue systems, where the models are pre-trained as language models and then are fine-tuned on

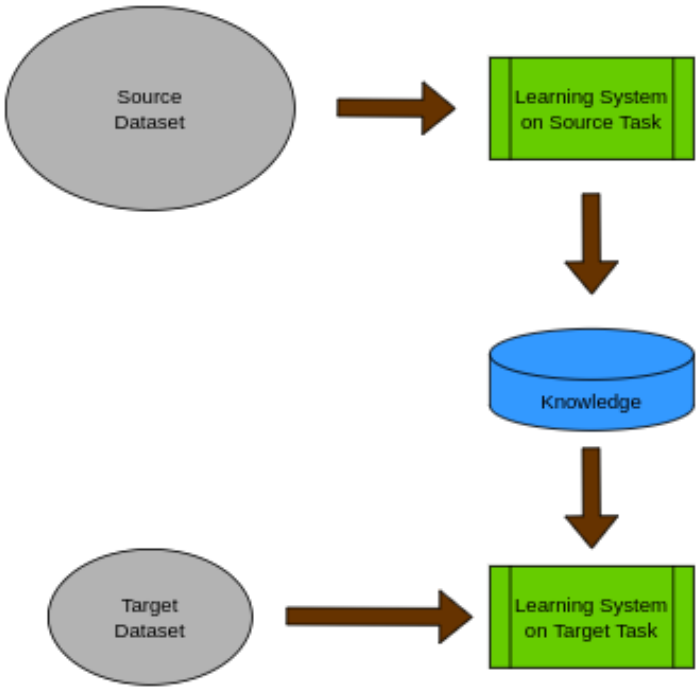


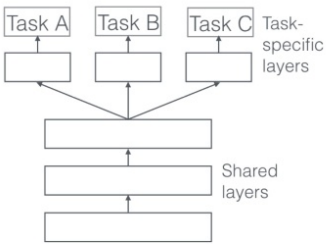
Figure 2.21: An illustration of the transfer learning technique.

the target dialog task.

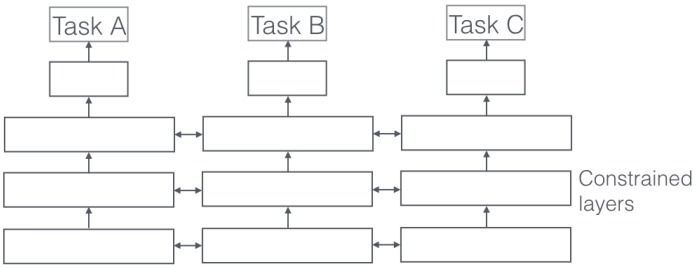
The process of transfer learning is illustrated in Figure 2.21.

2.6 Multi-task Learning

Multi-task learning (MTL) is a learning technique in which we want to train a model in many related tasks simultaneously. In other words, we want the model to generate predictions for multiple tasks at the same time. The motivation behind that is to try to take advantage of the information in one of the problems so that we can improve the performance in the other problems [100, 101]. In deep learning, the basic idea of implementing multi-task learning is to have different networks that are part of the same structure and have a few common parameters. Thus, the shared part is affected by all problems, while the non-shared parts are affected by the training data of each task independently.



(a) Hard parameter sharing approach.



(b) Soft parameter sharing approach.

Figure 2.22: Multi-task learning techniques. Image source:[15]

Multi-task learning can be implemented with hard or soft parameter sharing. In hard parameter shar-

ing all the hidden layers of the network are shared among all tasks, while there is an independent output head for each task for making a suitable prediction. Hard parameter sharing reduces the risk of overfitting. The idea behind this is that the more problems the model tries to learn to solve at the same time, the more it is forced to find a common representation that is effective for all problems, making a better generalization. On the other hand, in soft parameter sharing, some of the hidden parameters are shared and others not. The output heads are independent as in the first case, of course. An illustration of both methods is given in Figure 2.22.

2.7 Summary

In this Chapter, an introduction to the basic principles and the theoretical background of machine learning and deep neural networks is given. This background is cornerstone of this diploma thesis as some of the models which we will study in the next sections are based on the basic principles we have described. Specifically, a basic understanding of recurrent neural networks, attention mechanisms, transfer learning and multi-task learning is very important in order to fully grasp the ideas and experiments presented in the following. In the next Chapter we introduce the reader to the objective of Natural Language Processing (NLP), a significant field of research for understanding dialog systems.

Chapter 3

Natural Language Processing

3.1 Introduction

Everything we express, either verbally or in writing, carries huge amounts of information. The topic we choose, our tone, our selection of words, everything adds some type of information that can be interpreted and value can be extracted from it. In theory, we can understand and even predict human behavior using that information. However, when having thousands of declarations to analyze, it is almost impossible trying to encode text or speech with specific keywords. So, the matter is to understand the meaning behind those words

Natural Language Processing (NLP) is a field of computer science, artificial intelligence and linguistics, concerned with the interactions between computers and human (natural) languages. It is the process of a computer extracting meaningful information from natural language input and/or producing natural language output. It is an analysis of human language based on semantics and various parsing techniques [102]. NLP may focus on language processing or generation. The first of these refers to the analysis of language for the purpose of producing a meaningful representation, while the latter refers to the production of language from a representation. The task of language processing is equivalent to the role of the reader/listener, while the task of language generation is that of the writer/speaker. Much of the theory and technology are shared by these two divisions. [103]

NLP is performed by solving a number of sub-problems, where each sub-problem constitutes a level. We should note here that, a portion of those levels could be applied, not necessarily all of them. For example, some applications require the first three levels only. Also, the levels could be applied in a different order independent of their granularity. The levels that NLP examines are:

Level 1 - Phonology: This level is applied only if the text's origin is a speech. It deals with the interpretation of speech sounds within and across words [104]. There are, in fact, three types of rules used in the phonological analysis:

- phonetic rules – for sounds within words
- phonemic rules – for variations of pronunciation when words are spoken together
- prosodic rules – for fluctuation in stress and intonation across a sentence

Level 2 - Morphology: Deals with understanding distinct words according to their morphemes (morphemes are the smallest units of meanings). In other words, this level deals with words formation. English words are generally composed of a stem and an optional set of affixes. The stem, as a morpheme that cannot be removed, is the true morphological base of an English word. For example, the word "preregistration" can be morphologically analyzed into three separate morphemes: the prefix "pre", the root "registra", and the suffix "tion". [105]. This level of NLP can be useful in retrieval-based dialog systems, where in order to match more responses to a query, words can be stemmed.

Level 3 - Lexical: This level deals with understanding everything about distinct words according to their position in the speech, their meanings, and their relation to other words. Each word is analyzed with respect to its lexical meaning and part-of-speech. In this level words are assigned with a part-of-speech tag.

Level 4 - Syntactic: This level deals with analyzing the words of a sentence so as to uncover the grammatical structure of the sentence. Generally, it focuses on the scientific study of the structure of the sentence as an independent unit. Using the part-of-speech tagging output of the previous level we group words into phrases and study their grammatical structure. [106].

Level 5 - Semantic: Semantic processing determines the possible meanings of a sentence by relating the syntactic features and the different meanings of each word according to the context. Some people may think it's the level that determines the meaning, but actually, all the levels do [107, 108].

Level 6 - Discourse: While previous levels work with word-level and sentence-level units, the discourse level of NLP works with units of text longer than a sentence. It does not interpret texts as just concatenated sentences, rather it focuses on the properties of the text as a whole that convey meaning by making connections between component sentences [103]. In other words, in this level we study the anaphora relationships between words in text longer than a sentence.

Level 7 - Pragmatic: This level is concerned with the purposeful use of language in situations and utilizes context over and above the contents of the text for understanding. In other words, this level deals with the use of real-world knowledge and understanding of how this impacts the meaning of what is being communicated [103]. For example, let's consider the following two sentences:

- The city councilors refused the demonstrators a permit because they feared violence.
- The city councilors refused the demonstrators a permit because they advocated revolution.

As we can see, the meaning of "they" in the two sentences is different. In order to figure out the difference, "world knowledge" should be utilized.

3.2 Applications

Natural Language Processing is among the hottest topics in the field of data science. Companies are putting tons of money into research in this field, as they can benefit from it because any application that utilizes text is a candidate for NLP. The most common applications of NLP are:

Information Retrieval: The science of searching for documents, for information within documents, and for metadata about documents.

Information Extraction: The recognition, tagging, and extraction into a structured representation,of certain key elements of information, e.g., persons, companies, locations, organizations, from large collections of text.

Automatic Speech Recognition: The task of automatically recognizing speech or in other words extracting a textual representation of a spoken uttterance.

Natural Language Understanding (NLU): The task of understanding the natural language and producing computer-machine based representations.

Natural Language Generation (NLG): The task of generating natural language from computer-machine based representations.

Language Modeling: The task of determining the probability of a given sequence of words occurring in a sentence. Language models analyze text data and are able of predicting the next word or character in a document.

Dialogue Systems or Converastional Agent (CA): A computer system intended to converse with humans.

Machine Translation: The use of computer software in order to translate text or speech from one natural language to another.

Text Classification: A method for classifying texts (sentences, documents, etc) into a variety of specific categories.

Text Summarization: The task of producing a shorter version of one or several documents that preserves most of the input's meaning.

Named Entity Recognition: The task of tagging entities in text with their corresponding type. Approaches typically use BIO notation, which differentiates the beginning (B) and the inside (I) of entities. O is used for non-entity tokens.

Sentiment Analysis: The task of classifying the polarity of a given text. Sentiment Analysis, in its simplest form, aims to detect positive, neutral, or negative feelings from text.

3.3 Language Representations

In this section, we examine several ways through which words are transformed into mathematical representations, in order to be used as input in our models to solve a variety of problems of NLP. It is worth to mention that long before the organized establishment of the Natural Language Processing Industry/Community, the idea of using separate symbols for each word prevailed. However, a number of ways were proposed later to create more accurate representations. In the most recent studies, efforts are made to create a vector representation for each word in the vocabulary. Those vector representations are called $word\ embeddings$ or $word\ vectors$. The goal is to create word vectors that carry a sense of similarity or differentiation between words that are similar or unrelated respectively. So, once words are converted into word vectors, a variety of distance metrics such as Jaccard, Euclidean, Cosine, etc., can be used to model the similarity or differentiation between them.

There are two different semantic approaches to creating word representations. The first one, which is called denotational semantics, treats words as separate symbols, creates more sparse representations, and does not give any sense of similarity or dissimilarity between words (localist representation). On the contrary, the second one, called distributional semantics, creates representations based on context, retaining similarities or dissimilarities between words. Distributional semantics embraces a wide range of approaches based on the distributional hypothesis, in an attempt to capture meanings of linguistic entities (words, phrases) from their usage in language. The idea of the distributional hypothesis is that the distribution of words in a text holds a relationship with their corresponding meanings. More specifically, the more semantically similar two words are, the more they will tend to show up in similar contexts and with similar distributions. It is summarized with the statement that "words that occur in the same contexts tend to have similar meanings" [109]. Also, the hypothesis is often described by the famous quote "a word is characterized by the company it keeps" [110]. The direct implication of this hypothesis is that two words that are considered to be semantically similar are expected to occur in similar contexts, and vice-versa.

In the following sections we present approaches for creating the popular $word\ embeddings$. More specifically, in subsection 3.3.1 we examine some frequency based methods, in 3.3.2 we analyze the Word2Vec method, in 3.3.3 the Glove method and finally in 3.3.4 we refer to Contextualized embeddings.

3.3.1 Frequency-based Methods

One-Hot Vectorization: In this method each word is represented as a vector of size $\mathbb{R}^{|V|\times 1}$, where in every dimension we set the value 0, except one that receives the value 1 and the position in which it receives this value corresponds to the index in which it is stored in the dictionary. Let's provide an example to make it clear. Assuming a dictionary $V = \{network, artificial, intelligence, human\}$, with $|V| = 4$, the following one-hot vectors can be produced:

$w^{\text{network}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $w^{\text{artificial}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $w^{\text{intelligence}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $w^{\text{human}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ So, the matrix which for each word contains the corresponding representation has dimensionality $|V| \times |V|$. Consequently, the larger the dimension of the vocabulary gets, the larger the dimensionality of the representations will be.In this way, this matrix is quite sparse and we should also consider that for large dictionaries, we will waste a lot of memory as the dimensions of the matrix become very large. Furthermore, this method does not provide any semantic or relational information between the word vectors.

Count Vectorization: The idea of this method is very simple and similar to the previous one. Let's assume a corpus C of D documents (d_1, d_2 , ..., d_D) and N unique tokens extracted out of the corpus C. The N tokens form the vocabulary. We create the count vector matrix $M \in \mathbb{R}^{N\times D}$,where each element m_{ij} of the matrix M contains the frequency (raw count) of t_i in document d_j. However, we should note that the result is a sparse matrix M with very large dimensions, in this situation too. Moreover, there is not any semantic or relational information between the word vectors. A random visualization of the matrix M is illustrated in Figure 3.1.

TF-IDF Vectorization: This is another method which takes into account not just the occurrence of a word in a single document but in the entire corpus. Common words like "a","I","am",etc tend to appear quite frequently in comparison to the words which are more important to a document. What we want to do is to give more importance to words that appear in only a subset of documents and penalize the words that appear in almost all documents. TF-IDF works by penalising these common words by assigning them lower weights while giving importance to more "important" words. The TF-IDF is the product of two statistic terms, the $term\ frequency$ and $inverse\ document\ frequency$, as the acronym reveals.
For the $term\ frequency\ TF(t, d)$, the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the simplest scheme is $TF(t, d) = f_{t,d}$. Other variations include term frequency adjusted for document length TF$(t, d) = \frac{f_{t,d}}{\sum_{t'\in d} f_{t',d}}$.
The $inverse\ document\ frequency$ is a measure of how common or rare across all documents the word is. It is the logarithmically scaled inverse fraction of the documents that contain the word IDF$(t, D) = \log \frac{N}{|\{d\in D:t\in d\}|}$, where N is the total number of documents in the corpus ($N = |D|$) and $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears.
Then TF-IDF is calculated as $TF - IDF(t, d, D) = TF(t, d)\cdot IDF(t, D)$.

Window based co-occurrence vectorization: In it's simplest form, similarly with the above method,

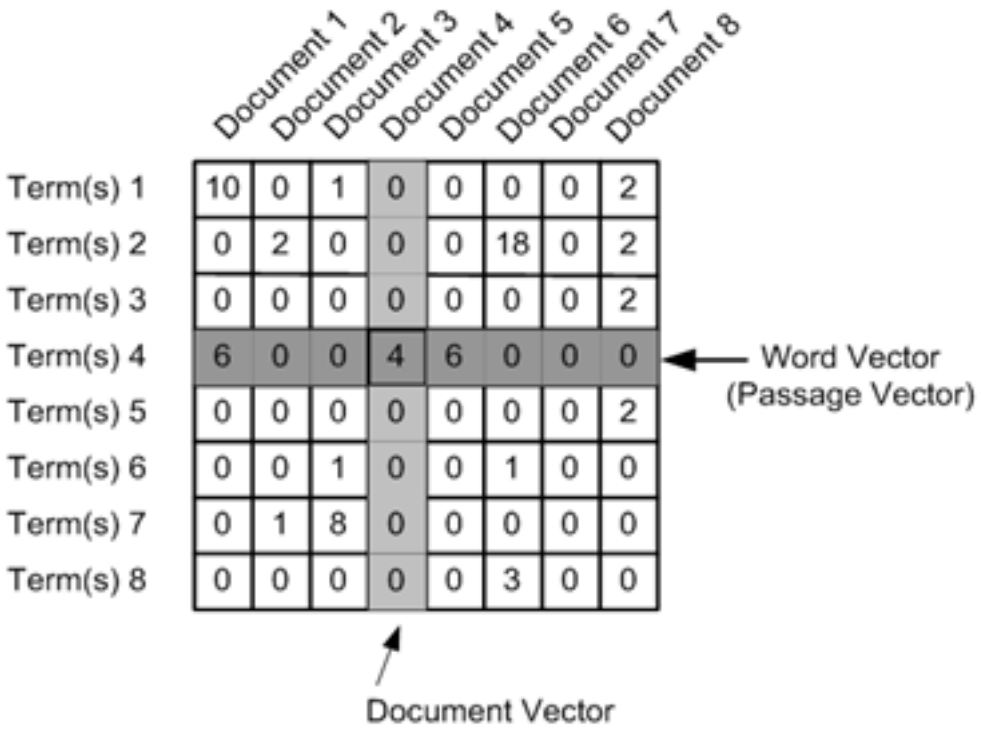


Figure 3.1: Visualization of count vectorization matrix M. (Image source: [16])

we store to the matrix M co-occurrences of the items in a specific window. This simple counting method results in a co-occurrence matrix, where the components of each vector can be interpreted as weights denoting the strength of the relationship between the target and the respective context word.

3.3.2 Iteration-based Methods - Word2Vec

Due to the fact that frequency-based methods are computationally expensive, other approaches for creating word vector representations have been more recently adopted [18, 111, 17]. The key idea of those methods is that "similar" worlds will appear in "similar" contexts and the word vectors of similar words should be also similar. The goal is to create word vector representations, by focusing on predicting the word from its context or predicting the context from the word on every iteration. The most important method adopted in the field of NLP, is the Word2Vec method, proposed by Mickolov [17].

Word2vec is a particularly computationally efficient predictive model for learning word embeddings from raw text. A large corpus of words is used as input to a two-layer neural network, which is trained to construct linguistic contexts of words. Word vectors are positioned in the representation space in such a way that words that share a common context in the corpus are located in close proximity to one another in the representation space. Word2Vec method combines two prediction-based techniques: CBOW(Continuous bag of words) and Skip-Gram , and the corresponding training methods which are the Negative Sampling [112] and the Hierarchical Softmax [113]. As shown in Figure 3.2, the CBOW mechanism tries to predict the "central" word given the context, while the Skip-Gram mechanism tries to predict the context given the "central" word.
Consequently, when the "central" word vector is not able to accurately predict the context of the "central" word, an error is calculated, and through backpropagation, the word vectors are updated. The Negative Sampling mechanism is used during training, to add "negative" samples in order to

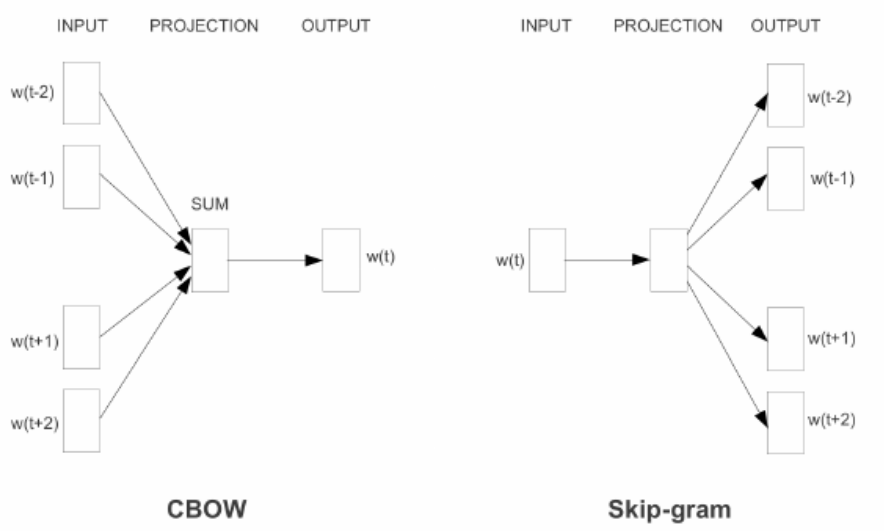


Figure 3.2: CBOW and Skip-Gram mechanisms used in Word2Vec. (Image source: [17])

better train the model and consequently the word embeddings, while the Hierarchical Softmax is used to obtain a better probability distribution over the model's vocabulary.

3.3.3 Glove

GloVe [114], unlike Word2vec, does not rely just on local statistics (local context information of words), but incorporates global statistics (word co-occurrence) to obtain word vectors. The idea of using global statistics to derive semantic relationships between words goes a long way back to latent semantic analysis (LSA) [115]. Before training the actual model, a co-occurrence matrix X is constructed. Given a corpus having V words, the co-occurrence matrix X will be a $|V| \times |V|$ matrix, where the i-th row and j-th column of X, X_{ij} denotes how many times word i has co-occurred with word j. Once X is ready, the word vectors are initialized for each word in the vocabulary and the training is done by minimizing an objective function J, based on the sum of the square errors. This way, the model utilizes the main benefit of count data — the ability to capture global statistics — while simultaneously capturing meaningful linear substructures (like word2vec), achieving the creation of a word vector space that retains important information concerning the meaning of the words.

3.3.4 Contextualized Embeddings

The word vectors described so far are static. However, a word may be used in different sentences having a completely different meaning each time. Consequently, using static word vectors can lead to misunderstandings as static vectors are not always able to represent the various meanings that a word may have. The idea that the context should form the word vector representation and using different word vectors according to the context may lead to better representations, gave rise to the use of contextualized embeddings [116]. Contextualized embeddings come from training models like Bert [4], ElMo [117], etc.

3.4 Language Modeling

In a variety of Natural Language Processing problems, we need to calculate the likelihood of occurrence of a number of words in a particular sequence. Language models are models which are able of calculating the referred probability. Let's denote the occurrence probability of a sequence of M words

$\{w_1, w_2, ..., w_M\}$ with $P(w_1, w_2, ..., w_M)$. The probability $P(w_1, w_2, ..., w_M)$ can be calculated as:

$$P(w_1, \ldots, w_M) = \prod_{i=1}^{M} P(w_i \mid w_1, \ldots, w_{i-1}) \tag{3.1}$$

We want the created model to give a little probability of occurrence to sequences that are syntactically incorrect or are not widely used and vice versa to give higher probability to commonly used or syntactically and grammatically correct sequences.

In the following parts of this section we will discuss and analyse some of the basic language models. It is worth mentioning here that in this section we present traditional models and not the state-of-the-art models for language modeling. However, in Chapter 4, we examine and analyze the state-of-the-art models such as Transformers, BERT and GPT2 in the view of language modeling and natural language generation.

3.4.1 N-Gram Language Models

We can refer to any sequence consisting of n consecutive words using the term "n-gram". We may often come across the terms "bigram" or "trigram" which actually refer to sequences consisting of two and three consecutive words respectively such as "good student" and "deep neural networks". The simplest language model that assigns probabilities to sequences is the n-gram language model. In n-gram language modeling, we have to split the word sequence and predict one word at a time. We can describe this procedure using the chain rule with the following equation:

$$P(w_1, w_2, \ldots, w_M) = P(w_1) P(w_2 \mid w_1) \ldots P(w_M \mid w_1, w_2, \ldots, w_{M-1}) \tag{3.2}$$

As we can see the language model probability $P(w_1, w_2, ..., w_M)$ is a product of word probabilities given the entire history of preceding words. However, using an n-gram model, instead of needing the entire history for computing the probability $P(w_1, w_2, ..., w_M)$, we can approximate this probability using the history of the last n words. This model is called $Markov\ chain$ model and the word probability distribution, limiting the history to n words can be described by the following equation:

$$P(w_M \mid w_1, w_2, \ldots, w_{M-1}) \approx P(w_M \mid w_{M-n}, \ldots, w_{M-2}, w_{M-1}) \tag{3.3}$$

While the n-th order Markov assumption is clearly wrong for any n as sentences can have arbitrarily long dependencies, it still produces strong language modeling results for relatively small values of n, and was the dominant approach for language modeling for many decades.

Let's now try to estimate that probability using maximum likelihood estimation. For the most common case of a bigram and a trigram language model, the estimation of the probabilities $P(w_2|w1)$ and $P(w_3|w_2, w1)$ is computed as:

$$P(w_2 \mid w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)} \tag{3.4}$$

$$P(w_3 \mid w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)} \tag{3.5}$$

More intuitively, for the estimation of the probability $P(w_2|w_1)$ we count how often in the training corpus the word w_1 is followed by the word w_2 as opposed to other words and for the estimation of the probability $P(w_3|w_2, w_1)$ we count how often the sequence of words w_1, w_2 is followed by the word w_3 as opposed to other words.

However, this type of language modeling has two main drawbacks. The first one concerns the fact that

the numerator in equation 3.3 would be zero if the sequence of words w_1, w_2, w_3 has never appeared in the training corpus. Consequently, the probability of having the word w_3 after the sequence of words w_1, w_2 will also be set to zero. To deal with this problem smoothing techniques are used [118], giving some chance of occurrence to those words. The second problem has to do with the occurrence of the sequence w_1, w_2. If those words never appear in sequential order the denominator will be zero. To face this problem, the backoff technique is used [119]. It is also worth mentioning here that the selection of n, increases the computational needs as the model requires much more memory. However, we should also keep in mind that the selection of n is crucial for the window of context which is taken into consideration when the model assigns a probability to a word. So, having in mind this trade-off, n should be carefully chosen.

3.4.2 Window-Based Neural Language Models

In 2003, Bengio proposed a window-based neural language model [18], overcoming "the curse of dimensionality", one of the major problems in the field of NLP. Non-linear neural network models allow conditioning on large context sizes with only a linear increase in the number of parameters, making the computational needs affordable. On the one hand, the model tries to learn a word vector representation space and on the other tries to learn a probability distribution for word sequences. The model takes as input the corresponding word representation vectors of an n-th length word window of previous words. This way, we can encode the words, whose word vectors are noted as $C(w_{t-n+1}), C(w_{t-2})$ and $C(w_{t-1})$ and are called word embeddings ($C(w) \in \mathbb{R}^{d_w}$). The word embeddings are concatenated and fed into a hidden layer, whose output is then provided to a softmax layer. The whole network is illustrated in Figure 3.3.

