Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας σηματων, ελεγχου και ρομποτικης

A Multi-Task BERT Model for Schema-Guided Dialogue State Tracking

Διπλωματικη Εργασια

του

ΕΛΕΥΘΕΡΙΟΥ ΚΑΠΕΛΩΝΗ

Επιβλέπων: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής

Αθήνα, Ιούνιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας σηματων, ελεγχου και ρομποτικης

A Multi-Task BERT Model for Schema-Guided Dialogue State Tracking

Διπλωματικη Εργασια

του

ΕΛΕΥΘΕΡΙΟΥ ΚΑΠΕΛΩΝΗ

Επιβλέπων: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Ιουνίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αλέξανδρος Ποταμιάνος	Κωνσταντίνος Τζαφέστας	Στέφανος Κόλλιας
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Καθηγητής

Αθήνα, Ιούνιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας σηματων, ελεγχου και ρομποτικης

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Ελευθέριος Καπελώνης

14 Ιουνίου 2022

Περίληψη

Τα διαλογικά συστήματα συχνά χρησιμοποιούν το υποσύστημα Dialogue State Tracking (DST) για να ολοκληρώσουν επιτυχώς τη συζήτηση. Το DST έχει σκοπό να παρακολουθεί το στόχο του χρήστη κατά τη διάρκεια ενός διαλόγου και είναι ένα αρκετά απαιτητικό πρόβλημα σε multi-domain σενάρια. Το schema-guided DST είναι μια νέα προσέγγιση, στην οποία το σχήμα, δηλαδή μία λίστα από τα υποστηριζόμενα πεδία και προθέσεις μαζί με αναπαραστάσεις σε φυσική γλώσσα, παρέχεται για κάθε υπηρεσία του διαλόγου. Πρόσφατες state-of-the-art υλοποιήσεις του DST βασίζονται στα σχήματα ποικίλλων υπηρεσιών για να βελτιώσουν την ευρωστία και να πραγματοποιούν zero-shot γενίκευση σε νέα domains, όμως τέτοιες μέθοδοι χρειάζονται συνήθως πολλαπλά transformer μοντέλα μεγάλης κλίμακας και μεγάλες ακολουθίες εισόδου για να έχουν καλή απόδοση.

Σε αυτή τη διπλωματική εργασία, πρώτα εισάγουμε τα βασικά της μηχανικής μάθησης, της βαθιάς μάθησης, της επεξεργασίας φυσικής γλώσσας και των διαλογικών συστημάτων εστιάζοντας στο DST. Στη συνέχεια, προτείνουμε ένα μοντέλο πολλαπλών εργασιών βασισμένο στο BERT για να λύσουμε ταυτόχρονα τα τρία DST προβλήματα: πρόβλεψη πρόθεσης, πρόβλεψη ζητούμενων πεδίων και ανάθεση τιμών πεδίων. Επιπλέον, προτείνουμε μια αποδοτική και φειδωλή κωδικοποίηση του ιστορικού του διαλόγου και του σχήματος των υπηρεσιών που δείχνουμε ότι βελτιώνει περαιτέρω την απόδοση.

Κωδικοποιούμε μόνο τους δύο τελευταίους γύρους του διαλόγου, μια μικρή αναπαράσταση για το σχήμα και την προηγούμενη κατάσταση του διαλόγου. Ο προηγούμενος γύρος του συστήματος αναπαρίσταται μέσω των υποκείμενων διαλογικών ενεργειών συστήματος που βελτιώνει σημαντικά την ακρίβεια του DST. Για το πρόβλημα ανάθεσης τιμών πεδίων προσθέτουμε επίσης μηχανισμούς μεταφοράς των πεδίων που ψάχνουν προηγούμενους γύρους και καταστάσεις διαλόγου για να ανακτήσουν την τιμή όταν αυτό απαιτείται. Ένας αριθμός από κεφαλές κατηγοριοποίησης που παίρνουν ως είσοδο διάφορα μέρη της κωδικοποιημένης από το BERT ακολουθίας εκπαιδεύονται ταυτόχρονα για την επίλυση των τριών προβλημάτων.

Η αξιολόγηση στο σύνολο δεδομένων SGD δείχνει ότι η απόδοση της προσέγγισής μας ξεπερνάει κατά πολύ αυτή του συστήματος αναφοράς SGP-DST και είναι κοντά στο state-of-the-art, ενώ είναι πολύ λιγότερο απαιτητική σε υπολογιστικούς πόρους. Πραγματοποιούμε αναλυτικά ablation studies που εξετάζουν τους καθοριστικούς παράγοντες για την επιτυχία του μοντέλου μας.

Λέξεις Κλειδιά

μηχανική μάθηση, βαθιά μάθηση, επεξεργασία φυσικής γλώσσας, bert, διαλογικά συστήματα, dialogue state tracking, μάθηση πολλαπλών εργασιών

Abstract

Dialogue systems often employ a Dialogue State Tracking (DST) component to successfully complete conversations. DST aims to track the user goal over the course of a dialogue and it is a particularly challenging task in multi-domain scenarios. Schema-guided DST is a new approach, where the schema, i.e. a list of the supported slots and intents along with natural language descriptions, is provided for each dialogue service. Recent state-of-the-art DST implementations rely on schemata of diverse services to improve model robustness and handle zero-shot generalization to new domains, however such methods typically require multiple large scale transformer models and long input sequences to perform well.