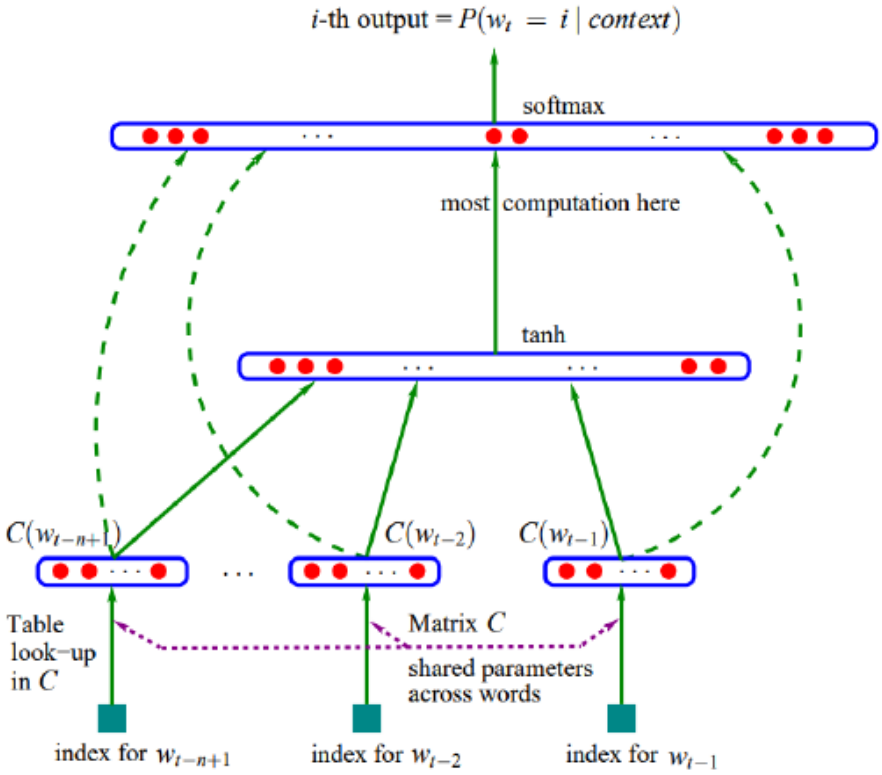


Figure 3.3: Window-based neural network language model proposed by Bengio. (Image source: [18])

More formally, this process can be described by the following equations:

$$\boldsymbol{x} = [C(w_{t-n+1}); C(w_{t-2}); \ldots; C(w_{t-1})] \tag{3.6}$$

$$\hat{y} = \text{softmax}\left(\tanh\left(xW_1 + b_1\right)W_2 + b_2\right) \tag{3.7}$$

where V is the vocabulary, $w_i \in V$,$W_1 \in \mathbb{R}^{n \cdot d_w \times d_{\text{hid}}}$, $b_1 \in \mathbb{R}^{d_{\text{hid}}}$, $W_2 \in \mathbb{R}^{d_{\text{hid}} \times |\mathcal{V}|}$, $b_2 \in \mathbb{R}^{|\mathcal{V}|}$.

More recently, feed-forward neural networks have been replaced with RNNs and LSTMs (described in Sec 2.4) for language modeling. Furthermore, the state-of-the-art models used, producing the best results, are the Transformers models, which are analysed in Chapter 4 in the view of language modeling and natural language generation.

3.5 Emotion Recognition in NLP

Emotion Detection and Recognition from text is a recent field of research that is closely related to Sentiment Analysis. While sentiment analysis in its simplest form aims to detect positive, neutral, or negative feelings from text, emotion recognition aims to detect and recognize more specific feelings based on text, such as anger, disgust, fear, happiness, sadness, and surprise. Undoubtedly, the Internet contains large amounts of text that can be useful for emotional analysis such as product reviews, news articles, stock market analyses, personal blogs/journals, social network websites, forums, fiction excerpts, critiques, or political debates; any place where people discuss and share their opinion freely could be a source. As emotion recognition is really important in understanding human experience and communication, there is growing interest in this domain these days.

So far, three major approaches have been proposed for emotion modeling by the psychology research community [120, 121]. These are the following:

- **The Categorical approach:** According to this approach, there is exists small number of basic emotions being universally recognized. The most commonly used model for emotion recognition is proposed in [122], which involves six basic emotions: happiness, sadness, anger, fear, surprise, and disgust.

- **The Dimensional approach:** According to this approach, the emotional states are not independent but related to each other in a systematic manner. They can be represented using a continuous emotional space of three dimensions: Valence, Arousal and Dominance. Valence expresses how much positive or negative an emotion is, arousal refers to how excited or apathetic an emotion is, while dominance refers to the power of the emotion.

- **The Appraisal-based approach:** This approach can be considered as an extension of the dimensional approach, based on appraisal theory. Appraisal theory is the theory in psychology that emotions are extracted from our evaluations (appraisals) of events that cause specific reactions in different people. To put it simply, our appraisal of a situation causes an emotional response. An example of this is when going on a first date. If the date is perceived as positive, one might feel happiness, joy, giddiness, excitement, and/or anticipation, because he has appraised this event as one that could have positive long-term effects, i.e. starting a new relationship, engagement, or even marriage. On the other hand, if the date is perceived negatively, the emotion caused might involve dejection, sadness, emptiness, or fear.

The approaches for detecting and recognizing emotions from the text, can be distinguished into five categories including keyword-based approaches, rule-based approaches, traditional learning-based approaches, deep learning approaches, and hybrid approaches [123]. A keyword-based approach relies on finding occurrences of keywords in a given text and assigning an emotion label based on the detected keyword [124, 125]. For example, the sentence “Sunny days always make me feel happy” explicitly expresses happiness and includes the emotion keyword “happy”. However, the presence of

an emotion keyword does not always match the expressed emotion. For example, the sentence "Do I look happy to you!" includes the emotion keyword "happy" but does not express that emotion. A rule-based approach is based on the manipulation of knowledge to interpret information in a useful way. First, text preprocessing is performed to the emotion dataset. Afterwards, some emotion rules are selected using linguistic and computational concepts.Finally, they are applied on the emotion dataset to determine the emotion labels [126].

A traditional learning-based approach provides systems the ability to automatically learn and improve from experience. Traditional machine learning algorithms are often used to extract emotion labels from text data [127, 128]. First, text preprocessing is performed on the emotion dataset. The preprocessing steps may include tokenization, stop word removal, lemmatization, and POS tagging. Then useful features are extracted from the text. Given the feature set and the emotion labels, traditional classification models, such as SVM, are trained on the data. Finally, the trained models are used to classify emotions in unseen text. However, deep learning approaches seem to achieve state-of-the-art results on emotion recognition as they can model complex concepts of natural language and understand the implied emotions. The most commonly used method is to use a language model to extract a language representation of the input and then apply classification to predict the corresponding emotion label. Transformers or Recurrent Neural Networks (RNNs) are usually used for extracting language representations, while Dense Neural Networks (DNNs) or Convolutional Neural Networks (CNNs) are used as emotion classifiers [129, 130, 131, 132, 133, 134]. Finally hybrid methods can combine the aforementioned approaches [135, 136, 137]. In [135] the researchers proposed a combination of keyword-based and learning-based approaches.

In this diploma thesis, we use emotion recognition on dialogue systems, in order to not only understand what is being discussed in the conversation but also to understand the implied feelings of the user. In this way, the conversational agent is able to produce more engaging responses. The simplest way to detect and recognize emotions in a dialog context is to use the word representations we introduced before and feed the data to a neural network. However, those representations may be poor as they are based on each word and the performance of the designed models may not be satisfying. More complex models such as Transformers can be used to create the appropriate language representations (contextualized embeddings), and with the addition of a simple neural network over the language model, we can achieve satisfying results. We will take a deeper look into those architectures while studying the models we have implemented, in Chapter 5.

3.6 Summary

In this Chapter we studied the basic principles of the Natural Language Processing (NLP) research field. The language representations methods presented in Section 3.3 are essential for converting natural language into a mathematical form, that can be consumed by the models we implement. Creating contextualized embeddings is an important field of research as they take the context into account, leading to a better representation of the input related to the specific task. Moreover, in Section 3.4 we studied the basic language models, which are also important for creating conversational agents. Understanding the basic ideas behind language modeling is essential as in the next chapter the modern models presented are pre-trained so that they can primarily model language satisfactorily. Finally, we gave a brief description of emotion recognition from text, another critical feature of the empathetic dialogue agents we want to develop.

Chapter 4

Dialog Generation using Generative Models - Theoretical Background

4.1 Introduction

As previously mentioned in Section 1.2, a main category of dialogue systems is the non-task-oriented dialogue systems known as chatbots too. Chatbots provide users with the means to participate in different activities such as a game, entertainment, chitchat rather than focus on a particular task or complete any task in a specific job [56, 33] like most task-oriented dialogue systems do [138, 32]. So the main goal of those systems is to carry on extended conversations mimicking the unstructured conversational or "chats" characteristics of human-human interaction.
Some examples of the very first chatbots are ELIZA [28] and PARRY [29]. Both systems did not use data for learning purposes, but in contrast with the most recent methods, they used a combination of rules and patterns. Data-driven approaches have been proposed to overcome some of the limitations of the hand-built rules. These approaches enabled chatbots to learn from massive amounts of available conversations between humans, such as conversations on chat platforms, on Twitter, or in movie dialogs, summarized by [139], which are available in great quantities and have been shown to resemble natural conversation. As mentioned in Section 1.2, the developed methods are retrieval-based or generation-based [140]. Retrieval-based models obtain response candidates from a pre-built index, rank the candidates and finally choose the response from the top-ranked ones [32, 141], while on the other hand, generation-based methods use natural language generation (NLG) to select the response [142].
In the following sections, we analyze in depth some of the generation-based approaches and we study the most common decoding and evaluation methods used in dialogue generation. More specifically, in Section 4.2 we present the related work, and from Section 4.3 to 4.9 we study in depth traditional and recent generation-models. Afterwards, in Section 4.10 we study the most common decoding methods providing typical examples and finally in Section 4.11 we take a look at the most widely used evaluation metrics.

4.2 Related Work

The basic idea behind generation-based methods is to synthesize a new sentence word by word as a response to the user's request [87] and was inspired by the work in machine translation. In 2001, the researchers of [143] used phrase-based machine translation (SMT) in order to translate a user turn to system response and showed that the SMT method was better-suited for response generation than retrieval-based models on Twitter dataset [143, 144]. More specifically, the phrase-based SMT model considers the strong structural relation between many request-response pairs (e.g., "the soup smells delicious" - "I will bet it looks gorgeous too"), and extracts phrases like "smell-look" and "delicious-gorgeous" from the dataset to translate the request to the response. However, this model could work badly since the responses are often not semantically matched to the requests as in a translation task. For example, it is likely that for a request the responses "having my fruit salad now", "but it is 2 am now" and "which restaurant" are appropriate. Afterwards, it became clear that the task of response

generation was a bit different from machine translation, as in machine translation words or phrases in the source and target sentences tend to align well with each other, but in conversation, a user utterance may share no words or phrases with a coherent response. Later, another technique derived from machine translation called sequence-to-sequence (seq2seq) seemed to work better in response generation tasks [87, 142, 145].

The sequence-to-sequence architecture consists of an encoder model which encodes the user input (request) and represents it as a vector, and a decoder model which decodes the vector (representation of encoded input) and generates a sentence word by word. The encoder and decoder models are usually recurrent neural network (RNN) models. Formally, the encoder-decoder architecture can be used in tasks which can be thought of as requiring the mapping of variable-length input sequences in source language to variable-length sequences in target (e.g.[146]). Subsequently, attention-based mechanisms were developed, which force the encoder to weigh parts of the encoded input more when predicting certain portions of the output during the decoding phase [12, 14, 89]. This mechanism obviates the need for direct input-output alignment, since attention-based models are able to learn input-output correspondences based on loose couplings of input representations and output texts [147, 148]. Moreover, in order to better encode the context of the source sentence bidirectional RNN (BRNN) models were used [149].Those models parse the input not only in forward direction, but in backward too and as a result the amount of input information available to the network is increased.

However, a main problem with the simple sequence-to-sequence response generation is the inability of the architecture to model prior context of the conversation. More specifically, the generated response is based on the previous turn while the huge amount of information derived from previous turns of the dialogue are ignored. To overcome this problem and to incorporate dialogue history in response generation the hierarchical recurrent encoder-decoder (HRED) architecture was adopted, allowing the model to summarize information over multiple prior turns [139, 20, 150, 151]. Later, in order to model complex dependencies between sub-sequences such as found between the utterances in a dialogue, the HRED model was enhanced with stochastic latent variables (VHRED) that span a variable number of time steps. The introduced latent variables seem to facilitate both the generation of meaningful, long and diverse responses and maintain dialogue state [152]. In addition, the need of providing more appropriate and more informative responses lead to the use of external knowledge. The researchers in [153] took advantage of the Memory Network [154] and proposed to condition responses on both dialogue history and external facts. However the architectures mentioned above focus on generating single responses and don't produce continuous responses that cohere across multiple turns. So, some other techniques, such as reinforcement learning [155] and adversarial networks [156, 157], were adopted to learn to choose responses that make the conversation seem more natural.

Later in 2017, a new simple network architecture based on the attention mechanism was proposed, achieving not only much better quality of the generated responses but requiring significantly less time to train too. This architecture is known as the Transformer architecture [3]. The Transformer model uses entirely the attention mechanism to draw global dependencies between input and output. This feature avoids recurrence and also allows for more parallelization. However, the encoder-decoder architecture is still used in the transformer model. Recently, a lot of new state of the art models have been proposed, based on the Transformer architecture. OpenAI-GPT2 [69], Tranfer-Transfo [158], Bert [4], T5 [5] are few of these models that have significant progress in NLP and specifically in dialog generation.

4.3 Vanilla seq2seq model

The simplest version of sequence-to-sequence (seq2seq) models is the Vanilla seq2seq model. As previously mentioned, the idea of the sequence-to-sequence model is to map a variable-length input sequence to a variable-length output sequence. In order to achieve this mapping an encoder model is used for obtaining a large fixed dimensional vector representation of the input sequence, representing the encoded information. Additionally, a decoder model is used for extracting the output sequence by decoding the fixed dimensional vector representation and generating a sentence word by word. In order to fully understand the model's underlying logic, we will go over the illustration in Figure 4.1, in the context of NLP.

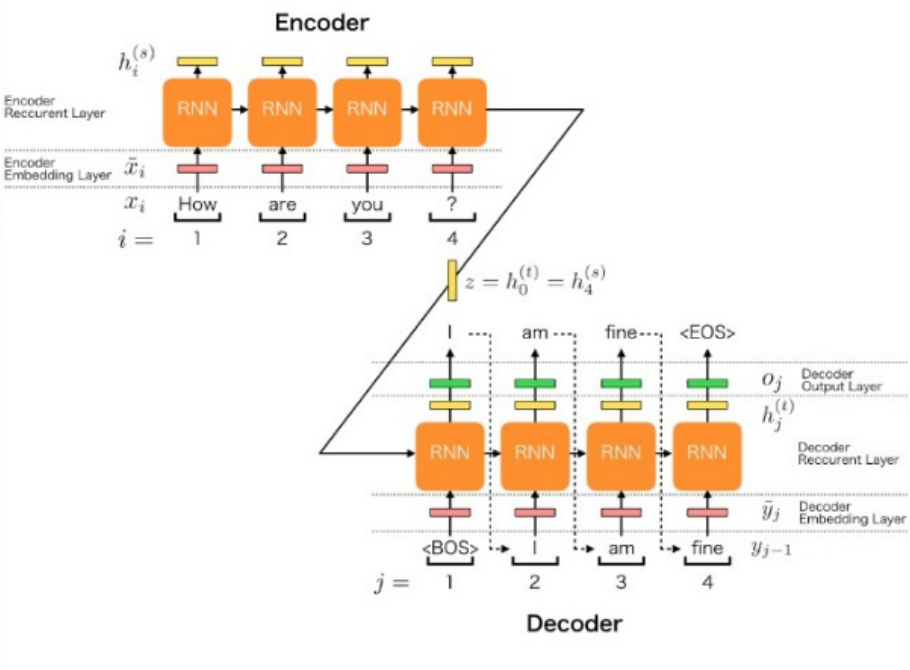


Figure 4.1: The vanilla seq2seq architecture. (Image source: [1])

Let's assume that we have an encoder-decoder architecture which consists of two RNN's, one for the encoder and another one for the decoder. We can further separate the encoder-decoder architecture (in the context of NLP) into five major layers:

- Encoder's Embedding Layer
- Encoder's Recurrent Layer
- Decoder's Embedding Layer
- Decoder's Recurrent Layer
- Decoder's Output Layer

So the encoder consists of two layers: the embedding layer and the recurrent layer, and the decoder consists of three layers: the embedding layer, the recurrent layer, and the output layer. Let's now explain each of these parts.

Encoder's Embedding Layer: As already mentioned in Section 3.3, each word is represented by an embedding vector. The embedding layer of the encoder is used to convert each word of the input sentence to the relevant embedding vector.

Encoder's Recurrent Layer: That is the RNN part as explained in Section 2.4. The encoder's recurrent layer generates the hidden vectors from the embedding vectors. More specifically, at each time step i, the recurrent layer gets as input the embedding vector of the i-th word.

Decoder's Embedding Layer: The decoder's embedding layer converts each word of the decoder's input to the relevant embedding vector.

Decoder's Recurrent Layer: The decoder's recurrent layer generates the hidden vectors from the embedding vectors. More specifically, at each time step j, the recurrent layer gets as input the embedding vector of the $j-1$-th output.

Decoder's Output Layer: The decoder's output layer generates the probability of the j-th word of the output sentence from the hidden vector. More specifically, at each time step j, it receives as input the hidden vector and for each word in the vocabulary produces the probability of selecting this word as the output word in position j.
So, having in mind the above parts we can now explain the whole procedure. Each word of the input sentence is converted through the embedding layer to the relevant embedding vector and then it is fed into the encoder RNN. At each time step i the encoder's recurrent layer outputs a hidden vector which contains the encoded information of the input sequence until the i-th word. Sequentially, the last hidden vector, which is received at the last time step, represents the encoded input sentence. We initialize the decoder's hidden state with this vector in order to generate an answer based on the encoded input sentence and then we give as input to the decoder the "<BOS>" word which is the virtual word representing the beginning of the sentence. The decoder's embedding layer will convert this word to the relevant embedding vector and the embedding vector will be fed as input to the decoder's recurrent layer. Then, a hidden vector is produced and fed to the decoder's output layer. The decoder's output layer produces for each word in the vocabulary the probability of selecting this word as the output word. Supposing that we use the "greedy" decoding method, which is the simplest decoding method from those mentioned in section 4.10, we select as output word the one with the maximum probability. At the next time step we feed the previous output as input and follow the procedure sequentially until a maximum number of words to be generated is reached or until generating the word "<EOS>" which is the virtual word representing the end of the sentence. Another alternative is to use teacher forcing [159], a method for quickly and efficiently training our model. Using this method, instead of feeding as input to the decoder the previous generated word, we feed the target word which is not the predicted but the golden (ground truth) one.

We can now explain the above procedure more formally. Let's consider a conversation consisting of m turns, denoted as $C = \{X_1, X_2, ..., X_m\}$. A turn X_m is a sequence of words $X_m = \{x_{m,1}, x_{m,2}, ..., x_{m,N_m}\}$ where N_m is the length of the m-th turn and each of $x_{i,j}$ is the one-hot vector of the words. For simplicity, we can simplify the notation. So, let's consider one of those turns as the input sequence, denoted with $X = \{x_1, x_2, , ..., x_n\}$ and the next one as the output sequence, denoted with $Y = \{y_1, y_2, ..., y_l\}$, where each of the x_i and y_j are one-hot vectors. Let's also assume as $\mathbf{z}$ the fixed-size vector which is produced by the encoder, containing the information of the encoded input sequence. We can describe the process of generating Y with the probability of generating the j-th element of the output sequence y_j, given the output sequence until the j-th element and the input sequence X, as follows:

$$P_\theta\left(y_j \mid Y_{<j}, X\right) = \Upsilon\left(h_j^{(dec)}, y_j\right) \tag{4.1}$$

$$h_j^{(dec)} = \Psi\left(h_{j-1}^{(dec)}, y_{j-1}\right) \tag{4.2}$$

where Ψ is the function to generate the hidden vectors of the decoder $h_j^{(dec)}$, and Υ is the function to calculate the generative probability of the one-hot vector y_j. Both of them are defined later. When

j=1, $h_{j-1}^{(dec)}$ or $h_0^{(dec)}$ is **z**, and y_{j-1} or y_0 is the one-hot vector of "<BOS>".

Let's also assume that H is the size of the hidden vector, D is the size of the embedding vector, $\overline{x}_i$ is the embedding vector of i-th word in the input sentence, $E^{(enc)}$ is the embedding matrix of the encoder, $h_i^{(enc)}$ is the i-th hidden vector of the encoder, $\overline{y}_j$ is the embedding vector of j-th word in the output sentence, $E^{(dec)}$ is the embedding matrix of the decoder and $h_j^{(dec)}$ is the j-th hidden vector of the decoder. A summarized table of the notations assumed is shown in 4.1.

Table 4.1: Summary of Mathematical Notation

Symbol	*Definition*
C	conversation consisting of m dialog turns
X or X_k	the input sequence (k-th turn)
Y or Y_{k+1}	the output sequence (k+1-th turn)
z	the fixed length representation provided by the encoder
H	the size of the hidden vector
D	the size of the embedding vector
x_i	the one-hot vector of i-th word in the input sequence
$\overline{x}_i$	the embedding vector of the i-th word in the input sequence
y_j	the one-hot vector of j-th word in the output sequence
$\overline{y}_j$	the embedding vector of the j-th word in the output sequence
$E^{(enc)}$	the embedding matrix of the encoder
$E^{(dec)}$	the embedding matrix of the decoder
$h_i^{(enc)}$	the i-th hidden vector of the encoder
$h_j^{(dec)}$	the j-th hidden vector of the decoder

According to the notation introduced each embedding vector for the encoder's embedding layer is calculated by the following equation:

$$\overline{x}_i = E^{(enc)} x_i \tag{4.3}$$

where $E^{(enc)} \in \mathbb{R}^{D \times |\mathcal{V}^{(enc)}|}$ is the embedding matrix of the encoder and $\mathcal{V}^{(enc)}$ the vocabulary of the inputs.

Similarly, each embedding vector for the decoder's embedding layer is calculated by the following equation:

$$\overline{y}_j = E^{(dec)} y_{j-1} \tag{4.4}$$

where $E^{(dec)} \in \mathbb{R}^{D \times |\mathcal{V}^{(dec)}|}$ is the embedding matrix of the decoder and $\mathcal{V}^{(dec)}$ is the vocabulary of the outputs.

Let's assume now that we use uni-directional RNNs of one layer with $tanh$ as activation function, for the encoder and the decoder. The encoder's hidden state is described by the following equation (defining Ψ):

$$h_i^{(enc)} = \tanh\left(W_{hh}^{(enc)} h_{i-1}^{(enc)} + W_{xh}^{(enc)} \overline{x}_i + b^{(enc)}\right) \tag{4.5}$$

where $W_{hh}^{(enc)} \in \mathbb{R}^{H \times H}$, $W_{xh}^{(enc)} \in \mathbb{R}^{H \times D}$ and $b^{(enc)} \in \mathbb{R}^H$ are matrices to be learned.

Similarly, the decoder's hidden state is described by the following equation:

$$h_j^{(dec)} = \tanh\left(W_{hh}^{(dec)} h_{j-1}^{(dec)} + W_{xh}^{(dec)} \overline{y}_j + b^{(dec)}\right) \tag{4.6}$$

where $W_{hh}^{(dec)} \in \mathbb{R}^{H \times H}$, $W_{xh}^{(dec)} \in \mathbb{R}^{H \times D}$ and $b^{(dec)} \in \mathbb{R}^{H}$ are matrices to be learned.

We must also use the encoder's hidden vector of the last position as the decoder's hidden vector of first position (initialise the hidden state of the decoder) as following:

$$h_0^{(dec)} = \mathbf{z} = h_n^{(enc)} \tag{4.7}$$

Finally, the decoder's output layer generates the probability of the j-th word of the output sentence from the hidden vector. Assuming that p_j is the probability of generating the one-hot vector y_j of the j-th word, p_j is calculated with the use of the following equation (defining Υ):

$$p_j = P_\theta\left(y_j \mid Y_{<j}, X\right) = \text{softmax}\left(W^{(o)} h_j^{(dec)} + b^{(o)}\right) \cdot y_j \tag{4.8}$$

where $W^{(o)} \in \mathbb{R}^{|\mathcal{V}^{(dec)}| \times H}$ and $b^{(o)} \in \mathbb{R}^{|\mathcal{V}^{(dec)}|}$ are also matrices to be learned.

4.4 Vanilla seq2seq model with attention

The main drawback of the sequence-to-sequence model was that it was almost incapable of representing long-term sequences with the use of a single fixed-length context vector. Thus, to overcome that problem, the attention mechanism, which was already known in the field of image recognition [160, 161, 162], was applied in the field of NLP too [84, 87, 12, 14]. The attention mechanism instead of relying only on the hidden vector of the decoder, forces the model to learn to focus (to attend) on specific parts of the input sequence when decoding. An example in a translation task is shown in Figure 4.2, where the model attends more specific parts of the input sequence when decoding the French word "la".

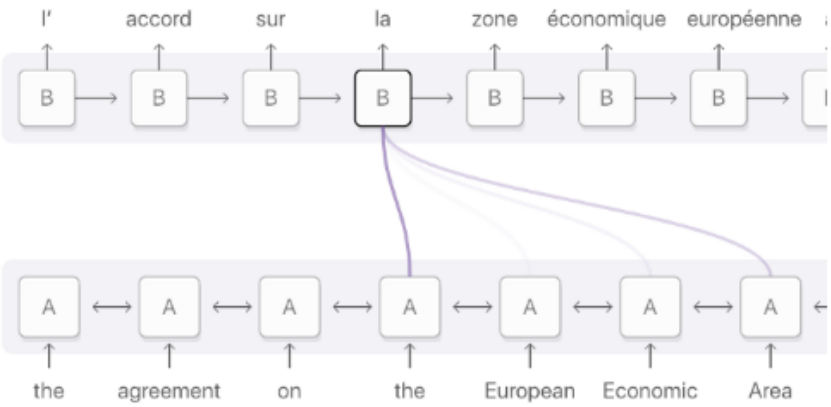


Figure 4.2: Alignment for the French word 'la' is distributed across the input sequence but mainly on these 4 words: 'the', 'European', 'Economic' and 'Area'. Darker purple indicates better attention scores. Image source: [19]

From the attention mechanisms described in subsection 2.4.2 we analyze and explain the one proposed by [12], expanding the sequence-to-sequence model. For placing different focus on different words, attention assigns each word with an attention score (also knows as alignment score). The attention scores define how much each source hidden state must be considered for each output and are calculated

by a score function using the encoder's and decoder's hidden states. The choice of the score function varies, but is usually one of those in equation 4.9.

$$f\left(h_{j-1}^{(dec)}, h_i^{(enc)}\right) = \begin{cases} {h_{j-1}^{(dec)}}^T h_i^{(enc)} & \text{dot} \\ {h_{j-1}^{(dec)}}^T W_a h_i^{(enc)} & \text{general} \\ v_a^T \tanh\left(W_a \left[h_{j-1}^{(dec)}; h_i^{(enc)}\right]\right) & \text{concat} \end{cases} \tag{4.9}$$

where $h_{j-1}^{(dec)}$ and $h_i^{(enc)}$ are the decoder's and encoder's hidden states we defined before, and v_a, W_a are both weight matrices to be learned in the alignment model (as described in equation 2.52). Those scores are then normalized by passing through a softmax layer and afterwards each of the encoder's hidden state is multiplied with the relevant attention score obtaining an alignment vector (one for each hidden state). Then, the alignment vectors are summed up, calculating the context vector. Sequentially, the context vector is concatenated with the decoder's input and passed through the decoder. The above process is presented in Figure 4.3 and the score (alignment) function used is the dot-product.

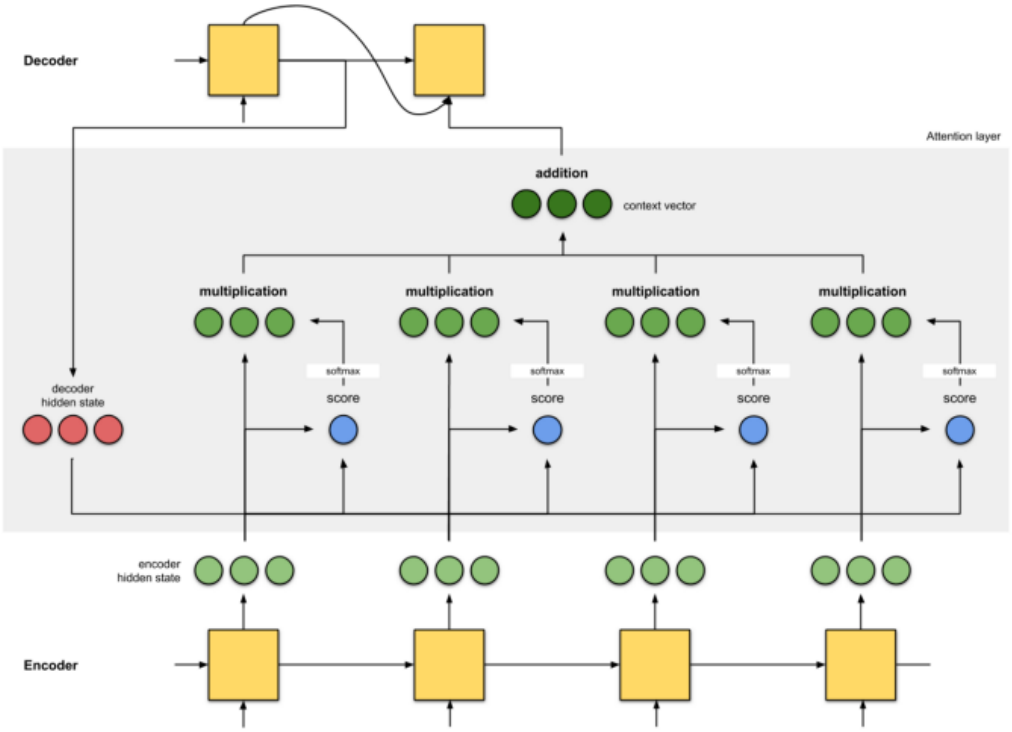


Figure 4.3: A Sequence to sequence model with Bahdanau attention. Image source:[2]

4.5 HRED model

A main problem with the simple sequence-to-sequence architectures is the inability of modeling the prior context of the conversation. The generated responses are based only on the previous turn of the conversation, while a huge amount of information derived from previous turns of the dialogue is ignored. The HRED (Hierarchical Encoder Decoder) model, which is an extension of the simpler Encoder-Decoder architecture, attempts to overcome the limitation of generating an output based only on the latest input received. In the case of conversational data, the HRED model considers each

conversation as a sequence of turns (utterances), and each turn (utterance) as a sequence of words. The HRED model consists of three different modules:

- the Encoder model
- the Context (Session) Encoder module
- the Decoder module

Below we analyse each one of the referenced modules.

Encoder: The encoder module handles the data at the lowest hierarchical level. It is responsible for encoding the turn (utterance) into a fixed length vector. So, by processing each utterance word by word, each utterance is mapped to an utterance vector which is the hidden state obtained after the last token of the utterance has been processed.

Context Encoder: The context encoder module handles the data at the highest hierarchical level. It is responsible for keeping the context of the conversation, thus it keeps track of the past utterances by processing iteratively each utterance vector and updating its hidden state after every utterance. By doing so, the context vector, which is the hidden state obtained after the last utterance is processed, represents the entire conversation up to the last turn received.

Decoder: Similarly with the simple encoder-decoder architecture, the decoder performs the next utterance prediction. The hidden state of the decoder is initialised with the context vector and then it produces a probability distribution over the tokens in the next utterance.

The HRED model is represented in Figure 4.4.

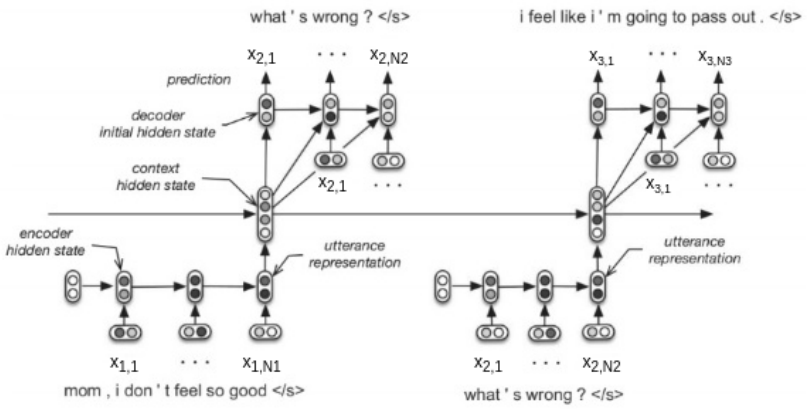


Figure 4.4: The computational graph of the HRED architecture for a dialogue composed of three turns. Each utterance is encoded into a dense vector and then mapped into the dialogue context, which is used to decode (generate) the tokens in the next utterance. The encoder RNN encodes the tokens appearing within the utterance, and the context RNN encodes the temporal structure of the utterances appearing so far in the dialogue, allowing information and gradients to flow over longer time spans. The decoder predicts one token at a time using a RNN. Image source: [20]

We can now explain the above procedure more formally, using the notation we introduced before. Let's assume that we use GRUs as encoder, context encoder and decoder models. The encoder for each turn X_m in the conversation C, produces a fixed length representation vector by sequentially

updating its hidden state. We denote with $h_i^{(enc)}$ the i-th hidden vector of the encoder, with $h_i^{(dec)}$ the i-th hidden vector of the decoder and with $h_i^{(cont)}$ the i-th hidden vector of the context encoder.

So the hidden state of the encoder is updated according to the following equation:

$$h_i^{(enc)} = GRU_{enc}\left(h_{i-1}^{(enc)}, \bar{x}_{m,i}\right), i = 1, \ldots, N_m \tag{4.10}$$

where $h_0^{(enc)} = 0$ the null vector, N_m is the length of the m-th turn and $x_{m,i}$ is the one-hot vector of i-th word in m-th turn as we mentioned before. The GRU_{enc} function is that one described in Section 2.4.

In summary, the encoder maps a turn to a fixed-length vector. Therefore, the obtained representation from the m-th turn $q_m \equiv h_{N_m}^{(enc)}$ is a general, non-contextual representation of the turn m. The computation of the $q_1, q_2, ..., q_m$ can be performed in parallel, thus lowering the computational cost. Afterwards, the context encoder takes as input the sequence of representations $q_1, q_2, ..., q_m$. Assuming that a GRU model is used, we can describe the process with the following equation:

$$h_i^{(cont)} = GRU_{cont}\left(h_{i-1}^{(cont)}, q_i\right), \quad i = 1, \ldots, m \tag{4.11}$$

where $h_i^{(cont)} \in \mathbb{R}^{H_{cont}}$ is the context-level recurrent state, H_{cont} is its dimensionality and $h_0^{(cont)} = 0$. The context-level recurrent state $h_i^{(cont)}$ summarizes the turns that have been processed up to position i. It is worth mentioning, that each $h_i^{(cont)}$ bears a particularly powerful characteristic: it is sensitive to the order of previous turns and, as such, it can potentially encode order-dependent reformulation patterns such as generalization or specification of the previous turns [163]. Additionally, it inherits from the representation vectors q_m the sensitivity to the order of words in the turns. Finally, the decoder model is responsible for predicting the next turn Y_m (output turn), given the previous turns $X_{1:m-1}$ (input turns), to estimate the probability shown in the next equation:

$$P\left(Y_m \mid X_{1:m-1}\right) = \prod_{n=1}^{N_m} P\left(y_{m,n} \mid y_{m,1:n-1}, X_{1:m-1}\right) \tag{4.12}$$

The desired conditioning on previous queries is obtained by initializing the hidden state of the decoder with a non-linear transformation of $h_{m-1}^{(cont)}$. So in this way, the information from the previous turns is transferred to the decoder. The decoder's hidden state is described by the following equation:

$$h_i^{(dec)} = GRU_{dec}\left(h_{i-1}^{(cont)}, \bar{x}_{m,i}\right), \quad i = 1, \ldots, N_m \tag{4.13}$$

Each state $h_{i-1}^{(dec)}$ is then used for computing the probability of generating the next word of the m-th turn $x_{m,i}$. This probability given the previous words and turns is expressed as:

$$P\left(y_{m,n} = v \mid y_{m,1:n-1}, X_{1:m-1}\right) = \text{softmax}\left(W^{(o)} h_{n-1}^{(dec)} + b^{(o)}\right) \cdot y_{m,n} \tag{4.14}$$

where $W^{(o)} \in \mathbb{R}^{|\mathcal{V}^{(dec)}| \times H}$ and $b^{(o)} \in \mathbb{R}^{|\mathcal{V}^{(dec)}|}$ are also matrices to be learned.

4.6 Transformer Encoder Decoder model

In 2017, the Transformer network architecture was proposed by [3], achieving not only much better quality of the generated responses but requiring significantly less time to train too. The Transformer model relies entirely on self attention mechanism to compute representations of its input and output without using sequence aligned RNNs or convolution. The model is based on the encoder-decoder

structure, having an encoder and a decoder stack. The encoder maps an input sequence X to a sequence of continuous representations $\mathbf{z}$. Given $\mathbf{z}$, the decoder then generates an output sequence Y, generating one element at a time. At each step the model is auto-regressive, consuming the previously generated words as additional input when generating the next [164].

Encoder Stack: The encoder stack consists of $N = 6$ identical layers, each one having two sub-layers. The first sub-layer is a multi-head self attention mechanism, while the second is a simple position-wise fully connected feed forward network. Residual connection [165] is also employed around each of the sub-layers followed by layer normalization [166]. So, the output of each sub-layer is LayerNorm$(x + \text{Sublayer}(x))$, where Sublayer(x) is the function implemented by the sub-layer itself (the sub-layer's structure, the layer normalization and the position-wise feed forward network are explained below).

Decoder Stack: The decoder stack also consists of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, each decoder layer has also a third sub-layer, which performs multi-head attention over the outputs of the encoder stack. Similarly to the encoder stack, residual connections are employed around each of the sub-layers followed by layer normalization. The self-attention sub-layer in each decoder layer is also modified to prevent positions from attending to subsequent positions. The masking which is applied, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i (masked self-attention).
A high-level illustration of the transformer architecture in a machine translation task is shown in Figure 4.5.

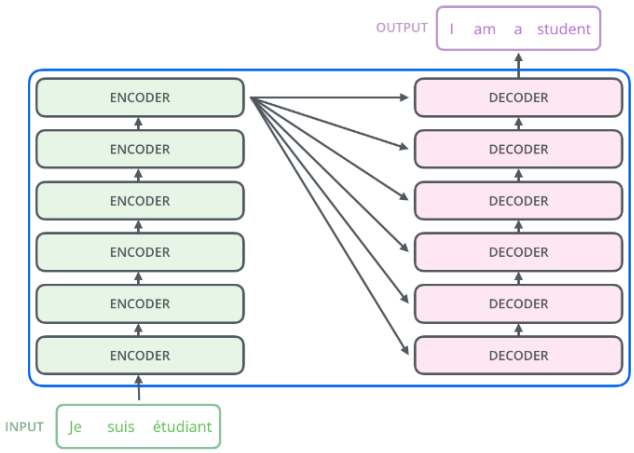


Figure 4.5: A high level illustration of the transformer. The encoder stack consists of 6 identical (encoder) layers and the decoder stack of 6 identical (decoder) layers too. Image source: [21]

In Figure 4.7 the encoder layer's architecture is illustrated. Similarly to other sequence transduction models, learned embeddings are used in order to convert input and output to vectors of dimension d_{model}. The d_{model} dimension is set to 512 [3]. The input embeddings are illustrated with dark green color. Since the model contains no recurrence and no convolution, there is no use of the order of the input sentences. To deal with that issue, positional "encodings" (embeddings) are used to inject some information about the relative or absolute position of the tokens in the sequence. There are many choices of positional "encodings" learned and fixed [167], however in the original model the following are used:

$$PE_{(pos,i)} = \begin{cases} \sin\left(pos/10000^{i/d_{\text{model}}}\right) & \text{, i is even} \\ \cos\left(pos/10000^{i-1/d_{\text{model}}}\right) & \text{, i is odd} \end{cases} \tag{4.15}$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. Learned embeddings can also be used instead [3, 167]. An example of the position encoding of 20 words is shown in Figure 4.6. The positional embeddings are then summed with the input embeddings resulting in the "light green" embeddings (Figure 4.7), which are then passed through the self-attention layer.

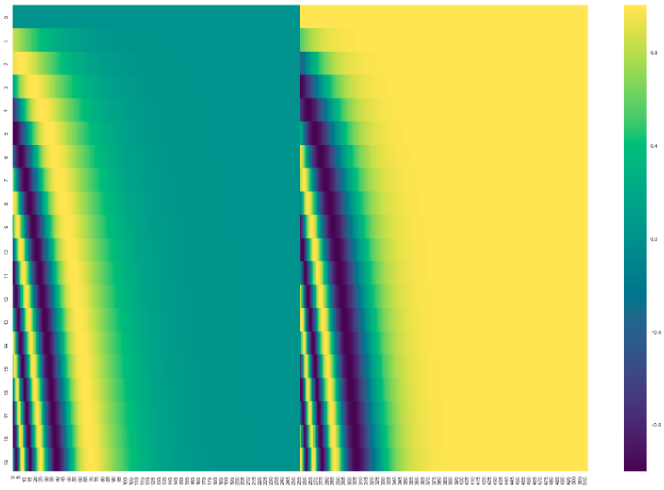


Figure 4.6: A real example of positional encoding for 20 words (rows) with an embedding size of 512 (columns). Each row corresponds the positional encoding of a vector. Each row contains 512 values – each with a value between 1 and -1. We've color-coded them so the pattern is visible.we can notice that it appears split in half down the center. That's because the values of the left half are generated by one function (which uses sine), and the right half is generated by another function (which uses cosine). They're then concatenated to form each of the positional encoding vectors. Image source: [21]

As already mentioned each encoder layer uses multi-head self attention. The original proposed model uses 8 heads (parallel attention layers) with each one having d_k=d_v=$d_{model}/h = 64$ (see multi-head attention in subsection 2.4.2). The 8 sets of Q, K, V weight matrices are calculated by multiplying the input embedding with the corresponding learned weight matrices W^Q, W^K, W^V (see equation 2.56) for each one of the 8 sets. Then a set of 8 $\mathbf{z}$ matrices is produced, which are then concatenated and multiplied by a weight matrix W^O (see equation 2.56) to extract a final representation $\mathbf{z}$. This procedure happens for all inputs in parallel. Then, the residual connection (adding to the $\mathbf{z}$ vector the corresponding input embedding as both have same dimension equal to 512) and layer normalization is applied as shown in Figure 4.7. In the next step, the outputs of the "Add & Normalize" layer are passed through the position-wise feed forward network, which is applied to each position separately and identically. This consists of two linear transformations with a $ReLU$ activation in between as descibed in the following equation:

$$\text{FFN}(x) = \max\left(0, xW_1 + b_1\right) W_2 + b_2 \tag{4.16}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$. Afterwards, the outputs of the feed forward layer and the outputs of the previous sub-layer are passed through the second "Add & Normalize" layer. The process is repeated for each one of the 6 layers, however only the first layer uses the input and positional embeddings, while the following layers use the ouput of the previous ones.

Having covered the encoder-side we will now focus on the decoder-side. Let's assume a transformer model of 2 stacked encoders and decoders too. An illustration of this model is shown in Figure 4.8.

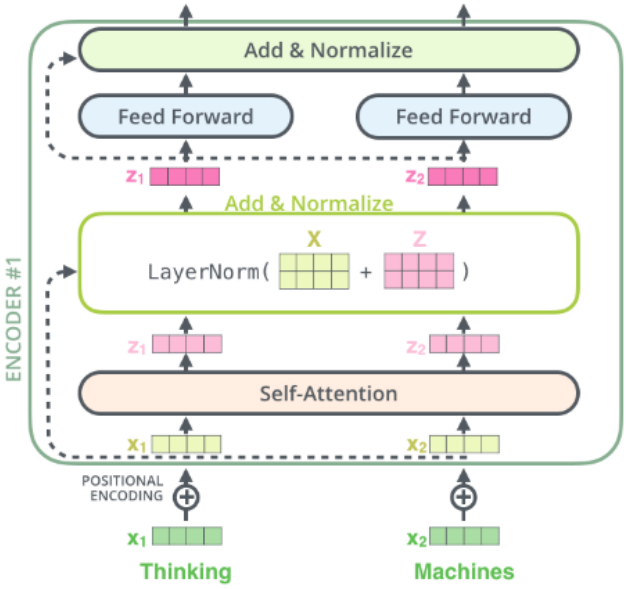


Figure 4.7: An illustration of the first encoder layer in the encoder stack. Image source: [21]

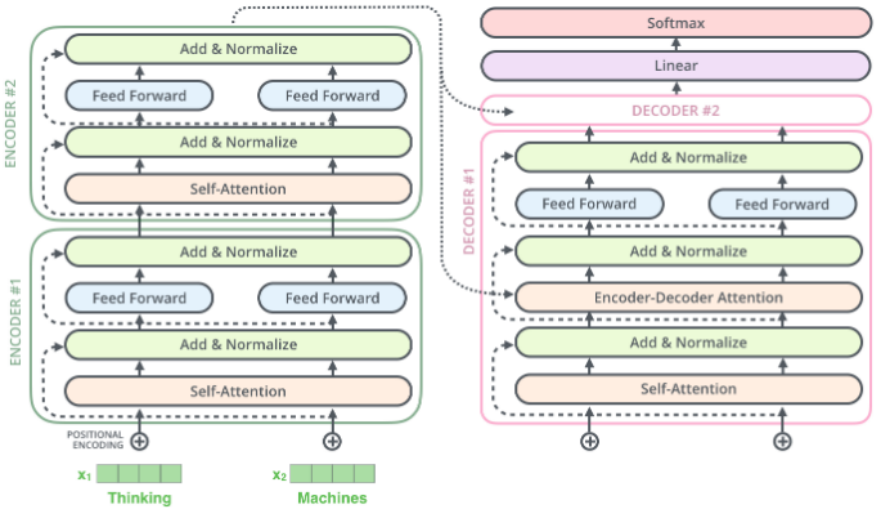


Figure 4.8: An illustration of transformer with 2 stacked encoders and decoders. Image source: [21]

On the decoder-side, the output embeddings (the embedding of the target outputs shifted right) summed with the corresponding positional encodings (shown in Figure 4.9) are given as input to the first sub-layer. So, similarly to seq2seq models the output of each step is fed to the bottom decoder in the next time step (after adding the positional embeddings).Afterwards, the same processas in the encoder layers, is followed for the first sub-layer. The only difference is that the self attention applied is masked, in order to ensure that the predictions for position i can depend only on the known outputs at positions less than i. As already has been mentioned, a third sub-layer has been added, named as "Encoder-Decoder Attention", which helps the decoder focus on appropriate places in the output of the encoder stack (which carry information from the input sequence). So, the output of the top encoder layer is transformed into a set of attention vectors K and V and those are used in the "Encoder-Decoder Attention" sub-layer, along with the outputs of the previous sub-layer. After receiving the outputs of the last decoder layer those are passed from a usual learned linear transformation and a softmax function to convert the decoder output to predicted next-token probabilities. In the original model [3] the same weight matrix between the two embedding layers and the pre-softmax linear transformation is used,

similar to [168].

Finally, a fully illustration of the transformer model is shown in Figure 4.9.

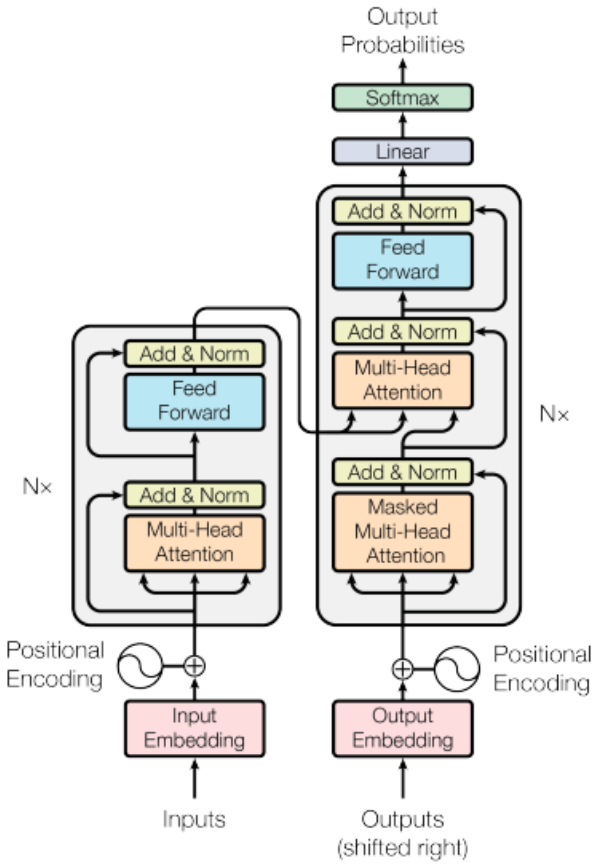


Figure 4.9: The Transformer-model architecture. Image source: [3]

4.7 Bert model

The year 2018 has been an inflection point for the NLP community, with the release of ELMo by Allen AI [117], Open-GPT by OpenAI [169], and BERT by Google [4]. Since 2018, a lot of attention has been given to Transfer Learning techniques as they are widely used. Many researchers have been able of conducting experiments, achieving much better results with less effort, time, and data just by using pre-trained models and then fine tuning them in specific tasks. So,the release of BERT model is an event described as marking the beginning of a new era in NLP.

The BERT (Bidirectional Encoder Representations from Transformers) [4] model is based on a number of clever ideas that have appeared in NLP recently, including but not limited to Semi-supervised Sequence Learning [170], ELMo [117], ULMFiT [171], the OpenAI transformer [169] and the Vaswani Transformer [3]. BERT can be used in a wide variety of language tasks, with only adding a small layer to the core model, such as classification, question-answering, named entity recognition tasks, etc.

One of the main reasons for the good performance of BERT on different NLP tasks was the pre-training on two unsupervised tasks, unlike traditional left-to-right or right-to-left language models [117, 169]. This way, the model is enabled to "understand" the patterns of the language.

The first task in which the model is pre-trained is called "masked language modeling" (MLM), which is also referred to as a "Cloze" task in the literature [172]. In this task, the 15% of all WordPiece tokens [173] of each sequence is masked randomly (using the $[MASK]$ token) and the final hidden vectors corresponding to the masked tokens are fed into an output softmax over the vocabulary, as in a standard Language Modeling (LM). It is worth mentioning here, that in contrast to denoising auto-encoders [174], only the masked words are predicted rather than reconstructing the entire input.

The second task is called "Next Sentence Prediction" (NSP). Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI - the task of determining whether a "hypothesis" is true (entailment), false (contradiction), or undetermined (neutral) given a "premise") are based on understanding the relationship between two sentences, which is not directly captured by language modeling. Consequently, in order to train a model that understands sentence relationships, the model is pre-trained for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as $IsNext$), and 50% of the time it is a random sentence from the corpus (labeled as $NotNext$). Despite its simplicity, the pre-training towards this task is very beneficial to both QA and NLI. For the pre-training procedure the BooksCorpus (800M words) [175] and English Wikipedia texts (2,500M words) are used. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks (MNLI - a task similar with NLI with more genres, NER - a task for locating and classifying named entities mentioned in unstructured text into pre-defined categories , SQuAD - a task for reading comprehension). An illustration of the overall pre-training and finetuning procedures is shown in Figure 4.10.

The BERT's model architecture is based on the original Transformer implementation that we already analysed in Section 4.6. BERT is basically a multi-layer bidirectional Transformer encoder, the multi-head self-attention mechanism (described in Section 4.6) to attend the input sequence in two directions as shown in Figure 4.11.

In the original paper two BERT models are presented. The $BERT_{BASE}$ has 12 layers in the Encoder stack, while $BERT_{LARGE}$ has 24 layers in the Encoder stack. BERT architectures (BASE and LARGE) also have larger feed forward-networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the Transformer architecture suggested in the original

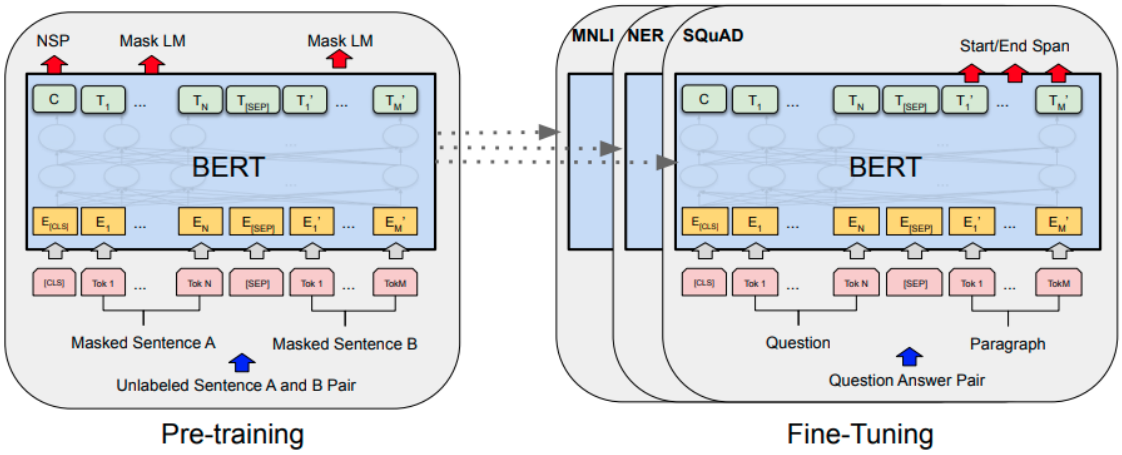


Figure 4.10: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different downstream tasks. During fine-tuning, all parameters are fine-tuned. $[CLS]$ is a special symbol added in front of every input example, and $[SEP]$ is a special separator token (e.g, separating questions/answers). Image source: [4]

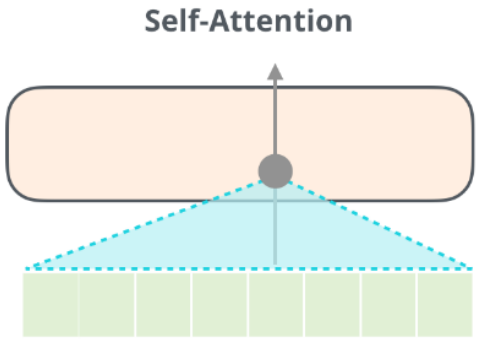


Figure 4.11: Self attention used in BERT model. Image source: [22]

paper (6 encoder layers, 512 hidden units, and 8 attention heads) [3]. Apart from extracting contextual language embeddings, the BERT model can also be used for various tasks such as classification, question answering or entity recognition tasks by simply adding a small network at the top of the model as head. A number of ways to use BERT in different tasks are shown in Figure 4.12.

More specifically, in order to make BERT able to handle a variety of tasks, the input representation should be able to unambiguously represent both a single sentence and a pair of sentences (e.g.,<Question, Answer>) in one token sequence. We should note here that a "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. First of all, we tokenize the input sequence. In the original paper [4], the WordPiece [173] tokenizer is used with a 30000 token vocabulary. The first token of every sequence is always a special classification token ($[CLS]$). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence and two ways are used to differentiate the two sentences. First, the sentences are separated by a special token ($[SEP]$) and second a learned embedding is added to each token indicating whether it belongs to sentence A or B. As shown in Figure 4.10,the input embedding is denoted as E, the final hidden vector of the special [CLS] token as $C \in \mathbb{R}^H$, and the final hidden vector for the i-th

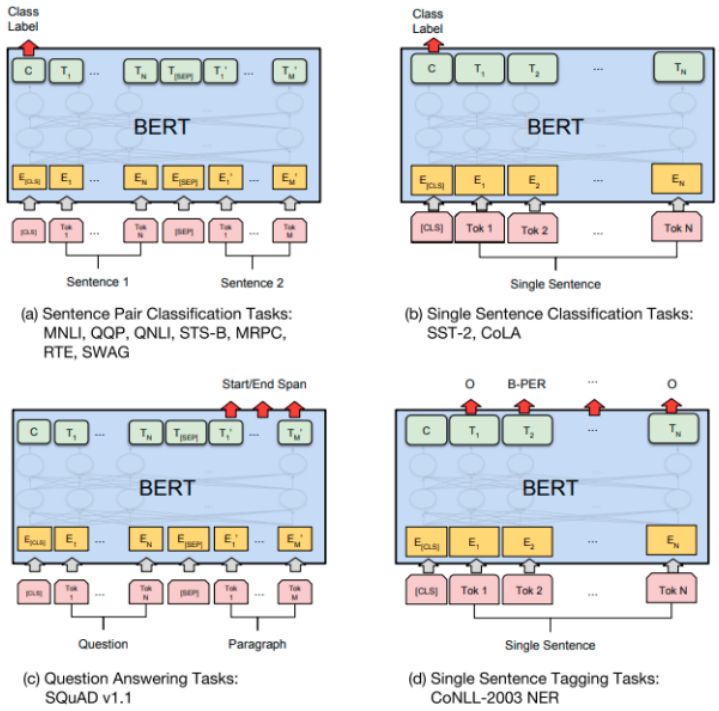


Figure 4.12: The BERT model used in various tasks. Image source: [23]

input token as $T_i \in \mathbb{R}^H$. For a given token, its input representation is constructed by summing the corresponding token, segment (first or second sentence), and position embeddings. A visualization of this construction can be seen in Figure 4.13. Finally, we should also note that all input sequences should be padded or truncated to a specific length to be fed into the model (512 tokens). An attention mask must also be given in order to neglect the information of the padding tokens.

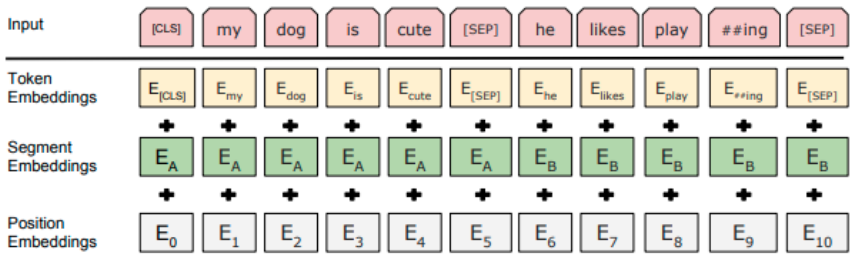


Figure 4.13: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. Image source: [4]

4.8 GPT-2 model

The OpenAI GPT-2 [69] exhibits impressive ability of writing coherent and passionate essays that exceed what we anticipated current language models are able to produce. The model's architecture is not a particularly novel architecture as it is very similar to the decoder-only transformer. So, the GPT-2 model is build using transformer decoder blocks. Like traditional language models, the model outputs one token at a time. It is an "auto-regression" model, as after each token (output) is produced, it is added to the sequence of inputs and the new sequence becomes the input to the model in the next

step. This idea has been already seen in the RNNs and many transformers such as the TransformerXL [176] and XLNet [177] follow the same concept.

The GPT-2 model is released in 4 versions, the small, the medium , the large and the extra-large. Each version differs in dimensionality having 768, 1024, 1280 and 1600 respectively. Depending on the model's dimensionality, the relative word embeddings dimensions are used. Each version of the model differs in the number of decoder layers used in the decoder stack too, having 12, 24, 36 and 48 decoder layers respectively. We should note here that in contrast to the BERT versions described in Section 4.7, the GPT-2's versions have more parameters with 117, 345, 774 and 1558 million parameters respectively [178, 179]. The model is trained without any explicit supervision on a large dataset, called WebText, containing slightly over 8 million documents for a total of 40 GB of text, trying to predict the next word.

We will now focus on the decoder layer of the GPT-2 model. The decoder layer consists of two sub-layers, the masked multi-head self-attention layer and the feed forward layer. One key difference in the self-attention layer with the BERT model, is that it masks future tokens – not by changing the word to $[MASK]$ like BERT, but by interfering in the self-attention calculation blocking information from tokens that are to the right of the position being calculated. This attention mechanism is called masked self-attention, and we have already described it in Section 4.6. An illustration of the simple self-attention (used in BERT) and the masked self-attention mechanism is shown in Figure 4.14.

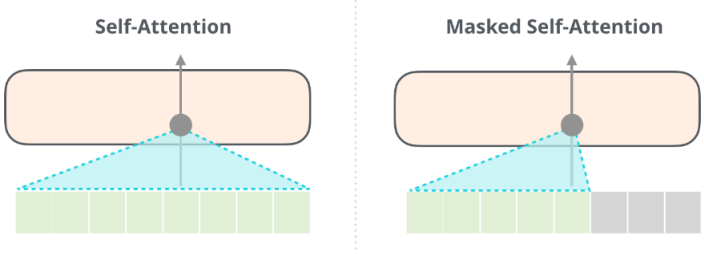


Figure 4.14: Self-attention (used in BERT) compared with masked self-attention (used in GPT2). While in simple self-attention the model attends the entire input sequence, in masked self-attention the model attends the words of the sequence until the current time-step. Image source: [22]

So, similarly to [180], the researchers used for the GPT2 the architecture shown in Figure 4.15. A similar architecture was also examined in [181] to create a language model that predicts one letter/character at a time. We should also note here that the GPT-2 model uses multi-head attention with 12, 24, 36

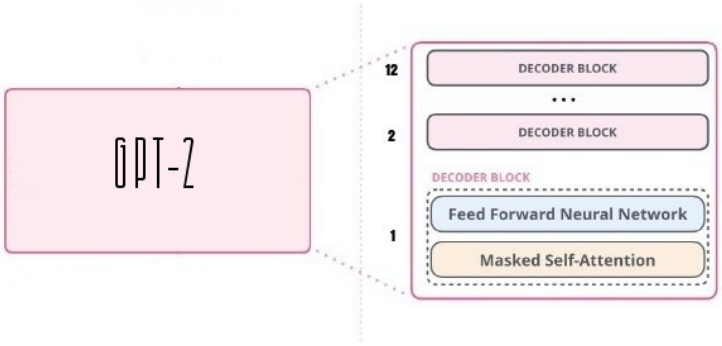


Figure 4.15: A simple illustration of the GPT-2 model. All the decoder layers are identical, each one consisting of a masked self-attention layer and a feed-forward neural network.

and 48 layers in the small, medium, large and extra-large versions respectively.

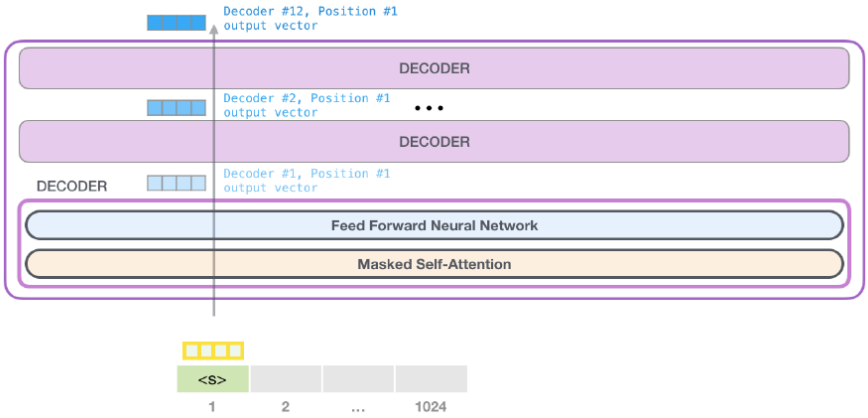


Figure 4.16: A simple illustration of producing the output vector using the "small" GPT-2 model. Image source: [22]

The GPT-2 model can process 1024 tokens as input. Each one of those tokens flows through all the decoder blocks along its own path. That's a major advantage of the transformers architecture, as the output generation can be performed in parallel, lowering significantly the computational cost.

The simplest way to run a trained GPT-2 is to allow it to generate sentences on its own (which is technically called generation of unconditional samples) or to give it a prompt in order to make it generate sentences about a certain topic (generating interactive conditional samples). The trained model uses the $< |endoftext| >$ as a start token for generating words. Let's denote it with $< s >$ instead. The input tokens are fed into the model, passing through all the decoder layers and finally producing an output vector.
According to the decoder method used, an output word is selected from the vocabulary based on the output vector. The most common decoding method is to select the word with the highest probability from the vocabulary. However, we can use more complex methods to achieve better decoding results. Those methods are analysed in Section 4.10. An illustration of the process described above is shown in Figure 4.16.

As already mentioned in Section 4.6, the transformers can not model the sequential order of the input words. Consequently, in order to maintain the sequential information the positional embeddings are used (with the same dimension as the input embeddings), which are added to the input emebeddings. In this way, the model keeps track of the sequential order of the input.

4.9 Text-To-Text Transfer Transformer (T5) model

As mentioned earlier, over the past few years, transfer learning has led to a new wave of state-of-the-art results in natural language processing (NLP). The "Text-To-Text Transfer Transformer" (T5) model, presented in [5], achieves state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of important tasks.

The basic idea behind the T5 model, is to convert all NLP tasks into a unified text-to-text-format where the input and output are always text strings. This approach is inspired by previous unifying frameworks for NLP tasks, including casting all text problems as question answering [182], language modeling [69], or span extraction [183] tasks. This approach of text-to-text format, allows to easily apply the same model, objective, training procedure and decoding process to every task considered, such as question answering, document summarization, sentiment classification and machine trans-

lation tasks. So, in order to train a single model on the diverse set of tasks described above, the researchers added a task-specific (text) prefix to the original input sequence before feeding it to the encoder, in order to specify which task the model should perform. For instance, to ask the model to translate the sentence "That is good." from English to German, the sequence "translate English to German: That is good." would be fed as input to the model and the sentence "Das ist gut." would be used as target.

The T5 model closely follows the original Encoder-Decoder Transformer architecture, proposed in [3] and described in Section 4.6. First, an input sequence of tokens is mapped to a sequence of embeddings, which is then passed into the encoder. The encoder consists of a stack of "encoder layers", each of which comprises two subcomponents: a self-attention layer followed by a small feed-forward network. The self-attention layer uses "fully-visible" attention mask. Fully-visible masking allows the self-attention mechanism to attend to any entry of the input when producing each entry of its output. Layer normalization is applied to the input of each subcomponent. A simplified version of layer normalization is used where the activations are only rescaled and no additive bias is applied. After layer normalization, a residual skip connection is used. Dropout is also applied within the feed-forward network, on the residual skip connection, on the attention weights, and at the input and output of the entire stack.

The decoder is similar in structure with the encoder, except that it also includes a standard attention mechanism after each self-attention layer, that attends to the output of the encoder. The self-attention layers of the decoder use causal masking, which only allows the model to attend to past outputs. An illustration of "fully-visible" and "causal" masking is shown in Figure 4.17. The output of the final decoder block is fed into a dense layer with a softmax output. A simplified form of position embeddings is used, where each embedding is simply a scalar that is added to the corresponding logit used for computing the attention weights. The position embeddings parameters are shared across all layers, but within a given layer each attention head uses different learned position embeddings. More specifically, for the T5-base model, the encoder and decoder stacks consist of 12 layers. The feed-forward networks in each layer consist of a dense layer with an output dimensionality of 3072 followed by a $ReLU$ non-linear activation function and another dense layer. All attention mechanisms have 12 heads and the "key" and "value" matrices of all attention mechanisms have an inner dimensionality of 64. All other sub-layers and embeddings have a dimensionality of $d_{model} = 768$. For regularization, a dropout probability of 0.1 is used. In total, the T5-base model has about 220 million parameters.

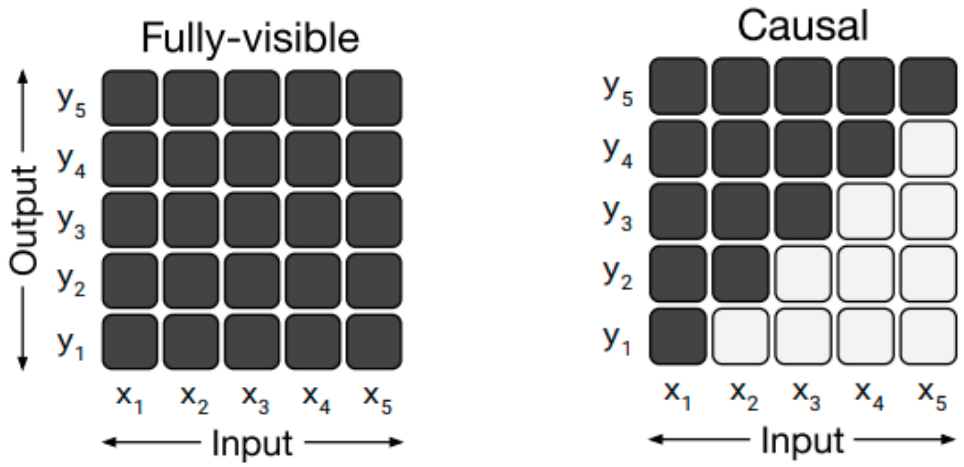


Figure 4.17: Fully-visible and causal masks used in self-attention mechanism.

The model is pre-trained on unlabeled data, in order to generalize knowledge that will be useful during fine-tuning. The "Colossal Clean Crawled Corpus" (C4), a large pre-training dataset of unlabeled data,

which is a cleaned version of Common Crawl dataset (text scraped from the web) that is two orders of magnitude larger than Wikipedia is used. With a denoising objective, the model is trained to predict missing or otherwise corrupted tokens in the input. Inspired by BERT's "masked language modeling" objective, already mentioned in Section 4.7, and the "word dropout" regularization technique [184], an objective that randomly samples and then drops out 15% of tokens in the input sequence, is designed. 90% of the corrupted tokens are replaced with a special mask token and 10% are replaced with a random token.The model is trained to predict those masked tokens.

As far as the fine-tuning process is concerned, the researchers experimented with different approaches such as fine-tuning on each downstream task, multitask training, leave-one-out multitask training etc.

Finally, an illustration of how the T5 model is used in different tasks is shown in Figure 4.18

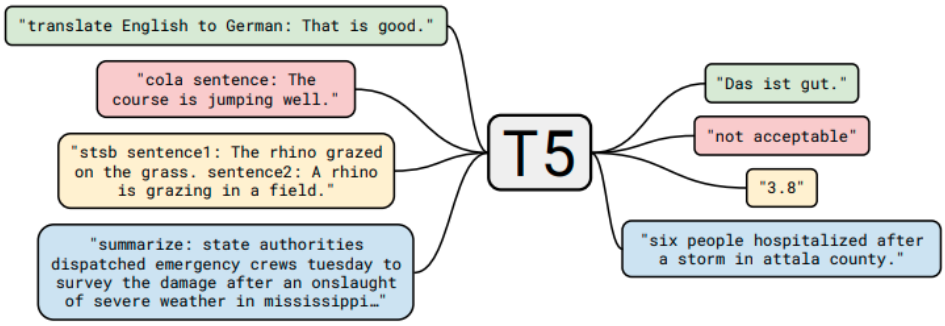


Figure 4.18: An illustration of how the T5 model is used in different tasks. Image source: [5]

4.10 Decoding Methods

4.10.1 Theoretical Background

Besides the improved transformer architectures and the massive unsupervised training data, decoding methods also play a significant role in generating coherent and fluent responses. A decoding method is a strategy applied on the decoder, according to which we select which word will be generated. Of course, those methods can be applied to various models, such as XLNet [177], OpenAi-GPT [169], CTRL [185], Transformer-XL [176], XLM [186], Bart [187] and T5 [5], for auto-regressive language generation. In short, auto-regressive language generation is based on the assumption that the probability distribution of a word sequence can be decomposed into the product of conditional previous words distributions:

$$P\left(x_{1:T} \mid X_0\right) = \prod_{t=1}^{T} P\left(x_t \mid x_{1:t-1}, X_0\right) \tag{4.17}$$

where $x_{1:0} = \emptyset$ and X_0 being the initial context word sequence. The length T of the word sequence is usually determined on-the-fly and corresponds to the timestep $t = T$ where the "<EOS>" or the "<|endoftext|>" token is generated.

In this subsection, we present the currently most prominent decoding methods, mainly greedy decoding, beam search, sampling, top-K sampling and top-p sampling.

Greedy decoding:
Greedy decoding is the simplest decoding method that can be used for generating sequences. According to this method, given the previous words that where generated, on each time-step we select the

word with the highest occurrence probability. More formally, the word to be generated is selected according to the following equation:

$$x_t = \text{argmax}_x P(x \mid x_{1:t-1}) \tag{4.18}$$

An example of greedy decoding is shown in Figure 4.19.

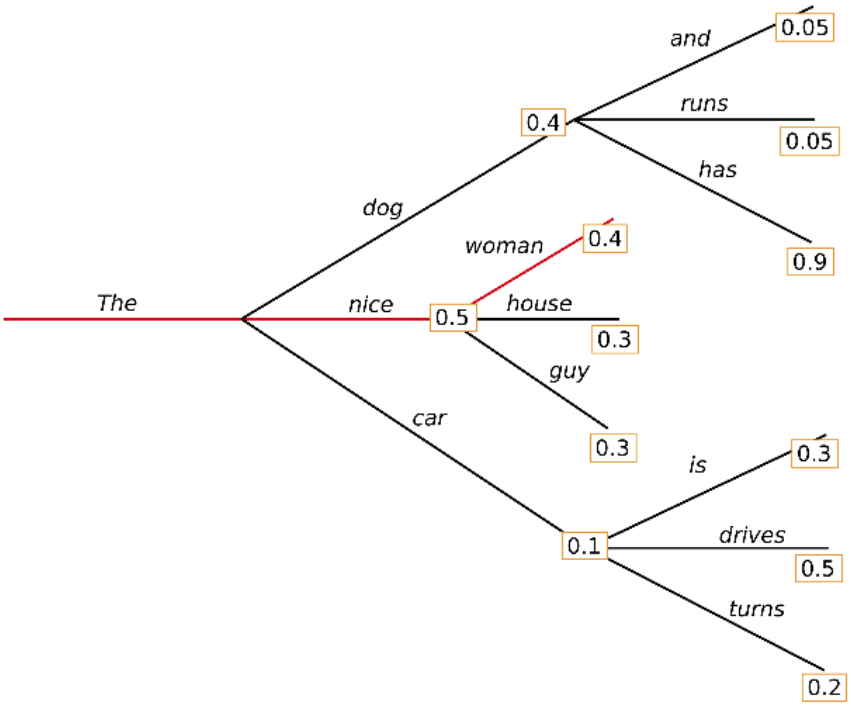


Figure 4.19: An example of greedy decoding. Image source: [24]

In this example, starting from the word "The", the algorithm greedily chooses the next word of highest probability "nice" and so on, so that the final generated word sequence is "The nice woman", having an overall probability of $0.5 \times 0.4 = 0.2$. As we previously mentioned the algorithm selects the word with the highest probability of occurrence at each time-step, checking the probabilities of the next time-step only, without "looking" any further. This leads to the problem, that a path with a higher probability of occurrence may be neglected. This fact can be clearly seen in the previous example, where the sequence "The dog has" is neglected, besides the fact that this path has a total probability of $0.4 \times 0.9 = 0.36$ which is greater than the one of the sequence "The nice woman". Another major problem that the use of greedy decoding creates, is that of repeating the same words or sequences. While the generated words following the context may be reasonable, the model may quickly start repeating itself. This is a very common problem in language generation in general and seems to be even more so in greedy decoding and beam search [188, 189].

Beam Search:
Beam search is one of the most common and reliable methods of decoding, that it still being used. It aims to solve the issue of ignoring word sequences with higher probability, occurred in greedy decoding method. It reduces the risk of missing hidden high probability word sequences by analyzing more paths at each time-step. So, at each time-step it keeps the most likely of hypotheses and eventually chooses the hypothesis that has the overall highest probability. The number of hypotheses kept at each time-step is a parameter determined by the researcher, called "number of beams". Let's denote this parameter with num_beams. An example of applying beam search with $num\_beams = 2$ is shown in Figure 4.20. At time step 1, besides the most likely hypothesis, which is "The woman",

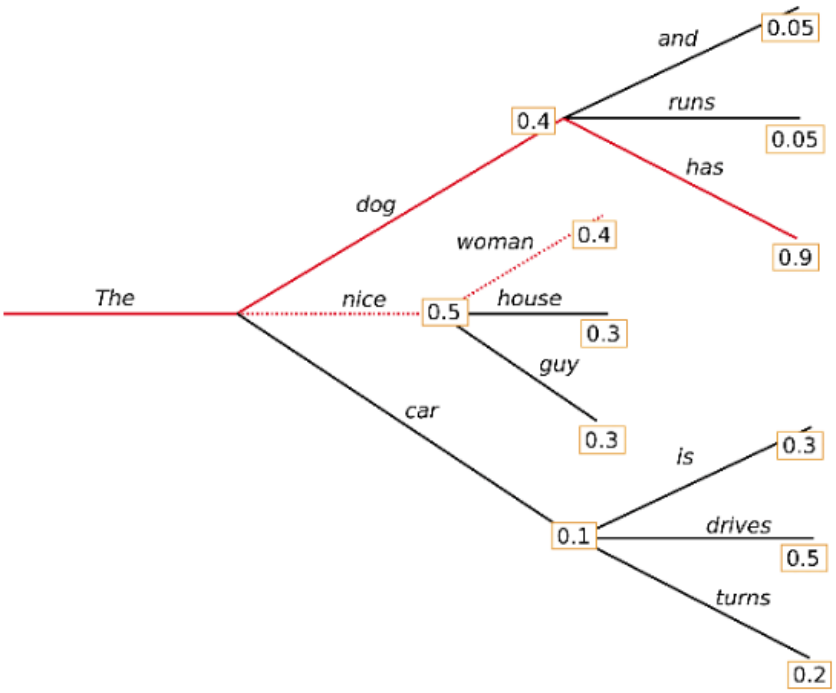


Figure 4.20: An example of beam search using a number of beams equal to 2. Image source: [24]

beam search also keeps track of the second most likely one, "The dog". At time step 2, beam search finds that the word sequence "The dog has" has a higher probability than "The nice woman", so it selects the most likely one. However, we should note here that beam search is not guaranteed to find the word sequence with the highest probability, but it always finds an output sequence with higher probability than greedy decoding.
Regarding the issue of repetition, appearing in greedy decoding method, it may still occur in beam search. However a simple approach to tackle this issue, is to use n-grams penalties as introduced by [91] and [190]. The most commonly used n-grams penalty, makes sure that no n-gram appears twice, by manually setting the probability of next words that could create an already seen n-gram to 0.
Another important feature of beam search that is worth mentioning, is that we can compare the top beams after generation and choose that one that best fits the situation. In other words, we can generate more than one sequences and then select the most suitable. However, the number of returned sequences must be of course less than the number of beams used.
However, beam search may not be always the best option for choosing in dialogue generation. Beam search can work very well in tasks where the length of the desired generation is more or less predictable, as in machine translation [191, 192] in contrast to dialogue generation where the desired output length can vary greatly. Moreover, high quality human language does not follow a distribution of high probability for next words, as mentioned in [193], as humans want to "generate" text in a surprising way, and not to be boring or predictable.

Sampling:
In order to encounter the issue mentioned by [193], and to introduce some randomness instead of producing "boring" responses, we can use sampling methods for decoding.
In its most basic form, sampling means randomly picking the next word x_t according to its conditional probability distribution as shown below:

$$x_t \sim P(x|x_{1:t-1}) \tag{4.19}$$

Using sampling, language generation is not deterministic anymore. To understand better the sam-

pling process an example is illustrated in Figure 4.21. In the example above, the word "car" is sam-

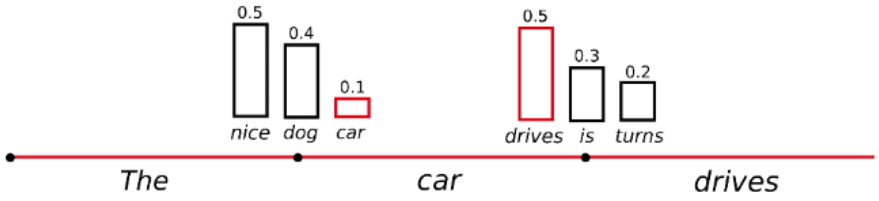


Figure 4.21: An example of sampling. Image source: [24]

pled from the conditioned probability distribution $P(x|"The")$, followed by sampling "drives" from $P(x|"The","car")$.
However, when sampling word sequences, it is very common that the models may produce incoherent responses. On way to encounter this problem, is to use "temperature" on the softmax layer. By lowering $temperature$, we make the probability distribution $P(x|x_{1:t-1})$ sharper, increasing the likelihood of high probability words and decreasing the likelihood of low probability words. We should note here that if we set $temperature = 0$ then the decoding method becomes the greedy decoding. An illustration of applying temperature on the previous example could look as follows (Figure 4.22):

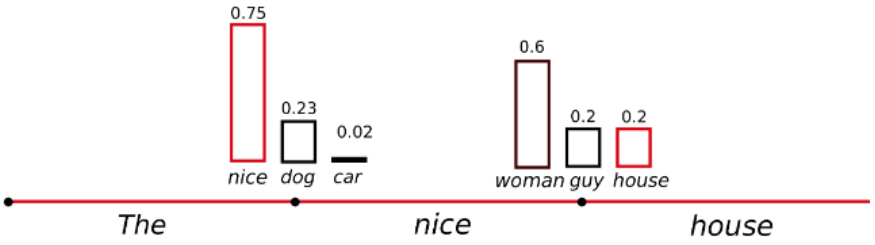


Figure 4.22: An example of sampling with temperature. Image source: [24]

Top-k Sampling:
Top-k sampling is a powerful sampling method introduced by [194]. During top-k sampling, the k most likely next words are filtered and the probability mass is redistributed among only those k words. However, this can be problematic as some words might be sampled from a very sharp distribution (words with high probability before the redistribution), whereas others from a much more flat distribution (words with very low probability before the redistribution).

Top-p (nucleus) Sampling:
To address the problem mentioned before with the use of top-k sampling, top-p (nucleus) sampling was introduced by [193]. Instead of sampling from the most likely k words, in top-p sampling we choose words from the smallest possible set of words, whose cumulative probability exceeds the threshold-probability p. This decoding method allows some dynamic selection of words.

Top-k Top-p Sampling: Top-p can also be used in combination with top-k sampling. In the first step we apply k filtering, and then we apply top-p sampling. In this way, we can avoid very low ranked words while allowing for some dynamic selection too.

4.10.2 Examples

Having already analysed the theoretical background of the most commonly used decoding methods in the previous subsection, in this subsection we provide some characteristic examples. For the examples

presented below, we fine-tuned the T5-base model on the EmpatheticDialogues dataset (described in Section 5.3), a dataset of one-to-one engaging conversations. After fine-tuning the model, we selected a conversation from the test set and we generated the last turn of the conversation using different decoding methods. We should note here that for all generation examples the maximum length used in generation is 40. The results are presented in the following.

Greedy decoding:
At first we used the simplest generation method which is greedy decoding. The generated sentence is shown in Figure 4.23.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: I bet it was. Did you get a good night's rest?

Figure 4.23: A generation example using greedy decoding.

Beam search:
Then, we experimented with using the beam search decoding method with beam size of 4. We also used a length penalty of 0.65. The generated sentence is shown in Figure 4.24.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: I'm glad you had a good time.

Figure 4.24: A generation example using beam search.

Sampling: Then, we experimented with using the sampling decoding method without the use of temperature. The generated sentence is shown in Figure 4.25.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: Did your night end well?

Figure 4.25: A generation example using sampling decoding method.

Sampling with temperature: We also experimented with using the sampling with temperature of 0.9. The generated sentence is shown in Figure 4.26.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: I love going in the woods, was it a game of football?

Figure 4.26: A generation example using sampling with temperature decoding method.

Top-k Sampling: We also experimented with top-k sampling decoding method. We set $topk$ equal to 10. The generated sentence is shown in Figure 4.27.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: I can imagine, what was it like?

Figure 4.27: A generation example using top-k sampling decoding method.

Top-p Sampling: Moreover, we experimented with top-p sampling decoding method. We set $topp$ equal to 0.9. The generated sentence is shown in Figure 4.28.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: That is a good thing to experience. Was there a strong wind?

Figure 4.28: A generation example using top-p sampling decoding method.

Top-k Top-p Sampling: Finally, we experimented with top-k top-p sampling decoding method. We set $topk$ equal to 10 and $topp$ equal to 0.9. The generated sentence is shown in Figure 4.29.

Conversation:
Speaker: I went walking in the woods one night. It was pretty scary out there
Listener: Did anything happen?
Speaker: No, everything was fine, but it was a chilling experience
Generated Sentence: That's good. What were you thinking about?

Figure 4.29: An generation example using top-k top-p sampling decoding method.

By reviewing the generated responses, we can notice that the first two (greedy decoding and beam search) are not very relevant to the context, but they are syntactically and grammatically correct. However, the responses that were generated using top-k, top-p and top-k top-p sampling are on topic. Those three sampling techniques generated engaging responses that are closely relevant to the context of the conversation, being also coherent and fluent.

4.11 Evaluation Metrics

In this section, we look into the most commonly used metrics, for evaluating the generated responses of an open-domain conversational agent. Those metrics are divided into automatic and human-based metrics.

Automatic metrics:
Although there is no well-established method for automatic evaluation of the response quality, there are some automatic metrics for reference.

- **Word Perplexity:** Word perplexity is a metric proposed to evaluate probabilistic language models [18, 195], that has seen significant use for evaluating end-to-end dialogue systems. This metric explicitly measures the probability that the model will generate the ground truth (actual) next word given some context of the conversation. The lower the perplexity is, the better the model. A re-weighted perplexity metric has also been proposed where stop-words, punctuation, and other special tokens are removed before evaluating to focus on the semantic content of the phrase [20]. However, in dialogue the distribution over the words in the next utterance can be highly multi-modal as we have many possible responses, so that metric is not always objective and may not be as suitable.

- **BLEU:** BLEU (bilingual evaluation understudy) metric [196] is also a metric widely used for reference in dialogue systems, borrowed from machine translation tasks. BLEU metric grades an output response according to n-gram matches to the reference. It is defined as:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{\text{N}} w_{\text{n}} \log \text{p}_{\text{n}}\right) \tag{4.20}$$

  where BP is the brevity penalty on the length of the utterance, p_n is probability that the n-grams in a generated response occur in the real response, N is the max number of grams and w_n is the weight for each n-gram (normally set as $\frac{1}{n}$). BLEU's output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts. So, a higher BLEU score is indicative of a better model as the generated response is closer to the real one. However, its effectiveness on automatically assessing dialogue response generation is unclear, as BLEU correlates poorly with human judgment according to [197].

- **Response Diversity:** Distinct-1 and Distinct-2 our two metrics introduced by [198], which respectively measure the number of distinct unigrams and bigrams of the generated responses. Those metrics try to measure the diversity of the generated responses. They may be useful in combination with other such as BLEU and perplexity.

Human-based metrics:
Currently human evaluation is still the most convincing method for judging the response quality and is widely applied in chatbot evaluation. The most common human-based metrics are:

- **pair-wise comparison** to let humans choose which of the two responses is more suitable, more appropriate, and more helpful, etc. [142, 143]

- **evaluating relevance:** Humans grade the generated responses according to whether they seem relevant to the conversation and on-topic. [25]

- **evaluating fluency/coherency:** Humans grade the responses according to whether they seem understandable, logically and syntactically correct. [25]

4.12 Summary

In this chapter we provided a theoretical background knowledge for dialogue generation. At first, we studied the vanilla seq2seq architecture using RNNs and we extended this architecture with the attention mechanism. This architecture comprises the cornerstone for building conversational agents. Then, we presented the HRED architecture, which comprises an essential approach for taking into account the previous context of the conversation, when generating a new response. Moreover, we

analyzed the Transformer Encoder Decoder architecture, a recent approach which achieves much better results in dialogue generation than the previously mentioned architectures, using multi-head self-attention on the encoder and multi-head causal attention on the decoder. Based on the transformer model, we then studied the Bert and GPT-2 models which can also be used for dialog generation achieving satisfying results. Furthermore, we presented the T5 model, a model pre-trained on a very large corpus, which can achieve state-of-the-art results in dialog generation tasks. We summarize the basic characteristics of the aforementioned models in table 4.2.

Table 4.2: Summary of the models we studied

Model Name	Characteristics
Vanilla seq2seq	simple seq2seq model
Vanilla seq2seq with attention	attention over the encoder
HRED	hierarchical encoding, able to encode previous context
Transformer Encoder Decoder	seq2seq, self-attention, masked self-attention, able to encode previous context
BERT	encoder stack, self-attention (bidirectional), emphasizes on encoding, usupervised pretrained
GPT2	decoding stack, masked self-attention, emphasizes on decoding, unsupervised pretraining
T5	seq2seq, self-attention, masked self-attention, usupervised pretraining

After studying in depth the aforementioned models, we reach to the conclusion that although vanilla seq2seq and vanilla seq2seq with attention are traditional models, they are the starting key for dialog generation as recent models have adopted many ideas of their architectures. For example, the Transformer model adopted the seq2seq architecture, enhancing it with the self-attention mechanism. While the transformer and the T5 models focus both on encoding and decoding processes, BERT and GPT2 do not follow the same way. In contrast, BERT focuses on the encoding process using an encoder stack, while GPT2 focuses on the decoding process using the decoder stack. However, both of them can achieve satisfactory results (as well as the T5 model) if they are used properly, as we will see in the following chapter.

Furthermore, in Section 4.10 we provided a theoretical background of the most commonly used decoding methods, giving also plenty of examples for better understanding. Finally, we presented both automated and human evaluation metrics for evaluating conversational agents. We should note here that the mentioned automatic metrics are essential for evaluation, but of course human evaluation should not be neglected as automatic metrics did not provide fully objective results. In the following chapter, we study in depth dialog generation with empathy providing essential experiments (using the state-of-the-art models we described in this chapter) and presenting the corresponding results.

Chapter 5

Dialogue Generation with Empathy using Generative Models

5.1 Introduction

The rapid development in the field of generative modeling using neural networks has helped in the creation of intelligent conversational agents. Current conversational agents achieve impressive results by being able to communicate with the user, keeping his interest high. However, beyond understanding what is being discussed, human communication requires an awareness of what someone is feeling. While it is straightforward for humans to recognize and acknowledge others' feelings in a conversation, this is a significant challenge for AI systems. Humans use different types of emotions depending on the situation of the conversation. Emotions also play an important role in mediating the engagement level with conversational partners. We should also note that natural communication is frequently prompted by people sharing their feelings or circumstances. A recent study found that 80% of Twitter users seem to post mostly about themselves [199], and ELIZA [28], one of the earliest chatbots developed, focused most of its attention on asking its conversational partners why they were feeling a certain way. For instance, let's take a look at the illustrated dialogue example in Figure 5.1. While giving a response like "Why would anyone promote you?" is contextually relevant, "Congrats! That's great!" is more natural because it acknowledges the underlying feelings of accomplishment. As shown in this example, people generally respond to others in a way that is empathetic or that acknowledges how the other person feels. Consequently, one of the most significant challenges for a human-facing dialogue agent is to appropriately respond to a conversation partner that is describing personal experiences, by understanding and acknowledging any implied feelings — a skill we refer to as empathetic responding.

In the view of creating chatbots that are capable of understanding and acknowledging any implied feelings, in this chapter we present the recent work done in that field, we conduct several experiments using generative models, and finally, we analyze and compare the corresponding results. More specifically, in Section 5.2 we present the related work that has already been done, in Section 5.3 we present

Figure 5.1: A dialogue example where acknowledging an inferred feeling is appropriate. Image source: [25]

and analyze the datasets used in our experiments, in Section 5.4 we present the baseline models that have already been used, in Section 5.5 we introduce our proposed approaches, and finally, in Section 5.6 we present the experiments that have been conducted and we compare the results of the proposed models with those of the baselines.

5.2 Related Work

Open-domain conversational models have been widely studied. Traditionally conversational agents are built using the seq2seq architecture [200]. Prior research has shown that engaging with these agents leads to short conversations [201] as the responses produced are dull and generic without containing an emotional tone [202, 200]. Efforts to make the conversation more engaging were made by keeping track of the context of the conversation [151, 20, 150, 152] or by trying to produce more diverse responses [198, 155]. Others tried to promote response diversity by combining retrieval and generation models [203, 204, 205]. Later, a trend was to produce personalized responses by conditioning the generation on a persona profile to make the responses more consistent through the dialogue [202]. The PersonaChat [206, 207] dataset was created and later it was extended in the ConvAI2 challenge [208]. Those works shown that we can make agents with more consistent personality by giving personality information as input to the model. A lot of work has been presented later based on persona profile conversational agents [209, 210, 207, 211, 158, 212, 213, 214]. However, those works focused only on creating a conversational agent enacting a consistent persona, without taking into account the feelings of the conversational partner.

Apart from producing engaging responses, understanding the situation, and producing the right emotional responses, is another desirable trait. A lot of researches have focused on emotion [215, 216, 217, 218, 219, 220, 221] and empathy in the context of dialogue systems [42, 222, 43]. A framework to control the sentiment and the emotion of the generated response through a manually specified target was successfully introduced by [44, 45, 46] , while [223] introduced a new Twitter conversation dataset and proposed to distantly supervise the generation model with emojis. Others focused on controlling the emotion of the generation response to encourage higher levels of affect [224]. Meanwhile, others proposed a new benchmark for empathetic dialogue generation [25] and trained models to jointly predict the current emotional state and generate a response [225, 226, 25]. Later, the researchers of [227] improved the initial baselines of [228] using the "Mixture of empathetic listeners" framework. Recently, [229] proposed a method, using reinforcement learning [230], for generating empathetic responses by improving the user sentiment look-ahead. Others [27, 36] used pretrained language models, and by fine-tuning them on the Empathetic dataset, improved the initial baselines.

Meanwhile, other researchers experimented with using larger amounts of data and scaling the size of the models, in order to provide state-of-the-art results in many NLP tasks. More specifically, Previous works [117, 169, 4] showed that leveraging a large amount of data to learn context-sensitive features from a language model can create state-of-the-art models for a wide range of tasks. Taking this further, [69, 177] deployed higher capacity models and improved the state-of-the-art results. Finally, while prior work has shown that scaling neural models in the number of parameters and the size of the data they are trained on gives improved results, Facebook AI research team shown that other ingredients are important for a high-performing chatbot too [35].

5.3 Data

In this section we present and analyse the datasets used in our experiments. We used the Empathetic-Dialogues Dataset [25] and the ConvAI2 Dataset [208] (which extends the PersonaChat Dataset [206]) which are analysed in the following part.

5.3.1 EmpatheticDialogues Dataset

The EmpatheticDialogues Dataset [25] is an open-domain conversation dataset consisting of around 25k conversations, publicly released with code to reproduce the main experimental results of the relevant paper [1]. We consider an open-domain one-on-one conversational setting where two people are discussing a situation that happened to one of them, related to a given feeling. Each conversation is grounded in a situation, which one participant writes about with a given emotion label. Then the person who wrote the situation (Speaker) has an one-on-one conversation with another (Listener). In the following part, we take a deeper look in the conversation format and then we analyse the data collection procedure.

Emotion labels: As mentioned before, each conversation is grounded in a situation, which one participant writes about in association with a given emotion label. There are 32 emotion labels, covering a wide range of positive and negative emotions. Each conversation is provided with a single emotion label in order to have a situation strongly related to (at least) one particular emotional experience. However, in a given conversation similar emotions may be invoked as some emotions are closely related. In Figure 5.2, a distribution over the emotion labels within the training set is depicted.

Emotion	Most-used speaker words	Most-used listener words	Training set emotion distrib
Surprised	got,shocked,really	that's,good,nice	5.1%
Excited	going,wait,i'm	that's,fun,like	3.8%
Angry	mad,someone,got	oh,would,that's	3.6%
Proud	got,happy,really	that's,great,good	3.5%
Sad	really,away,get	sorry,oh,hear	3.4%
Annoyed	get,work,really	that's,oh,get	3.4%
Grateful	really,thankful,i'm	that's,good,nice	3.3%
Lonely	alone,friends,i'm	i'm,sorry,that's	3.3%
Afraid	scared,i'm,night	oh,scary,that's	3.2%
Terrified	scared,night,i'm	oh,that's,would	3.2%
Guilty	bad,feel,felt	oh,that's,feel	3.2%
Impressed	really,good,got	that's,good,like	3.2%
Disgusted	gross,really,saw	oh,that's,would	3.2%
Hopeful	i'm,get,really	hope,good,that's	3.2%
Confident	going,i'm,really	good,that's,great	3.2%
Furious	mad,car,someone	oh,that's,get	3.1%
Anxious	i'm,nervous,going	oh,good,hope	3.1%
Anticipating	wait,i'm,going	sounds,good,hope	3.1%
Joyful	happy,got,i'm	that's,good,great	3.1%
Nostalgic	old,back,really	good,like,time	3.1%
Disappointed	get,really,work	oh,that's,sorry	3.1%
Prepared	ready,i'm,going	good,that's,like	3%
Jealous	friend,got,get	get,that's,oh	3%
Content	i'm,life,happy	good,that's,great	2.9%
Devastated	got,really,sad	sorry,oh,hear	2.9%
Embarrassed	day,work,got	oh,that's,i'm	2.9%
Caring	care,really,taking	that's,good,nice	2.7%
Sentimental	old,really,time	that's,oh,like	2.7%
Trusting	friend,trust,know	good,that's,like	2.6%
Ashamed	feel,bad,felt	oh,that's,i'm	2.5%
Apprehensive	i'm,nervous,really	oh,good,well	2.4%
Faithful	i'm,would,years	good,that's,like	1.9%

Figure 5.2: Distribution of emotion labels within EMPATHETICDIALOGUES training set and top 3 content words used by speaker/listener per category. Image source: [25]

One-on-One conversation: The person who wrote the situation description,is the "Speaker" ,and initiates a conversation to talk about it. The other conversation participant, is the "Listener" and he is unaware of the emotion label or the theme of the conversation. The Listener becomes aware of

[1] https://github.com/facebookresearch/EmpatheticDialogues

the underlying situation through what the Speaker says and responds. The Speaker and the Listener exchange up to 6 turns.

Collection Details: The dialogues are collected using the ParlAI platform [2], by hiring 810 US workers to interact with Amazon Mechanical Turk (MTurk). A pair of workers are asked to:

- select an emotion word each (among 32 emotions) and describe a situation when they felt that way
- and to have a conversation about each of the situations, as outlined below.

Each worker had to contribute to at least one situation description and one pair of conversations: one as a Speaker about the situation he/she contributed, and one as a Listener about the situation contributed by another worker. The workers were allowed to participate as many times they wanted for the first 10k conversations approximately, but then the most ”frequently active” workers were limited to a maximum number of 100 conversations. Finally, the median number of conversations per worker was 8, while the average was 61 (some workers were more active contributors than others).

Setup: So, at first the workers were asked to describe in a few sentences a situation based on a feeling label, trying to keep these descriptions between 1-3 sentences. In the second stage, two workers were paired and asked to have two short chats with each other. In each chat, one worker (speaker) starts a conversation about the situation they previously described, and the other worker (listener) responds. Neither can see what the other worker was given as emotion label or the situation description they submitted, so they must respond to each others' stories based solely on cues within the conversation. Each conversation is allowed to be 4 to 8 utterances long. After the first few initial rounds of data collection the workers were forced to select an emotion among three emotion labels that had been the least chosen overall so far, if it was their first time working on the task. If they had already performed the task, the offered emotion labels were among those that they had chosen the least often before. In this way, the distribution over the emotion labels is almost balanced, as shown in Figure 5.2.

Summing up, the resulting dataset consists of 24,850 conversations. The data are splitted into approximately 80% train, 10% validation and 10% test partitions and in order to prevent overlaps between partitions, all sets of conversations with the same speaker providing the initial situation description are set in the same partition. The final train/val/test split has 19533 / 2770 / 2547 conversations, respectively. For the situation descriptions the average length is 19.8 words. Moreover, each conversation has on average 4.31 utterances and the average utterance length is 15.2 words long. The basic statistics are presented in table 5.1.

Table 5.1: Statistics of Empathetic Dialogue dataset

	Train	*Valid.*	*Test*
Num. of conversations	19433	2770	2547
Num. of utterances	84324	12078	10973
Avg (utt.) length conversations	4.31	4.36	4.31

In Figure 5.3 we provide some conversations from the training set.

[2] https://github.com/facebookresearch/ParlAI

Label: Content
Situation: Speaker felt this when...
"eating my favorite meal makes me happy."
Conversation:
Speaker: i am at my best when i have my favorite meal.
Listener: nice
Speaker: i love enchiladas
Listener: really?
Speaker: yes. enchiladas for the win!

Label: Anticipating
Situation: Speaker felt this when...
"I cant wait to go on my end of summer trip"
Conversation:
Speaker: I cant wait to go on my end of summer trip in texas.
Listener: Sounds like fun. What you got planned ?
Speaker: not really sure but im excited to just be invited
Listener: Got any family out there? Cousins perhaps

Label: Joyful
Situation: Speaker felt this when...
"I have had a great week!"
Conversation:
Speaker: I have had a great start to my week!
Listener: That's great. Do you think the rest of the week will be as great?
Speaker: I hope so! It looks promising!!
Listener: Lucky you. Are you always a positive person or it's just been an amazing week really?
Speaker: haha. Kind of both. And also probably too much coffee to start my shift tonight

Label: Terrified
Situation: Speaker felt this when...
"I got home for lunch and found a bat outside on my front porch."
Conversation:
Speaker: I got home for lunch and found a bat outside on my front porch. It probably has rabies. Bats shouldn't be out during the day.
Listener: Doesn't rabies cause sensativity to light? Either way I would freak out...
Speaker: It can but, it also causes anmails to behave erratically... like bats wadering around in the middle of the day.
Listener: Oh yeah, gotcha. I really don't like animals that are small and move quickly
Speaker: Generally yes.

Figure 5.3: Random examples from Empathetic Dialogues training set.

5.3.2 ConvAI2 Dataset

The ConvAI2 dataset [208] is publicly available in ParlAI [3] and is based on the Persona-Chat dataset [206]. As the ConvAI2 dataset extends the Persona-Chat dataset, it is considered necessary to first give a brief description of the Persona-Chat dataset.

The Persona-Chat dataset is a crowd-sourced dataset, collected via Amazon Mechanical Turk, consisting of one-on-one open domain dialogue conversations, where each of the pair of speakers conditions their dialogue on a given profile, which is provided. The data collection is based on the three following stages:

Personas collection: A set of 1155 possible personas is crowdsourced, each consisting of at least 5 profile sentences. From the collected personas, 100 "never seen before" personas are set aside for the validation set and 100 for test.

Revised personas: In order to avoid modeling that takes advantage of trivial word overlap, additional rewritten sets of the same 1155 personas are crowdsourced, with related sentences that are rephrases, generalizations or specializations.

Persona Chat: Two Turkers are paired and a random original persona is assigned on each one. Then, they are asked to chat, while playing the part of the given character. The dialogs are turn-based, with a maximum of 15 words per message.

[3] https://github.com/facebookresearch/ParlAI/tree/master/parlai/tasks/convai2

The resulted Persona-Chat dataset consists of 162,064 utterances over 10,907 dialogs, where 15,602 utterances (1000 dialogs) of which are set aside for validation, and 15,024 utterances (968 dialogs) for test.

The ConvAI2 dataset extends the original Persona-Chat dataset by crowdsourcing further data. The format of the dialogues remains the same. In table 5.2 the basic statistics of the dataset are shown. We should note here, that the test set of the dataset is not publicly available, however that is not a matter as the dataset was used only for pretraining our models. In table 5.3 an example dialogue is shown.

Table 5.2: Statistics of ConvAI2 dataset

	$Train$	$Valid.$	$Test(Hidden)$
Num. of conversations	17,878	1000	1015
Num. of utterances	131438	7801	6634
Num. of personas	1155	100	100

Table 5.3: ConvAI2 dialogue example

Persona 1	**Persona 2**
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

5.4 Baseline architectures

In this Section we describe in depth the generative models that have been already proposed in [25, 27, 36]. The official results of those models are presented in 5.6. We should also mention that all the models are fine-tuned on the EmpatheticDialogues dataset. Before analyzing each of those models, we provide a summary table that contains the basic characteristics of each model (Table 5.4).

Model 1 - Vaswani Full Transformer: In [26, 25] the researchers used the full transformer architecture, which we already described in Section 4.6. The "base" model consists of 4 layers and 6 heads, while the "large" one has 5 layers instead.

Table 5.4: Summary table of baseline architectures

Model Name	Pretraining	Architecture Characteristics	Bib.
(1) Vaswani Full Transformer	Reddit Dataset	transformer	[26, 25]
(2) Multitask Transformer	Reddit Dataset	transformer, multitasking emo	[26, 25]
(3) Prepend-k	Reddit Dataset	transformer, prepending emo/topic	[26, 25]
(4) Ensemble of Encoders	Reddit Dataset	transformer, ensembling emo	[26, 25]
(5) CAiRE	Book Corpus, PersonaChat	GPT, multitasking (3 obj.)	[27]
(6) GPT2-baseline	WebText	GPT2	[36]
(7) GPT2-prepend	WebText	GPT2, prepending (emo+situattion)	[36]
(8) DodecaDialogue MT	Reddit, Twitter	transformer, multitasking (12 tasks)	[37]
(9) DodecaDialogue MT+FT	Reddit, Twitter	transformer, multitasking (12 tasks), finetuning	[37]
(10) BST Generative	Reddit Dataset	transformer, multitasking (4 tasks)	[35]

Model 2 - Multitask Transformer: In [26] the researchers extended the previous architecture (Model 1) with multitask learning. They altered the objective function to also optimise for predicting the given emotion label. They changed the architecture by adding a linear and a softmax layer on the encoder, for predicting the emotion label from the context sentences. The objective function is altered to be the average of the negative log-likelihood of predicting the next utterance and the negative log-likelihood of the added linear layer being able to predict the correct emotion. An illustration of the model is shown in Figure 5.4.

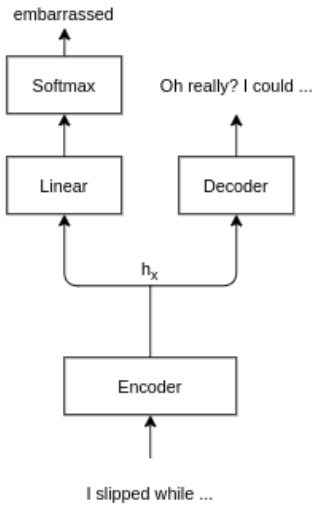


Figure 5.4: Illustration of model 2 - Multitask Transformer, an architecture proposed by [26]. The context representation h_x outputted by the context encoder is used both as input to an emotion classifier, and to generate the next utterance as in the "model 1" setting.

Model 3 - Prepend-k: In [26, 25] the researchers extended the full transformer architecture by adding the best k predictions from a simple classifier to the input text. Then the concatenated text is forwarded to the encoder. Two versions (EmoPrepend-k and TopicPrepend-k) of that model are presented in [25], by trying to add supervised information from two prediction tasks: emotion detection and topic detection. More specifically, for the "EmoPrepend-k" model, the top k predicted emotion labels from the supervised classifier are merely prepended to the beginning of the token sequence as encoder input,

while for the "TopicPrepend-k" model, the top k predicted topic labels from the supervised classifier are prepended to beginning of the token sequence. We explain further the training procedure of the classifier in Section 5.6. An illustration of the "EmoPrepend-k" architecture is shown in Figure 5.5.

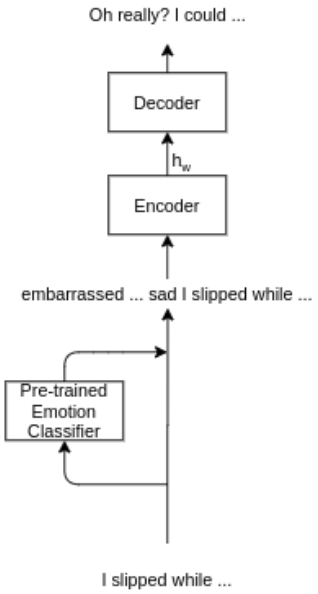


Figure 5.5: Illustration of model 3 - EmoPrepend-k, an architecture proposed by [26, 25]. The input sequence is first run through a pre-trained emotion classifier, and the top k predicted emotion labels are prepended to the sequence, which is then run through the encoder to output a hidden representation h_w. The hidden representation is then used to generate the next utterance.

Model 4 - Ensemble of Encoders: In [26, 25] the researchers extended the full transformer architecture by replacing the simple encoder with the "Ensemble Encoder". The ensemble encoder takes the encoding h_x from the simple transformer encoder, and concatenates it with the representation h_c from the penultimate layer of a deep classifier trained for emotion prediction. Then, the concatenated encodings are linearly projected to the dimension required by the decoder and the output is given as input to the decoder. Finally, the next utterance is produced. An illustration of the architecture described, is shown in Figure 5.6. We should note here that as referred in [26], when training the dialogue model, the basic transformer encoder and the emotion classifier are freezed (referred as "pre-trained" in Figure 5.6).

Model 5 - CAiRE: In [27] the researchers used the Generative Pre-trained Transformer (GPT) [169] model as a pretrained language model. The GPT model is a previous version of the GPT2, which has been already described in Section 4.8. The GPT model is a causal (unidirectional) transformer pre-trained using language modeling on a large corpus with long range dependencies, the Toronto Book Corpus [231]. They used a version with 12 layers, 768 hidden states and 12 attention heads, having in total 110M parameters. We should note here that they first pre-trained the model on the PersonaChat dataset and then they fine-tuned it on EmpatheticDialogues. In Figure 5.7 an illustration of the model's architecture is shown. Following the fine-tuning schema of [158], they created a custom persona for the CAiRE model and then they concatenated the custom persona, the dialogue history and the reply with special separate tokens, representing all the input sources with the summation of

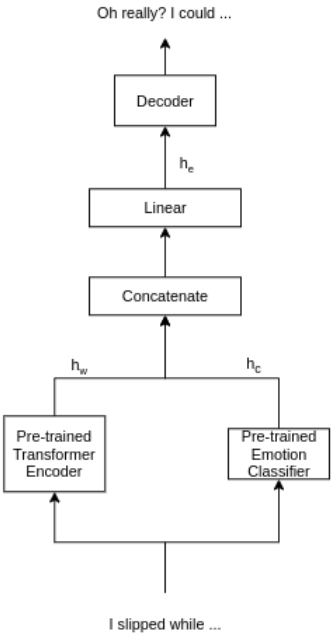


Figure 5.6: Illustration of model 4 - Ensemble Encoders, an architecture proposed by [26]. The input sequence is run through the encoder as well as a pre-trained emotion classifier with the last layer removed. The outputs h_w and h_c are concatenated and linearly projected into a representation h_e. Then the representation h_e is fed to the decoder and the next utterance is produced.

trainable positional embeddings, word embeddings, and dialogue state embeddings. Positional embeddings and word embeddings are required for transformer input, while dialogues state embeddings were added to help CAiRE to model the hierarchical dialogue structure and to distinguish the persona sentences, the dialogue context and the response. After the input representation is fed into the model, we get the contextualized representations. They denoted as SEN the contextualized representation coming from the last special token and as EMO the contextualized representation of the special token before the reply. Using those contextualized representations they trained the model for generating the next utterance (for further training details about the use of the distractor, shown in Figure 5.7, see in Section 5.6).

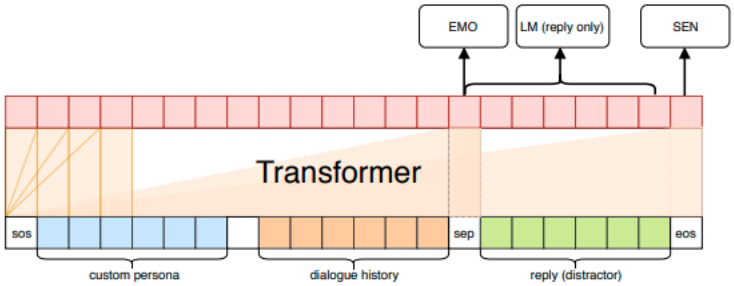


Figure 5.7: Illustration of model 5 - CAiRE, a model proposed by [27].

Model 6 - GPT2-baseline: In [36], the researchers used the GPT2 pre-trained language model, which we described in Section 4.8. They used a version with 12 layers and 12 heads, having in total 117M parameters, and they tried to predict the next response using the dialog history.

Model 7 - GPT2-prepend: In [36], the researchers extended the baseline model (Model 6 - GPT2-baseline) by prepending the emotion labels and the situation context to the dialog history, using special tokens for separating them.

Model 8 - DodecaDialogue MT: In [37] the researchers employed a full transformer based architecture [3] which accepts an image (from ImageChat and IGC datasets - tasks for creating chatbots able to discuss given images), external textual information and a dialog history as input and generates a response. They performed multi-task learning on twelve dialogue tasks, building a single conversational model. More specifically, they used a modification of the transformer seq2seq architecture, by additionally adding pre-trained image features (from the ResNeXt-IG-3.5B model) to the encoder. Their model consists of 8 layers, 512 dimensional embeddings and 16 attention heads and is based on the ParlAI implementation [232].

Model 9 - DodecaDialogue MT+FT: In [37] the researchers extending the previous work, by fine-tuning the models after multi-task training. They trained the previous model on twelve dialogue tasks using multi-task learning and then they fine-tuned it on EmpatheticDialogues to improve the results further. We should also mention here, that they followed the same procedure for each of the twelve tasks, but we focus only on the model which was fine-tuned on the EmpatheticDialogues dataset.

Model 10 - BST Generative: In [35] the researchers used a standard seq2seq transformer architecture, as described in Section 4.6, to generate responses, having approximately 90 million parameters. In this approach they trained the model using multi-tasking on four dialogue tasks (Blended Skills, ConvAI2, EmpatheticDialogues and Wizard-of-Wikipedia). The Blended Skills dataset [233], is a new dataset consisting 76000 utterances, combining three different skills: engaging personality from ConvAI2 [208], empathy from EmpatheticDialogues [25], and knowledge from Wizard-of-Wikipedia (WoW) dataset [234]. We should also note that they also experimented with larger generative models (having billion of parameters), retrieval and retrieve-and-refine models, but we do not refer them in this diploma thesis as they considered to be out of scope.

In Section 5.6 we will discuss the training details and the reported results for all the previously referred models.

5.5 Proposed architectures

In this Section we describe some generative models that were already introduced by [235, 5] in other works, and we propose using them in EmpatheticDialogues Dataset. We compare the results of all models in Section 5.6.

Model 11 - BERT2BERT: This model uses the seq2seq architecture, with encoder and decoder both composed from Transformer layers. More specifically, a Bert encoder and a Bert decoder are used. The model was proposed by [235] for sequence generation. We initialise the encoder and decoder parameters, using the "bert-base-uncased" checkpoint from the HuggingFace library [179]. The BERT2BERT model has 224M parameters, as the BERT encoder and decoder models have 110M parameters (12 layers, 768 hiddens, 12 heads) each.

Model 12 - BERT2GPT2: This model also uses the seq2seq architecture as the previous one. We use Bert as encoder and GPT2 model as decoder. Both are initialised with the corresponding public checkpoints. We use the BERT vocabulary for the input and the GPT2 vocabulary for the output. The model was firstly proposed by [235]. The BERT2GPT2 model has 224M parameters, as the Bert encoder model has 110M parameters (12 layers, 768 hiddens, 12 heads) while the GPT2 decoder model

has 117M parameters (12 layers, 768 hiddens, 12 heads).

Model 13 - T5: This model architecture has been already discussed in Section 4.9. We use the base model from the HuggingFace library [179] having 220M parameters with 12 layers, 768 hidden-states, 3072 feed-forward hidden-states and 12 heads.

Model 14 - T5-multitask1: We extend the architecture of the T5 baseline model (model 13) with multitask learning. We adopt the same idea applied in model 2 (described in Section 5.4). More specifically, we add a classification head over the encoder, trying to predict the emotion of the user. In this way, during the training process the model learns to generate responses according to the user's emotion.

Model 15 - T5-multitask2: We extend the architecture of the T5-multitask1 model (model 14) by adding a classification head over the decoder too. This approach not only tries to predict the emotion of the user, but to penalise the model if the generated response is emotionally misclassified. The architecture used is illustrated in Figure 5.8.

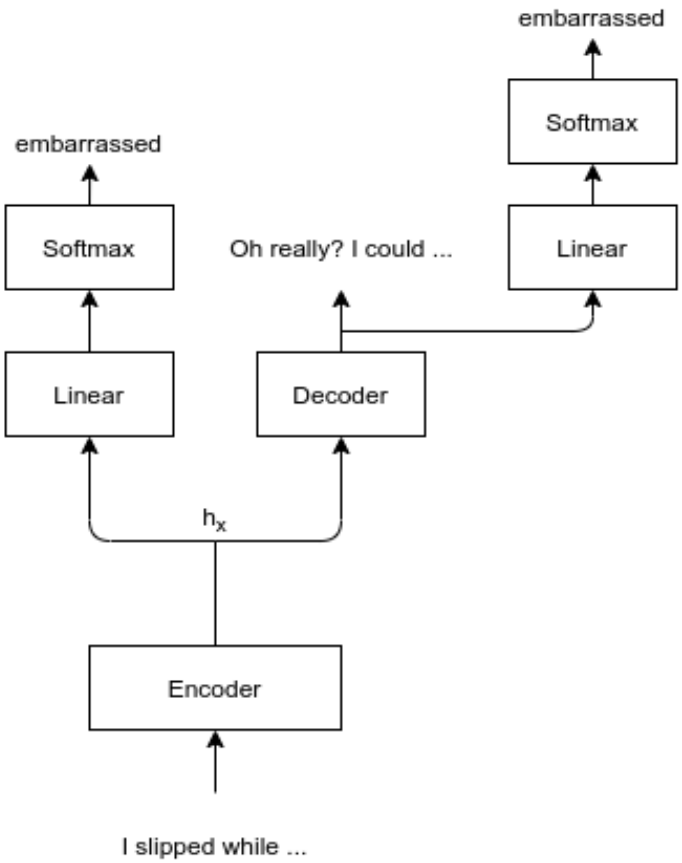


Figure 5.8: Illustration of model 15.

We should also note here, that we have also implemented some of the models that were described in Section 5.4 with slight differences in the training process. We denote those models in the tables of Section 5.6 with the "ours" tag, in order to distinguish them from the original ones.

5.6 Experiments & Results

In this Section, we analyse the training details for the models described in Sections 5.4, 5.5 and then we present the experiments that have been conducted. Finally, we compare the results of the proposed models with those of the baselines.

Training Details of baseline models:

Models 1-4: The researchers in [26, 25], used a dump of 1.7 billion conversations to pre-train the models on predicting replies. Then, they fine-tuned the models on EmpatheticDialogues dataset by predicting the next utterance of the conversation using a context window of four previous utterances, as that is the average length of each dialogue in the EmpatheticDialogues dataset. They also limited the maximum number of word tokens in the context and response to 100 words. All models were trained for up to 10 epochs and the versions with the lowest loss on the validation set were kept. They used 300-d word embeddings, pretrained on common-crawl data using fastText [236]. The Adamax optimizer was used, with a learning rate of 8e-4. For the multitask model (model 2), the objective function was altered during the training procedure and it was set to be the average of the negative log-likelihood of predicting the next utterance and the negative log-likelihood of the added linear layer being able to predict the correct emotion. Moreover, for the Ensemble of Encoders model (model 4), the researchers frozen both the Transformer encoder and the pretrained classifier and trained only the linear layers used to predict the user's emotion and the Transformer decoder. We have to note that they experimented with two versions of the model. In the first one (Ensem-DM) they used the DeepMoji classifier [237] with the weights as released by the authors, while in the second one (Ensem-DM+) they used the same DeepMoji architecture, re-trained on the situation descriptions of the training set of EmpatheticDialogues dataset. Finally, at inference time, they used diverse beam search from [188].

Model 5: The researchers in [27], followed the transfer learning strategy of [158] by pre-training the model on the PersonaChat dataset [206], improving in this way the engagement and the consistency of the model. Then, they fine-tuned the model on the EmpatheticDialogues dataset with a custom persona and three objectives:

- language modeling
- response prediction
- dialogue emotion detection

As already mentioned, they created a custom persona with sentences such as "my name is caire","i am a good friend of humans", and formed the input by concatenating the custom persona, the dialog history and the reply separating them with special tokens. Using the contextualized representations produced by the model they trained the model in the three mentioned objectives:

- To optimize the emotion prediction objective, they pass the EMO representation (the contextualized representation before the reply) into a linear classification head to predict the emotion. For the emotion classification among 32 emotions, they use the cross-entropy loss, denoted with L_e.
- To optimize the response language modeling objective, they use the contextualized representation of gold reply to predict the next tokens, and finally compute the language model loss, denoted with L_l using cross-entropy.
- To optimize the response prediction objective, at each training step they make two predictions. One using the reply and another one using a distractor. At first they pass the SEN representation (the contextualized representation of the last special token - see in Figure 5.7) using the reply

to the input, to a linear classifier to classify if the response is correct or not. Then they sample a distractor response from the training samples against the gold response. Using that distractor to the input instead of the reply, they take one more SEN representation which is forwarded to the linear classifier, predicting if the response is correct or not. Then, they calculate the cross-entropy loss of that binary classification task, denoted with L_s.

Finally the final fine-tuning loss used is a weighted sum of the aforementioned losses, as shown in the following equation:

$$L = \alpha \times L_l + L_s + L_e \tag{5.1}$$

where α is chosen by the researchers.

Models 6,7: The researchers in [36], fine-tuned the pretrained models on EmpatheticDialogues dataset using cross-entropy loss. The vocabulary size used was 50263. During inference time, they used the nucleus sampling technique (which is described in Section 4.10) with p=0.9, instead of using beam search.

Models 8,9: The researchers in [37], pre-trained the models in Reddit and Twitter (to some extend) datasets, before multi-tasking on all of the twelve tasks or multi-tasking and then fine-tuning on a specific task. The pre-training process only included text, while the image encoder was pre-trained separately in previous work [238]. The models were trained with a batch size of 3072 sequences for approximately 3 million updates using a learning rate of 5e-4, and an inverse square root scheduler. Then the models were trained using multi-tasking on all tasks or fine-tuned on a specific task after multi-tasking.

Models 10: The researchers in [35], pre-trained the model in Reddit dataset, and then they fine-tuned it using multi-task learning on the four tasks of Blended-Skills, ConvAI2, EmpatheticDialogues and WoW. In each blended dialogue, the model is provided a two sentence persona to condition on following PersonaChat [206], and additionally during one third of the conversations a WoW topic name as well.

Training Details of our models:

In the following part we provide the training details for the models we implemented and applied on EmpatheticDialogues. Before analysing the training details of each model, we should note here that for predicting an utterance of the conversation, we use as context (history), a context window of four previous utterances (turns).

Models 11-13 (BERT2BERT, BERT2GPT2, T5-baseline):
For the models 8 to 10 that we applied on EmpatheticDialogues dataset, we present two training approaches.The first one is to train them on EmpatheticDialogues and the other one is to pre-train them on ConvAI2 dataset [208] and then to fine-tune them on EmpatheticDialogues. Due to limited resources, we trained the models using small batches of sizes 8, 12 or 16.
In both approaches we used the "bert-base-uncased", "gpt2" and "t5-base" versions to start training (or pre-training). For the "Ours-Vaswani Full Transformer" models we used Transformer networks as encoder and decoder with 4 layers, 6 attention heads and 500 hidden states each. We also used a dropout rate of 0.2 and an attentional dropout rate of 0.1. We also used the $gelu()$ activation function and 1 million 300-d word vectors trained on Wikipedia 2017, UMBC webbase corpus, and statmt.org news dataset [236].
During the training process (both pre-training and fine-tuning), we trained both the word embeddings and the positional embeddings in the BERT2BERT, BERT2GPT2 and T5 models, while in the "Ours-Vaswani Full Transformer" models we trained only the word embeddings and we used positional embeddings coming from sine and cosine functions as in equation 4.15. During fine-tuning, for the

BERT2BERT, BERT2GPT2 and T5 (without multitasking) models we used the Adam optimizer with a learning rate of 2e-5, and for the "Ours-Vaswani Full Transformer" models we used the Adam optimizer with a learning rate of 2e-6. We also used weight decay equal to 1e-6, for all the aforementioned models. We should also mention here that all the models were trained with early-stopping, keeping the checkpoint with the best language model loss in the validation set.

Model 14 (T5-multitask1):
For the T5-multitask1 model we used the "t5-base" version of T5, extending it with a linear classifier. We first pre-trained the model (freezing the classifier) on ConvAI2 dataset and then we fine-tuned the T5-multitask1 model on EmpatheticDialogues dataset. During fine-tuning, we focus on two objectives:

- language modeling
- dialogue emotion detection

The decoder is used for the language modeling objective, while the classification head over the encoder, for detecting the emotion of the dialogue. Using the representations produced by the encoder and the decoder, we train the model to optimize the mentioned objectives.

- To optimize the language modeling objective, we use the representation coming from the decoder to predict the next tokens, and finally we compute the language modeling loss denoted with L_{lm} using cross-entropy.
- To optimize the emotion prediction objective, we pass the representation of the dialogue history (coming from the encoder) through the emotion classifier. We try to classify correctly the emotion of the dialogue among 32 emotions, using the cross-entropy loss, denoted with L_{em}

Finally the final fine-tuning loss used is a weighted sum of the aforementioned losses, as shown in the following equation:

$$L = L_{lm} + \alpha \times L_e \tag{5.2}$$

where $\alpha = 0.5$. During fine-tuning, we use a batch size of 8, Adam optimizer with a learning rate of 8e-5 and a weight decay of 1e-6, training both the word and the positional embeddings. Finally, we keep the model with the best language modeling loss in the validation set.

Model 15 (T5-multitask2):
For the T5-multitask2 model we used the "t5-base" version of T5, extending it with two linear classifiers, one over the encoder and one over the decoder. We fine-tune the model on EmpatheticDialogues dataset without pre-training on ConvAI2 dataset as we did with model 14. During fine-tuning, we focus on three objectives:

- language modeling
- dialogue emotion detection
- emotion detection on the generated response

The decoder is used for the language modeling objective, while the classification head over the encoder, for detecting the emotion of the dialogue. We use the classification head over the decoder to detect the emotion of the generated response. Using the representations produced by the encoder and the decoder, we train the model to optimize the mentioned objectives.

- To optimize the language modeling objective, we do exactly what we did with model 14 (T5-multitask1). Let's denote the loss with L_{lm}.

- To optimize the emotion prediction objective of the dialogue, we do exactly what we did with model 14 (T5-multitask1). Let's denote the loss with L_{em-enc}
- To optimize the emotion prediction objective over the generated response, we pass the representation coming from the decoder through the emotion classifier. We try to classify correctly the emotion of the generated response among 32 emotions, using the cross-entropy loss, denoted with L_{em-dec}. In this way, we penalise the model if the emotion of the generated response is not the appropriate (forcing empathy).

Finally the final fine-tuning loss used is a weighted sum of the aforementioned losses, as shown in the following equation:

$$L = L_{lm} + \alpha \times L_{em-enc} + \beta \times L_{em-dec} \tag{5.3}$$

where $\alpha = 0.3$ and $\beta = 0.8$. During fine-tuning, we use a batch size of 8, Adam optimizer with a learning rate of 8e-5 and a weight decay of 1e-6, training both the word and the positional embeddings. Finally, we keep the model with the best language modeling loss in the validation set.

Finally we should also mention that during inference time, we use top-p (nucleus) sampling method with top-k filtering (top-p/top-k filtering), setting threshold probability $p = 0.9$, $topk = 10$ and $temperature = 1.0$. We also add length penalty equal to 0.6 and we set the maximum length of the generated response to be equal to 40.

Experimental Evaluation:
We evaluate the models on their ability to reproduce the Listener's portion of the conversation (i.e. the ability to react to someone else's story), using automatic metrics. We compute the BLEU scores [196] for the generated response, comparing against the actual response, following the practice of earlier works in dialogue generation. Moreover, we compute and report the perplexity (PPL) of the actual response. We report those two metrics on all of the models in order to compare them with the current state-of-the-art models.

Results:
In Table 5.5, we report the results for all of the generative models described in the previous sections. In Table 5.6 we present a summary of state-of-the-art models trained on EmpatheticDialogues dataset.

As it concerns the baseline architectures, we notice that the CAiRE model outperformed all the models proposed by [26, 25], but finally the DodecaDialogue MT+FT and DodecaDialogue MT models performed state-of-the-art results. More specifically, DodecaDialogue MT+FT model achieved a state-of-the-art perplexity of 11.4, making clear that multi-tasking learning in different tasks boosts the model's performance. So, training a conversational agent with the view of being able to handle multiple tasks and having multiple skills, makes the agent generalizable, being able to handle appropriately each situation with giving accurate responses. As it concerns our experiments, we should note first that the Vaswani Full Transformer models failed in producing satisfying results. We can easily notice that both of the models, that we implemented, have a huge difference in perplexity, compared with the original one [26, 25]. More specifically, "Ours-Vaswani Full Transformer (ED)" and "Ours-Vaswani Full Transformer (P+ED)" models have perplexity of 33.46 and 28.46 respectively, while the official baseline model has 21.24. This difference makes sense, as the official baseline model was pre-trained on the Reddit dataset, consisting of 1.7 billion conversations, while ours were pre-trained on ConvAI2 dataset having significantly fewer conversations. Pre-training from scratch a model having approximately 126 million parameters, on such a small dataset, makes it difficult to model the natural language and to generate accurate responses. Unfortunatelly, we did not have the ability to use the Reddit dataset to produce comparable results due to limited resources availability. Despite that fact, the rest of the models that were used, were successfully trained as those models were already pre-trained in large datasets. Both BERT2BERT and BERT2GPT2 produce comparable results

Table 5.5: Summary of results of all experiments

Model	PPL	AVG BLEU
Vaswani Full Transformer [26, 25]	21.24	6.27
Multitask Transformer [26, 25]	24.07	5.42
EmoPrepend-1 [26, 25]	24.30	4.36
TopicPrepend-1 [26, 25]	25.40	4.17
Ensem-DM [26, 25]	19.05	6.83
Ensem-DM+ [26, 25]	19.10	6.77
CAiRE [27]	13.32	7.03
GPT2-baseline [36]	18.32*	7.71*
GPT2-prepend [36]	19.49*	7.78*
BST Generative [35]	11.48*	-
DodecaDialogue MT+FT [37]	**11.4**	8.1
DodecaDialogue MT [37]	11.5	8.4
Ours-Vaswani Full Transformer (ED)	33.46	-
Ours-Vaswani Full Transformer (P+ED)	28.64	-
BERT2BERT (ED)	20.77	5.53
BERT2BERT (P+ED)	19.54	6.78
BERT2GPT2 (ED)	17.93	7.22
BERT2GPT2 (P+ED)	21.48	7.19
T5 (ED)	12.40	9.31
T5 (P+ED)	12.51	**9.68**
T5-multitask1	12.58	9.28
T5-multitask2	12.96	9.13

* denotes results reported on validation set

with the baselines of [26, 25]. The BERT2GPT2(ED) model outperforms both in perplexity and in average BLEU score the baselines of [26, 25]. Moreover, it has better perplexity than the baselines proposed by [36], however it has worse average BLEU score. We should also note here that with a pre-training on a larger dataset, such as the Reddit dataset, the BERT2BERT and BERT2GPT2 models may lead to significantly better results, as the pretraining on ConvAI2 dataset does not improve the models' performance due to its limited size. However, pre-training boosts the average BLEU score (in most cases), as the models generate more diverse responses. As it concerns the approach of using the T5 model, we clearly notice that there is a significant improvement compared with the previously proposed models. Both the simple and multitask learning architectures, using the T5 model, perform state-of-the-art results concerning the average BLEU metric. More specifically, the T5 (P+ED) model achieves close results in perplexity to the current state-of-the-art model (DodecaDialogue MT+FT), having a 9.7% difference, and it outperforms the current state-of-the-art model (DodecaDialogue MT) in average BLEU metric , by a difference of 19.5%.

Having studied the aforementioned results we can now draw some high-level conclusions. First of all, we come to the conclusion that pretraining is really important for creating successful conversational agents, especially when pretraining is done on large conversational data. Moreover, in most circumstances, we notice that seq2seq models seem to perform better in dialog generation tasks rather than auto-regressive models, as seq2seq architectures can be used for successful conditional generation. Furthermore, it is also worth mentioning that multi-task learning on various dialog tasks enhances the conversational agent, having multiple dialog skills. However, multi-task learning techniques for emotion classification may not always improve system's performance.

Table 5.6: Summary of results of state-of-the-art models

Model	PPL	AVG BLEU
BST Generative [35]	11.48*	-
DodecaDialogue MT+FT [37]	**11.4**	8.1
DodecaDialogue MT [37]	11.5	8.4
T5 (ED)	12.40	9.31
T5 (P+ED)	12.51	**9.68**
T5-multitask1	12.58	9.28
T5-multitask2	12.96	9.13

* denotes results reported on validation set

Finally, we should note that perplexity is a metric that measures how well a model can predict the test-set samples, without "real" generation. Of course, more than one responses can be suitable and emotionally relevant to a dialogue history-context. So, judging a conversational agent only from the view of the perplexity is not objective. However, the average BLEU metric measures the quality of the response using the "real" generation method. All of our models, based on T5, perform state-of-the-art results in average BLEU metric while having comparable perplexity with the current state-of-the-art.

In the following, we provide some of the responses provided by the T5 (P+ED) model compared with the actual responses in Figures 5.9, 5.10, 5.11, 5.12 and 5.13.

Example 1:
Emotion: excited
Situation: I love golf and watched the Scottish Open recently. I know a guy who was playing in it and he won! What a fantastic four days of watching and following him.

Conversation:
Speaker: I was watching the Scottish Open golf tournament a few weeks ago. I know a guy who was playing in it. Guess what? He won! Fantastic four days of baited breath and cheering him on!
Listener: That is great! Can you call him and see if we could all celebrate together?
Speaker: He's playing in the PGA Championship right now or I'm sure he'd be delighted to, lol!
Generated Response: I am sure he will! I hope he wins!
Actual Response: Well yeah, that sounds about right. He has to go to the Championship first and win, then party time!!!!!

Figure 5.9: Example 1 of generation process.

Example 2:
Emotion: content
Situation: I paid all my bills, I feel great

Conversation:
Speaker: I paid all my bills today, i feel so relived and good
Generated Response: That's great! How much did you have to pay?
Actual Response: Every little accomplishment counts! What bills did you have to pay?

Figure 5.10: Example 2 of generation process.

Example 3:
Emotion: sad
Situation: I read in the news a family had a house fire and lost all of their animals. It makes me upset to think about how they died.

Conversation:
Speaker: I read in the news about a family who lost all six of their animals in a house fire :(
Generated Response: Oh no! I'm so sorry to hear that. What happened?
Actual Response: thats so sad oh no! thats terrible. :(

Figure 5.11: Example 3 of generation process.

Example 4:
Emotion: angry
Situation: I was in charge of moving a whole bunch of stuff at work from one building to another, and one of my co-workers bailed on us.

Conversation:

Speaker: One of the things I had to do when I started my job was be in charge of moving a department. When we were doing it, one of my coworkers bailed on me.
Generated Response: That is terrible. How did you handle it?
Actual Response: Did it leave you to do it alone, or were there others?

Figure 5.12: Example 4 of generation process.

Example 5:
Emotion: guilty
Situation: when i was young i hit a car on the parking lot and left without saying anynthing beacuse i did not see a mark on the other car

Conversation:
Speaker: when i was young i hit a car on the parking lot and did not say anything'
Generated Response: Oh no! Did you get hurt?
Actual Response: Why didn't you say anything?

Figure 5.13: Example 5 of generation process.

The above results show that the model is able to reproduce not only syntactically and grammatically coherent responses, but also to express the appropriate emotion. We clearly notice the strong ability of modeling the user's emotions and generating empathetic and engaging responses, with the T5 based architectures that we used.

5.7 Summary

In this chapter we studied in depth dialogue generation, not only focusing on the part of generating syntactically and grammatically correct responses, but producing responses that will vary in emotional content, thus engaging the user. More specifically, we focused on creating dialogue agents, using generative models, that will be empathetic or in other words they will be emotionally relevant responses with the user's emotion. We used the EmpatheticDialogues dataset, a dataset focusing on conversations with empathy, and the ConvAI2 dataset to enrich the context of conversations. We also studied the recent work done in the field, using the most famous models in language modeling, the

Transformers and their expansions. We also applied variants of transformers in the EmpatheticDialogues dataset, achieving state-of-the-art results in the average BLEU metric while achieving close perplexities to the current state-of-the-art models. Moreover, we introduced a novel architecture for modeling empathy using multitask learning, which improved the current state-of-the-art BLEU metric, but did not perform as well as the non-multitasking approach (using the T5 model as is). Finally, we presented some of the generated responses of our best model, noticing that the generated responses, not only seem to be fluent and coherent, but also to be rich in emotional context, relevant with that of the user.

Chapter 6

Epilogue

6.1 Conclusions

In this diploma thesis we studied in depth the work done in the field of creating empathetic conversational agents using generation-based models and we also proposed ways to further improve upon these systems. More specifically, at first, we analyzed the traditional architectures used for dialogue generation, including the vanilla seq2seq model, its expansion with the attention mechanism, and the HRED model. Then, we studied the state-of-the-art models that can be used in dialogue generation, including the Vaswani encoder-decoder transformer [3], the Bert, the GPT2 and the T5 models. After providing a theoretical background for the aforementioned models we focused on the EmpatheticDialogues task, a task proposed by Facebook for building empathetic dialog systems.

After presenting and studying the related work on the task we conducted several experiments, testing various architectures for improving the results on the task further. The experiments conducted with the use of the BERT2BERT and BERT2GPT2 models did not improve the state-of-the-art results. However, the experiments based on the T5 architectures provided state-of-the-art results concerning the BLEU metric, while achieving perplexity close to that of current state-of-the-art models. More specifically, we experimented with three different architectures.

- The first one, which is our baseline architecture is to use The T5 model as is, on the EmpatheticDialogues task.
- The second one, extends the baseline model with multi-task learning, adding over the encoder a classification head. The goal of this extension is to better understand the implied feelings of the conversation.
- The third one extends the second by adding a classification head over the decoder too. In this way, we aim to model empathy by indirectly forcing the decoder to generate a response having the same emotion as the rest of the conversation (that is extracted by the classification head over the encoder). We achieve this, by using the same emotion label while training both emotion classifiers.

Finally, all the T5-based architectures proposed, improved the state-of-the-art BLEU score, with the baseline providing the best among all, improving the current state-of-the-art model (DodecaDialogue MT) in average BLEU metric by 19.5%. Furthermore, the baseline architecture achieved close results in perplexity to the current state-of-the-art model (DodecaDialogue MT+FT), having a difference of 9.7%.

We should also note here that perplexity is a metric that measures how well a model can predict the test-set samples, without "real" generation, while the BLEU score measures the quality of the response using "real" generation. So, using only the perplexity metric for evaluation is not really objective. However, the BLEU metric is a metric usually applied for machine translation tasks, so we can not rely only on this one. From this point of view, in the next section, we propose future work concerning both the evaluation and the modification of the presented architectures.

6.2 Future Work

In order to further improve our work, in this section we refer to future extensions and modifications for future study. More specifically, we suggest to:

- Train the T5-based models in various dialogue tasks, as it was done in [37, 35]. More specifically, we can train the models using multi-tasking in more dialogue tasks, such as the EmpatheticDialogues, the ConvAI2, the WoW and the BST tasks, and then fine-tune them on EmpatheticDialogues. In this way, the models gain various skills and become able to generate more appropriate responses.
- Use pre-trained classifiers for emotion classification to better understand the implied feelings of the conversation.
- Extract emotion representations, in a d-dimensional space, through the use of emotion classifiers instead of extracting a simple emotion. Then, we can use both the emotion representations extracted from the encoder and the decoder, and add an auxiliary loss to better model empathy using one of the following methods:

  1. Calculate the distance between the emotion representations using $p-norm$ (e.g. calculate euclidean distance using $p = 2$) in order to measure their similarity. The less the distance, the more empathetic the model.
  2. Calculate the triplet margin loss, which is given by the following equation for each sample in the mini-batch:

$$\mathrm{L}(\mathrm{a}, \mathrm{p}, \mathrm{n}) = \max\left\{\mathrm{d}\left(\mathrm{a_i}, \mathrm{p_i}\right) - \mathrm{d}\left(\mathrm{a_i}, \mathrm{n_i}\right) + \mathrm{margin}, 0\right\} \tag{6.1}$$

  where a denotes the anchor, p denotes a positive example, n denotes a negative example and $\mathrm{d}\left(\mathrm{x_i}, \mathrm{y_i}\right) = \|\mathbf{x_i} - \mathbf{y_i}\|_{\mathrm{k}}$ with k denoting the degree of the pairwise distance. The emotion representation coming from the decoder is set as the anchor, while the one coming from the encoder is set as the positive example. We also sample one "negative" emotion label and we set it as the negative example. In this way, we have three different situations for the calculated loss:

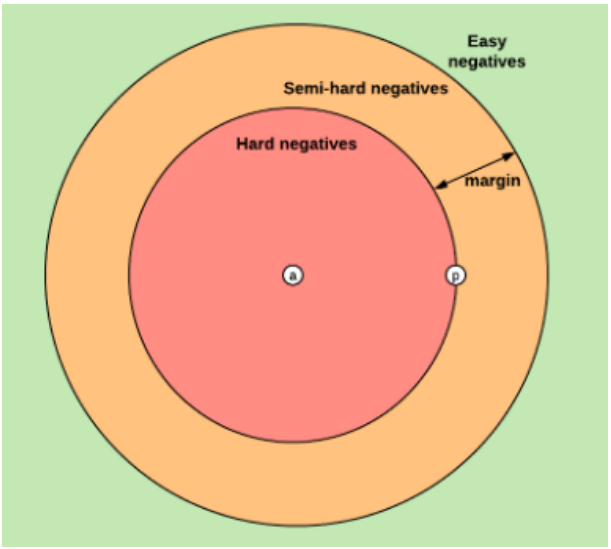


Figure 6.1: An illustration of triplet margin loss.

  – **Easy triplets**: $d(\mathrm{a_i}, \mathrm{n_i}) > d(\mathrm{a_i}, \mathrm{p_i}) + margin$ The negative sample is already sufficiently distant from the anchor sample with respect to the positive sample in the emotion representation space, so the calculated loss is 0.

- **Hard triplets**: $d(\mathrm{a_i}, \mathrm{n_i}) < d(\mathrm{a_i}, \mathrm{p_i})$ The negative sample is closer to the anchor than the positive. The loss is positive (and greater than $margin$).
- **Semi-Hard triplets**: $d(\mathrm{a_i}, \mathrm{p_i}) < d(\mathrm{a_i}, \mathrm{n_i}) < d(\mathrm{a_i}, \mathrm{p_i}) + margin$ The negative sample is more distant from the anchor than the positive, but the distance is not greater than the $margin$, so the loss is still positive (and smaller than $margin$).

An illustration of the above loss is shown in Figure 6.1. In this way, the smaller the computed loss, the more empathetic the model.

- Have humans evaluate the generated dialogues. More specifically, we propose grading the generated responses concerning relevance, fluency, coherency and empathy, and also doing a pairwise comparison between the responses of the current state-of-the-art models (DodecaDialogue MT, DodecaDialogue MT+FT) and the proposed ones (T5, T5-multitask1, T5-multitask2).
- Finally, we also suggest doing an ablation study over the models, in order to better understand the behavior of the constituent components.

Bibliography

[1] "Write a sequence to sequence (seq2seq) model." [Online]. Available: https://docs.chainer.org/en/stable/examples/seq2seq.html

[2] R. Karim, "Attn: Illustrated attention." [Online]. Available: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv e-prints*, p. arXiv:1706.03762, Jun. 2017.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints*, p. arXiv:1810.04805, Oct. 2018.

[5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *arXiv e-prints*, p. arXiv:1910.10683, Oct. 2019.

[6] A. Sivalingam, "What is machine learning?" [Online]. Available: https://medium.com/swlh/what-is-machine-learning-9b569ff7858a

[7] A. Karpathy, "Convolutional neural networks for visual recognition." [Online]. Available: https://cs231n.github.io/neural-networks-1/

[8] S. Kampakis, "What deep learning is and isn't." [Online]. Available: https://thedatascientist.com/what-deep-learning-is-and-isnt/

[9] C. Olah, "Understanding lstm networks." [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10] ——, "Neural networks, types, and functional programming." [Online]. Available: https://colah.github.io/posts/2015-09-NN-Types-FP/

[11] M. Thapliyal, "Deep learning basics: Gated recurrent unit (gru)." [Online]. Available: https://mc.ai/deep-learning-basics-gated-recurrent-unit-gru/

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv e-prints*, p. arXiv:1409.0473, Sep. 2014.

[13] J. Cheng, L. Dong, and M. Lapata, "Long Short-Term Memory-Networks for Machine Reading," *arXiv e-prints*, p. arXiv:1601.06733, Jan. 2016.

[14] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," *arXiv e-prints*, p. arXiv:1508.04025, Aug. 2015.

[15] S. Ruber, "An overview of multi-task learning in deep neural networks." [Online]. Available: https://ruder.io/multi-task/

[16] N. S. Sarwan, "An intuitive understanding of word embeddings: From count vectors to word2vec." [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv e-prints*, p. arXiv:1301.3781, Jan. 2013.

[18] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[19] S. C. Chris Olah, "Attention and augmented recurrent neural networks." [Online]. Available: https://distill.pub/2016/augmented-rnns/#attentional-interfaces

[20] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models," *arXiv e-prints*, p. arXiv:1507.04808, Jul. 2015.

[21] J. Alammar, "The illustrated transformer." [Online]. Available: http://jalammar.github.io/illustrated-transformer/

[22] ——, "The illustrated gpt-2." [Online]. Available: http://jalammar.github.io/illustrated-gpt2/

[23] ——, "The illustrated bert." [Online]. Available: http://jalammar.github.io/illustrated-bert/

[24] P. von Platen, "How to generate text: using different decoding methods for language generation with transformers." [Online]. Available: https://huggingface.co/blog/how-to-generate

[25] H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau, "Towards Empathetic Open-domain Conversation Models: a New Benchmark and Dataset," *arXiv e-prints*, p. arXiv:1811.00207, Oct. 2018.

[26] H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau, "I know the feeling: Learning to converse with empathy," *ArXiv*, vol. abs/1811.00207, 2018.

[27] Z. Lin, P. Xu, G. Indra Winata, F. B. Siddique, Z. Liu, J. Shin, and P. Fung, "CAiRE: An Empathetic Neural Chatbot," *arXiv e-prints*, p. arXiv:1907.12108, Jul. 2019.

[28] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, p. 36–45, Jan. 1966. [Online]. Available: https://doi.org/10.1145/365153.365168

[29] K. M. Colby, "Modeling a paranoid mind," *Behavioral and Brain Sciences*, vol. 4, no. 4, p. 515–534, 1981.

[30] L. Zhou, J. Gao, D. Li, and H.-Y. Shum, "The Design and Implementation of XiaoIce, an Empathetic Social Chatbot," *arXiv e-prints*, p. arXiv:1812.08989, Dec. 2018.

[31] B. Abushawar and E. Atwell, "Alice chatbot: Trials and outputs," *Computación y Sistemas*, vol. 19, 12 2015.

[32] Y. Wu, W. Wu, Z. Li, and M. Zhou, "Response Selection with Topic Clues for Retrieval-based Chatbots," *arXiv e-prints*, p. arXiv:1605.00090, Apr. 2016.

[33] R. E. Banchs and H. Li, "IRIS: a chat-oriented dialogue system based on the vector space model," in *Proceedings of the ACL 2012 System Demonstrations*. Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 37–42. [Online]. Available: https://www.aclweb.org/anthology/P12-3007

[34] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le, "Towards a Human-like Open-Domain Chatbot," *arXiv e-prints*, p. arXiv:2001.09977, Jan. 2020.

[35] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, Y.-L. Boureau, and J. Weston, "Recipes for building an open-domain chatbot," *arXiv e-prints*, p. arXiv:2004.13637, Apr. 2020.

[36] S. Santhanam and S. Shaikh, "Emotional Neural Language Generation Grounded in Situational Contexts," *arXiv e-prints*, p. arXiv:1911.11161, Nov. 2019.

[37] K. Shuster, D. Ju, S. Roller, E. Dinan, Y.-L. Boureau, and J. Weston, "The Dialogue Dodecathlon: Open-Domain Knowledge and Image Grounded Conversational Agents," *arXiv e-prints*, p. arXiv:1911.03768, Nov. 2019.

[38] D. Jurafsky and J. H. Martin, "Dialog systems and chatbots," *Speech and language processing*, vol. 3, 2017.

[39] G. Qiu, X. He, F. Zhang, Y. Shi, J. Bu, and C. Chen, "Dasa: Dissatisfaction-oriented advertising based on sentiment analysis," *Expert Systems with Applications*, vol. 37, pp. 6182–6191, 09 2010.

[40] M. Itani, C. Roast, and S. Al-Khayatt, "Developing resources for sentiment analysis of informal arabic text in social media," *Procedia Computer Science*, vol. 117, pp. 129 – 136, 2017, arabic Computational Linguistics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917321580

[41] V. Stoyanov, C. Cardie, and J. Wiebe, "Multi-perspective question answering using the opqa corpus." 01 2005.

[42] D. Bertero, F. B. Siddique, C.-S. Wu, Y. Wan, R. H. Y. Chan, and P. Fung, "Real-time speech emotion and sentiment recognition for interactive dialogue systems," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1042–1047. [Online]. Available: https://www.aclweb.org/anthology/D16-1110

[43] A. Chatterjee, K. N. Narahari, M. Joshi, and P. Agrawal, "SemEval-2019 task 3: EmoContext contextual emotion detection in text," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 39–48. [Online]. Available: https://www.aclweb.org/anthology/S19-2005

[44] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu, "Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory," *arXiv e-prints*, p. arXiv:1704.01074, Apr. 2017.

[45] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward Controlled Generation of Text," *arXiv e-prints*, p. arXiv:1703.00955, Mar. 2017.

[46] K. Wang and X. Wan, "Sentigan: Generating sentimental texts via mixture adversarial networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 4446–4452. [Online]. Available: https://doi.org/10.24963/ijcai.2018/618

[47] W. Levinson, R. Gorawara-Bhat, and J. Lamb, "A study of patient clues and physician responses in primary care and surgical settings," *JAMA : the journal of the American Medical Association*, vol. 284, pp. 1021–7, 11 1999.

[48] J. Fraser, I. Papaioannou, and O. Lemon, "Spoken conversational ai in video games-emotional dialogue management increases user engagement," 10 2018.

[49] S. S. Kim, S. Kaplowitz, and M. Johnston, "The effects of physician empathy on patient satisfaction and compliance," *Evaluation the health professions*, vol. 27, pp. 237–51, 10 2004.

[50] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[51] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[52] S. Russel *et al.*, *Artificial intelligence: a modern approach*.

[53] G. E. Hinton, T. J. Sejnowski, T. A. Poggio *et al.*, *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[54] X. Zhu, "Semi-supervised learning literature survey," *Comput Sci, University of Wisconsin-Madison*, vol. 2, 07 2008.

[55] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[56] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. USA: Prentice Hall PTR, 2000.

[57] R. HECHT-NIELSEN, "Iii.3 - theory of the backpropagation neural network**based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 ieee." in *Neural Networks for Perception*, H. Wechsler, Ed. Academic Press, 1992, pp. 65 – 93. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780127412528500108

[58] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 2004.

[59] O. Veksler, "Cs4341/541a lecture notes." [Online]. Available: http://www.cs.haifa.ac.il/~rita/ml_course/lectures/SVM.pdf

[60] A. Ng, "Cs229 lecture notes." [Online]. Available: https://akademik.bahcesehir.edu.tr/~tevfik/courses/cmp5101/cs229-notes1.pdf

[61] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[62] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[63] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[64] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[65] P. A. Rutecki, "Neuronal excitability: voltage-dependent currents and synaptic transmission," *Journal of clinical neurophysiology*, vol. 9, no. 2, pp. 195–211, 1992.

[66] W. J. Freeman, "Three centuries of category errors in studies of the neural basis of consciousness and intentionality," *Neural Networks*, vol. 10, no. 7, pp. 1175–1183, 1997.

[67] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[68] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," *arXiv e-prints*, p. arXiv:1606.08415, Jun. 2016.

[69] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[70] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[71] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, p. 318–362.

[72] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[73] J. Zhang, "Gradient Descent based Optimization Algorithms for Deep Learning Models Training," *arXiv e-prints*, p. arXiv:1903.03614, Mar. 2019.

[74] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[75] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.

[76] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 07 2011.

[77] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *arXiv e-prints*, p. arXiv:1212.5701, Dec. 2012.

[78] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München," 1991.

[79] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[80] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," *arXiv e-prints*, p. arXiv:1211.5063, Nov. 2012.

[81] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, S. C. Kremer and J. F. Kolen, Eds. IEEE Press, 2001.

[82] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, p. 1735—1780, November 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[83] F. Gers, "Learning to forget: continual prediction with lstm," *IET Conference Proceedings*, pp. 850–855(5). [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/cp_19991218

[84] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv e-prints*, p. arXiv:1406.1078, Jun. 2014.

[85] G. Weiss, Y. Goldberg, and E. Yahav, "On the Practical Computational Power of Finite Precision RNNs for Language Recognition," *arXiv e-prints*, p. arXiv:1805.04908, May 2018.

[86] L. Weng, "Attention?attention!" [Online]. Available: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html?fbclid=IwAR1WZv3cGAlMm7AA6yWq8SlE1ZmTmWBqLYUcoGBDNFXX-nWP1yBG3QZ7omU#whats-wrong-with-seq2seq-model

[87] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 3104–3112.

[88] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," *arXiv e-prints*, p. arXiv:1410.5401, Oct. 2014.

[89] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," *arXiv e-prints*, p. arXiv:1502.03044, Feb. 2015.

[90] S. Ruder, "Deep learning for nlp best practices." [Online]. Available: https://ruder.io/deep-learning-nlp-best-practices/

[91] R. Paulus, C. Xiong, and R. Socher, "A Deep Reinforced Model for Abstractive Summarization," *arXiv e-prints*, p. arXiv:1705.04304, May 2017.

[92] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A Decomposable Attention Model for Natural Language Inference," *arXiv e-prints*, p. arXiv:1606.01933, Jun. 2016.

[93] Z. Li, Y. Li, and H. Lu, *Improve Image Captioning by Self-attention*, 12 2019, pp. 91–98.

[94] Z. Lin, M. Feng, C. Nogueira dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A Structured Self-attentive Sentence Embedding," *arXiv e-prints*, p. arXiv:1703.03130, Mar. 2017.

[95] A. Singh, "Brief introduction to attention models." [Online]. Available: https://towardsdatascience.com/attention-networks-c735befb5e9f

[96] K. Gregor, I. Danihelka, A. Graves, D. Jimenez Rezende, and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," *arXiv e-prints*, p. arXiv:1502.04623, Feb. 2015.

[97] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive Exploration of Neural Machine Translation Architectures," *arXiv e-prints*, p. arXiv:1703.03906, Mar. 2017.

[98] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[99] J. West, D. Ventura, and S. Warnick, "Spring research presentation: A theoretical foundation for inductive transfer," *Brigham Young University, College of Physical and Mathematical Sciences*, vol. 1, no. 08, 2007.

[100] J. Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.

[101] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Advances in neural information processing systems*, 1996, pp. 640–646.

[102] S. Dhuria, "Analysis : An approach in natural language processing for data extraction," 2015.

[103] O. Enayet, "Natural language processing – the big picture." [Online]. Available: https://omarsbrain.wordpress.com/tag/natural-language-processing-linguistics-phonology-morphology-discourse-pragmatic-summarization/

[104] R. C. Major, *Phonological Analysis*. American Cancer Society, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781405198431.wbeal0908

[105] *Morphology Sphere*. John Wiley Sons, Ltd, 2016, ch. 3, pp. 89–125. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119145554.ch3

[106] *Syntax Sphere*. John Wiley Sons, Ltd, 2016, ch. 4, pp. 127–243. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119145554.ch4

[107] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[108] A. Miranda-García and J. Calle-Martín, "The authorship of the disputed federalist papers with an annotated corpus," *English Studies*, vol. 93, no. 3, pp. 371–390, 2012. [Online]. Available: https://doi.org/10.1080/0013838X.2012.668795

[109] Z. S. Harris, "Distributional structure," WORD, vol. 10, no. 2-3, pp. 146–162, 1954. [Online]. Available: https://doi.org/10.1080/00437956.1954.11659520

[110] J. R. Firth, "A synopsis of linguistic theory 1930-55." vol. 1952-59, pp. 1–32, 1957.

[111] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (almost) from Scratch," *arXiv e-prints*, p. arXiv:1103.0398, Mar. 2011.

[112] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *arXiv e-prints*, p. arXiv:1310.4546, Oct. 2013.

[113] J. Goodman, "Classes for Fast Maximum Entropy Training," *arXiv e-prints*, p. cs/0108006, Aug. 2001.

[114] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[115] T. Landauer, P. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, pp. 259–284, 1998.

[116] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in Translation: Contextualized Word Vectors," *arXiv e-prints*, p. arXiv:1708.00107, Jul. 2017.

[117] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv e-prints*, p. arXiv:1802.05365, Feb. 2018.

[118] K. Church, "Dedication to william a. gale," *Nat. Lang. Eng.*, vol. 8, no. 4, p. 275–277, Dec. 2002. [Online]. Available: https://doi.org/10.1017/S1351324902003066

[119] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 181–184 vol.1, 1995.

[120] H. Gunes, J. Vallverdú, and M. Pantic, "Automatic, dimensional and continuous emotion recognition," *International journal of synthetic emotions*, vol. 1, no. 1, pp. 68–99, 1 2010, 10.4018/jse.2010101605.

[121] D. Grandjean, D. Sander, and K. Scherer, "Conscious emotional experience emerges as a function of multilevel, appraisal-driven response synchronization," *Consciousness and cognition*, vol. 17, pp. 484–95, 07 2008.

[122] M. Piórkowska and M. Wrobel, *Basic Emotions*, 07 2017.

[123] N. Alswaidan and M. Menai, "A survey of state-of-the-art approaches for emotion recognition in text," *Knowledge and Information Systems*, 03 2020.

[124] C. Ma, H. Prendinger, and M. Ishizuka, "Emotion estimation and reasoning based on affective textual interaction," in *Proceedings of the First International Conference on Affective Computing and Intelligent Interaction*, ser. ACII'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 622–628. [Online]. Available: https://doi.org/10.1007/11573548_80

[125] I. Perikos and I. Hatzilygeroudis, "Recognizing emotion presence in natural language sentences," in *Engineering Applications of Neural Networks*, L. Iliadis, H. Papadopoulos, and C. Jayne, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 30–39.

[126] S. Y. M. Lee, Y. Chen, and C.-R. Huang, "A text-driven rule-based system for emotion cause detection," in *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. Los Angeles, CA: Association for Computational Linguistics, Jun. 2010, pp. 45–53. [Online]. Available: https://www.aclweb.org/anthology/W10-0206

[127] C. O. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, Oct. 2005, pp. 579–586. [Online]. Available: https://www.aclweb.org/anthology/H05-1073

[128] S. Aman and S. Szpakowicz, "Identifying expressions of emotion in text," 09 2007, pp. 196–205.

[129] Y. Wang, S. Feng, D. Wang, G. Yu, and Y. Zhang, "Multi-label chinese microblog emotion classification via convolutional neural network," in *APWeb*, 2016.

[130] H. Meisheri and L. Dey, "Tcs research at semeval-2018 task 1: Learning robust representations using multi-attention architecture," in *SemEval@NAACL-HLT*, 2018.

[131] C. Baziotis, A. Nikolaos, A. Chronopoulou, A. Kolovou, G. Paraskevopoulos, N. Ellinas, S. Narayanan, and A. Potamianos, "NTUA-SLP at SemEval-2018 task 1: Predicting affective content in tweets with deep attentive RNNs and transfer learning," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 245–255. [Online]. Available: https://www.aclweb.org/anthology/S18-1037

[132] K. Shrivastava, S. Kumar, and D. K. Jain, "An effective approach for emotion detection in multimedia text data using sequence based convolutional neural network," *Multimedia Tools and Applications*, vol. 78, pp. 29 607 – 29 639, 2019.

[133] W. Ragheb, J. Azé, S. Bringay, and M. Servajean, "Attention-based Modeling for Emotion Detection and Classification in Textual Conversations," *arXiv e-prints*, p. arXiv:1906.07020, Jun. 2019.

[134] C. Huang, A. Trabelsi, and O. R. Zaïane, "ANA at SemEval-2019 Task 3: Contextual Emotion detection in Conversations through hierarchical LSTMs and BERT," *arXiv e-prints*, p. arXiv:1904.00132, Mar. 2019.

[135] Y.-S. Seol, D. Kim, and H.-W. Kim, "Emotion recognition from text using knowledge-based ann," 2008.

[136] S. Gievska, K. Koroveshovski, and T. Chavdarova, "A hybrid approach for emotion detection in support of affective interaction," in *2014 IEEE International Conference on Data Mining Workshop*, 2014, pp. 352–359.

[137] S. Shaheen, W. El-Hajj, H. Hajj, and S. Elbassuoni, "Emotion recognition from text based on automatically generated rules," in *2014 IEEE International Conference on Data Mining Workshop*, 2014, pp. 383–392.

[138] R. Yan, Y. Song, and H. Wu, "Learning to respond with deep neural networks for retrieval-based human-computer conversation system," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 55–64. [Online]. Available: https://doi.org/10.1145/2911451.2911542

[139] R. T. Lowe, N. Pow, I. Serban, L. Charlin, C.-W. Liu, and J. Pineau, "Training end-to-end dialogue systems with the ubuntu dialogue corpus," *Dialogue Discourse*, vol. 8, pp. 31–65, 2017.

[140] E. H. Almansor and F. K. Hussain, "Survey on intelligent chatbots: State-of-the-art and future research directions," in *CISIS*, 2019.

[141] Z. Ji, Z. Lu, and H. Li, "An Information Retrieval Approach to Short Text Conversation," *arXiv e-prints*, p. arXiv:1408.6988, Aug. 2014.

[142] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," *CoRR*, vol. abs/1503.02364, 2015. [Online]. Available: http://arxiv.org/abs/1503.02364

[143] A. Ritter, C. Cherry, and W. B. Dolan, "Data-driven response generation in social media," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, Jul. 2011, pp. 583–593. [Online]. Available: https://www.aclweb.org/anthology/D11-1054

[144] A. Ritter, C. Cherry, and B. Dolan, "Unsupervised modeling of twitter conversations," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, Jun. 2010, pp. 172–180. [Online]. Available: https://www.aclweb.org/anthology/N10-1020

[145] W. Zhang, T. Liu, Y. Wang, and Q. Zhu, "Neural personalized response generation as domain adaptation," *World Wide Web*, 01 2017.

[146] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1700–1709. [Online]. Available: https://www.aclweb.org/anthology/D13-1176

[147] O. Dušek and F. Jurčíček, "Training a natural language generator from unaligned data," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 451–461. [Online]. Available: https://www.aclweb.org/anthology/P15-1044

[148] O. Dušek and F. Jurčíček, "Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings," *arXiv e-prints*, p. arXiv:1606.05491, Jun. 2016.

[149] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, p. 2673–2681, Nov. 1997. [Online]. Available: https://doi.org/10.1109/78.650093

[150] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, "A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion," *arXiv e-prints*, p. arXiv:1507.02221, Jul. 2015.

[151] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan, "A Neural Network Approach to Context-Sensitive Generation of Conversational Responses," *arXiv e-prints*, p. arXiv:1506.06714, Jun. 2015.

[152] I. Vlad Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio, "A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues," *arXiv e-prints*, p. arXiv:1605.06069, May 2016.

[153] M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley, "A Knowledge-Grounded Neural Conversation Model," *arXiv e-prints*, p. arXiv:1702.01932, Feb. 2017.

[154] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-To-End Memory Networks," *arXiv e-prints*, p. arXiv:1503.08895, Mar. 2015.

[155] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep Reinforcement Learning for Dialogue Generation," *arXiv e-prints*, p. arXiv:1606.01541, Jun. 2016.

[156] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial Learning for Neural Dialogue Generation," *arXiv e-prints*, p. arXiv:1701.06547, Jan. 2017.

[157] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," *arXiv e-prints*, p. arXiv:1609.05473, Sep. 2016.

[158] T. Wolf, V. Sanh, J. Chaumond, and C. Delangue, "TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents," *arXiv e-prints*, p. arXiv:1901.08149, Jan. 2019.

[159] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio, "Professor Forcing: A New Algorithm for Training Recurrent Networks," *arXiv e-prints*, p. arXiv:1610.09038, Oct. 2016.

[160] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas, "Learning where to Attend with Deep Architectures for Image Tracking," *arXiv e-prints*, p. arXiv:1109.3737, Sep. 2011.

[161] D. Britz, "Attention and memory in deep learning and nlp." [Online]. Available: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

[162] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order Boltzmann machine," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1243–1251. [Online]. Available: http://papers.nips.cc/paper/4089-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine.pdf

[163] J. Huang and E. N. Efthimiadis, "Analyzing and evaluating query reformulation strategies in web search logs," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, ser. CIKM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 77–86. [Online]. Available: https://doi.org/10.1145/1645953.1645966

[164] A. Graves, "Generating Sequences With Recurrent Neural Networks," *arXiv e-prints*, p. arXiv:1308.0850, Aug. 2013.

[165] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv e-prints*, p. arXiv:1512.03385, Dec. 2015.

[166] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv e-prints*, p. arXiv:1607.06450, Jul. 2016.

[167] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," *arXiv e-prints*, p. arXiv:1705.03122, May 2017.

[168] O. Press and L. Wolf, "Using the Output Embedding to Improve Language Models," *arXiv e-prints*, p. arXiv:1608.05859, Aug. 2016.

[169] A. Radford, "Improving language understanding by generative pre-training," 2018.

[170] A. M. Dai and Q. V. Le, "Semi-supervised Sequence Learning," *arXiv e-prints*, p. arXiv:1511.01432, Nov. 2015.

[171] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification," *arXiv e-prints*, p. arXiv:1801.06146, Jan. 2018.

[172] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism Quarterly*, vol. 30, no. 4, pp. 415–433, 1953. [Online]. Available: https://doi.org/10.1177/107769905303000401

[173] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. S. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *ArXiv*, vol. abs/1609.08144, 2016.

[174] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," 01 2008, pp. 1096–1103.

[175] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. USA: IEEE Computer Society, 2015, p. 19–27. [Online]. Available: https://doi.org/10.1109/ICCV.2015.11

[176] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context," *arXiv e-prints*, p. arXiv:1901.02860, Jan. 2019.

[177] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," *arXiv e-prints*, p. arXiv:1906.08237, Jun. 2019.

[178] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "HuggingFace's Transformers: State-of-the-art Natural Language Processing," *arXiv e-prints*, p. arXiv:1910.03771, Oct. 2019.

[179] HuggingFace, "Transformers." [Online]. Available: https://huggingface.co/transformers/index.html

[180] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating Wikipedia by Summarizing Long Sequences," *arXiv e-prints*, p. arXiv:1801.10198, Jan. 2018.

[181] R. Al-Rfou, D. Choe, N. Constant, M. y. Guo, and L. Jones, "Character-Level Language Modeling with Deeper Self-Attention," *arXiv e-prints*, p. arXiv:1808.04444, Aug. 2018.

[182] B. McCann, N. Shirish Keskar, C. Xiong, and R. Socher, "The Natural Language Decathlon: Multitask Learning as Question Answering," *arXiv e-prints*, p. arXiv:1806.08730, Jun. 2018.

[183] N. Shirish Keskar, B. McCann, C. Xiong, and R. Socher, "Unifying Question Answering, Text Classification, and Regression via Span Extraction," *arXiv e-prints*, p. arXiv:1904.09286, Apr. 2019.

[184] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," *arXiv e-prints*, p. arXiv:1511.06349, Nov. 2015.

[185] N. Shirish Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A Conditional Transformer Language Model for Controllable Generation," *arXiv e-prints*, p. arXiv:1909.05858, Sep. 2019.

[186] G. Lample and A. Conneau, "Cross-lingual Language Model Pretraining," *arXiv e-prints*, p. arXiv:1901.07291, Jan. 2019.

[187] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," *arXiv e-prints*, p. arXiv:1910.13461, Oct. 2019.

[188] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crand all, and D. Batra, "Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models," *arXiv e-prints*, p. arXiv:1610.02424, Oct. 2016.

[189] L. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil, "Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models," *arXiv e-prints*, p. arXiv:1701.03185, Jan. 2017.

[190] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-Source Toolkit for Neural Machine Translation," *arXiv e-prints*, p. arXiv:1701.02810, Jan. 2017.

[191] K. Murray and D. Chiang, "Correcting Length Bias in Neural Machine Translation," *arXiv e-prints*, p. arXiv:1808.10006, Aug. 2018.

[192] Y. Yang, L. Huang, and M. Ma, "Breaking the Beam Search Curse: A Study of (Re-)Scoring Methods and Stopping Criteria for Neural Machine Translation," *arXiv e-prints*, p. arXiv:1808.09582, Aug. 2018.

[193] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The Curious Case of Neural Text Degeneration," *arXiv e-prints*, p. arXiv:1904.09751, Apr. 2019.

[194] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical Neural Story Generation," *arXiv e-prints*, p. arXiv:1805.04833, May 2018.

[195] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," vol. 2, 01 2010, pp. 1045–1048.

[196] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://www.aclweb.org/anthology/P02-1040

[197] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "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*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2122–2132. [Online]. Available: https://www.aclweb.org/anthology/D16-1230

[198] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A Diversity-Promoting Objective Function for Neural Conversation Models," *arXiv e-prints*, p. arXiv:1510.03055, Oct. 2015.

[199] M. Naaman, J. Boase, and C.-H. Lai, "Is it really about me? message content in social awareness streams," in *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 189–192. [Online]. Available: https://doi.org/10.1145/1718918.1718953

[200] O. Vinyals and Q. Le, "A Neural Conversational Model," *arXiv e-prints*, p. arXiv:1506.05869, Jun. 2015.

[201] A. Venkatesh, C. Khatri, A. Ram, F. Guo, R. Gabriel, A. Nagar, R. Prasad, M. Cheng, B. Hedayatnia, A. Metallinou, R. Goel, S. Yang, and A. Raju, "On Evaluating and Comparing Open Domain Dialog Systems," *arXiv e-prints*, p. arXiv:1801.03625, Jan. 2018.

[202] J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan, "A persona-based neural conversation model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 994–1003. [Online]. Available: https://www.aclweb.org/anthology/P16-1094

[203] Y. Wu, F. Wei, S. Huang, Y. Wang, Z. Li, and M. Zhou, "Response Generation by Context-aware Prototype Editing," *arXiv e-prints*, p. arXiv:1806.07042, Jun. 2018.

[204] J. Weston, E. Dinan, and A. H. Miller, "Retrieve and Refine: Improved Sequence Generation Models For Dialogue," *arXiv e-prints*, p. arXiv:1808.04776, Aug. 2018.

[205] D. Cai, Y. Wang, V. Bi, Z. Tu, X. Liu, W. Lam, and S. Shi, "Skeleton-to-Response: Dialogue Generation Guided by Retrieval Memory," *arXiv e-prints*, p. arXiv:1809.05296, Sep. 2018.

[206] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing Dialogue Agents: I have a dog, do you have pets too?" *arXiv e-prints*, p. arXiv:1801.07243, Jan. 2018.

[207] I. Kulikov, A. H. Miller, K. Cho, and J. Weston, "Importance of Search and Evaluation Strategies in Neural Dialogue Modeling," *arXiv e-prints*, p. arXiv:1811.00907, Nov. 2018.

[208] E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, J. Williams, J. Pineau, M. Burtsev, and J. Weston, "The Second Conversational Intelligence Challenge (ConvAI2)," *arXiv e-prints*, p. arXiv:1902.00098, Jan. 2019.

[209] P.-E. Mazaré, S. Humeau, M. Raison, and A. Bordes, "Training millions of personalized dialogue agents," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2775–2779. [Online]. Available: https://www.aclweb.org/anthology/D18-1298

[210] B. Hancock, A. Bordes, P.-E. Mazaré, and J. Weston, "Learning from Dialogue after Deployment: Feed Yourself, Chatbot!" *arXiv e-prints*, p. arXiv:1901.05415, Jan. 2019.

[211] A. Madotto, Z. Lin, C.-S. Wu, and P. Fung, "Personalizing dialogue agents via meta-learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5454–5459. [Online]. Available: https://www.aclweb.org/anthology/P19-1542

[212] C. K. Joshi, F. Mi, and B. Faltings, "Personalization in Goal-Oriented Dialog," *arXiv e-prints*, p. arXiv:1706.07503, Jun. 2017.

[213] S. Yavuz, A. Rastogi, G.-L. Chao, and D. Hakkani-Tur, "DeepCopy: Grounded Response Generation with Hierarchical Pointer Networks," *arXiv e-prints*, p. arXiv:1908.10731, Aug. 2019.

[214] Y. Zemlyanskiy and F. Sha, "Aiming to know you better perhaps makes me a more engaging dialogue partner," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 551–561. [Online]. Available: https://www.aclweb.org/anthology/K18-1053

[215] N. Lee, Z. Liu, and P. Fung, "Team yeon-zi at SemEval-2019 task 4: Hyperpartisan news detection by de-noising weakly-labeled data," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 1052–1056. [Online]. Available: https://www.aclweb.org/anthology/S19-2184

[216] Y. Fan, J. C. K. Lam, and V. O. K. Li, "Multi-Region Ensemble Convolutional Neural Network for Facial Expression Recognition," *arXiv e-prints*, p. arXiv:1807.10575, Jul. 2018.

[217] ——, "Unsupervised domain adaptation with generative adversarial networks for facial emotion recognition," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4460–4464.

[218] Y. Fan, J. C. K. Lam, and V. O. K. Li, "Video-based emotion recognition using deeply-supervised neural networks," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ser. ICMI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 584–588. [Online]. Available: https://doi.org/10.1145/3242969.3264978

[219] P. Xu, A. Madotto, C.-S. Wu, J. H. Park, and P. Fung, "Emo2Vec: Learning generalized emotion representation by multi-task training," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 292–298. [Online]. Available: https://www.aclweb.org/anthology/W18-6243

[220] G. I. Winata, O. Kampman, Y. Yang, A. Dey, and P. Fung, "Nora the empathetic psychologist," in *Proc. Interspeech 2017*, 2017, pp. 3437–3438.

[221] G. I. Winata, A. Madotto, Z. Lin, J. Shin, Y. Xu, P. Xu, and P. Fung, "CAiRE_HKUST at SemEval-2019 task 3: Hierarchical attention for dialogue emotion classification," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 142–147. [Online]. Available: https://www.aclweb.org/anthology/S19-2021

[222] A. Chatterjee, U. Gupta, M. K. Chinnakotla, R. Srikanth, M. Galley, and P. Agrawal, "Understanding emotions in text using deep learning and big data," *Computers in Human Behavior*, vol. 93, pp. 309 – 317, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747563218306150

[223] X. Zhou and W. Y. Wang, "MojiTalk: Generating Emotional Responses at Scale," *arXiv e-prints*, p. arXiv:1711.04090, Nov. 2017.

[224] N. Asghar, P. Poupart, J. Hoey, X. Jiang, and L. Mou, "Affective Neural Response Generation," *arXiv e-prints*, p. arXiv:1709.03968, Sep. 2017.

[225] N. Lubis, S. Sakti, K. Yoshino, and S. Nakamura, "Eliciting positive emotion through affect-sensitive dialogue response generation: A neural network approach," in *AAAI*, 2018.

[226] N. Lubis, S. Sakti, K. Yoshino, and S. Nakamura, "Positive emotion elicitation in chat-based dialogue systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 866–877, 2019.

[227] Z. Lin, A. Madotto, J. Shin, P. Xu, and P. Fung, "MoEL: Mixture of empathetic listeners," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 121–132. [Online]. Available: https://www.aclweb.org/anthology/D19-1012

[228] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," *arXiv e-prints*, p. arXiv:1701.06538, Jan. 2017.

[229] J. Shin, P. Xu, A. Madotto, and P. Fung, "HappyBot: Generating Empathetic Dialogue Responses by Improving User Experience Look-ahead," *arXiv e-prints*, p. arXiv:1906.08487, Jun. 2019.

[230] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[231] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books," *arXiv e-prints*, p. arXiv:1506.06724, Jun. 2015.

[232] A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston, "ParlAI: A Dialog Research Software Platform," *arXiv e-prints*, p. arXiv:1705.06476, May 2017.

[233] E. M. Smith, M. Williamson, K. Shuster, J. Weston, and Y.-L. Boureau, "Can You Put it All Together: Evaluating Conversational Agents' Ability to Blend Skills," *arXiv e-prints*, p. arXiv:2004.08449, Apr. 2020.

[234] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston, "Wizard of Wikipedia: Knowledge-Powered Conversational agents," *arXiv e-prints*, p. arXiv:1811.01241, Nov. 2018.

[235] S. Rothe, S. Narayan, and A. Severyn, "Leveraging Pre-trained Checkpoints for Sequence Generation Tasks," *arXiv e-prints*, p. arXiv:1907.12461, Jul. 2019.

[236] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning Word Vectors for 157 Languages," *arXiv e-prints*, p. arXiv:1802.06893, Feb. 2018.

[237] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1615–1625. [Online]. Available: https://www.aclweb.org/anthology/D17-1169

[238] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the Limits of Weakly Supervised Pretraining," *arXiv e-prints*, p. arXiv:1805.00932, May 2018.