In this diploma thesis, we first introduce the basics of machine learning, deep learning, natural language processing and dialogue systems focusing on DST. We then propose a single multi-task BERT-based model that jointly solves the three DST tasks of intent prediction, requested slot prediction and slot filling. Moreover, we propose an efficient and parsimonious encoding of the dialogue history and service schemata that is shown to further improve performance.

We only encode the last two utterances, a compact schema representation and the previously predicted dialogue state. The preceding system utterance is represented as its underlying system actions which significantly benefits accuracy. For the slot filling task we additionally incorporate slot carryover mechanisms that search previous dialogue utterances and states to retrieve values when necessary. A number of classification heads which take as input various parts of the BERT-encoded sequence are jointly trained to perform the tasks.

Evaluation on the SGD dataset shows that our approach outperforms the baseline SGP-DST by a large margin and performs well compared to the state-of-the-art, while being significantly more computationally efficient. Extensive ablation studies are performed to examine the contributing factors to the success of our model.

Keywords

machine learning, deep learning, natural language processing, bert, dialogue systems, dialogue state tracking, multi-task learning

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Αλέξανδρο Ποταμιάνο για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο SLP-NTUA.

Επίσης ευχαριστώ ιδιαίτερα τους Υποψήφιους Διδάκτορες Ευθύμη Γεωργίου και Γιώργο Παρασκευόπουλο για την καθοδήγησή τους και την εξαιρετική συνεργασία που είχαμε.

Ευχαριστώ επιπλέον τους καθηγητές κ. Τζαφέστα και κ. Κόλλια που ανταποκρίθηκαν με προθυμία να συμμετάσχουν στην τριμελή επιτροπή.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την καθοδήγηση και την στήριξη που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Ιούνιος 2022

Ελευθέριος Καπελώνης

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 0

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

0.1.1 Κίνητρο

Μηχανές που μπορούν να συζητούν με φυσικό τρόπο με ανθρώπους έχουν αποτελέσει το αντικείμενο ουτοπικών έργων επιστημονικής φαντασίας. Παρόλα αυτά, λόγω των τελευταίων εξελίξεων στην τεχνητή νοημοσύνη και πιο συγκεκριμένα στη βαθιά μάθηση, ευφυή διαλογικά συστήματα αρχίζουν να γίνονται πραγματικότητα πετυχαίνοντας κάτι που προηγουμένως ήταν μόνο στη φαντασία μας. Δημοφιλείς ψηφιακοί βοηθοί όπως η Siri, η Cortana, το Google Assistant, η Alexa κλπ είναι ικανοί να εκτελέσουν εργασίες ή/και να ψυχαγωγήσουν τους χρήστες μέσω συζητήσεων.

Μια αξιοσημείωτη κατηγορία πρακτόρων συζήτησης είναι τα διαλογικά συστήματα προσανατολισμένα σε συγκεκριμένο σκοπό (task-oriented). Τέτοια συστήματα στοχεύουν να βοηθήσουν τους χρήστες να ολοκληρώσουν καθημερινές δραστηριότητες όπως να κάνουν κράτηση σε εστιατόριο, να κλείσουν εισιτήρια κλπ. Ένα κρίσιμο υποσύστημα σε ένα task-oriented διαλογικό σύστημα είναι το Dialogue State Tracking (DST) που παρακολουθεί και καταγράφει το στόχο του χρήστη κατά τη διάρκεια πολλαπλών γύρων του διαλόγου. Με βάση τα λόγια του τελευταίου γύρου του διαλόγου (utterance) και το ιστορικό του διαλόγου, το DST προβλέπει την κατάσταση του διάλογου (dialogue state) που αποτυπώνει το στόχο του χρήστη. Η κατάσταση του διαλόγου που προβλέφθηκε έπειτα χρησιμοποιείται από άλλα υποσυστήματα για να ανακτηθούν στοιχεία από μια βάση δεδομένων, να εκτελεστούν οι ενέργειες που έχουν ζητηθεί από το χρήστη και να απαντήσουν κατάλληλα [15]. Πιο πρόσφατα, τα συστήματα που έχουν προταθεί είναι multi-domain· μπορούν να χειριστούν πολλαπλά διαλογικά domains στον ίδιο διάλογο.

Η ύπαρξη ολοένα και αυξανόμενων διαφορετικών υπηρεσιών που προσφέρονται από τα εμπορικά task-oriented συστήματα οδήγησε την ανάπτυξη του καθοδηγούμενου από το σχήμα προτύπου (schema-guided paradigm) [1]. Αυτό το πρότυπο έχει σκοπό να επιτρέψει στα μοντέλα DST να δουλεύουν σε νέες, άγνωστες υπηρεσίες βασιζόμενα στη γνώση που έχουν αποκτήσει από τις υπηρεσίες που είδαν κατά την εκπαίδευση. Η περιγραφή σε φυσική γλώσσα του σχήματος των υπηρεσιών καθοδηγεί το μοντέλο σε τέτοια καινούρια σενάρια.

Παράλληλα, πρόσφατα προεκπαιδεύμενα transformer γλωσσικά μοντέλα όπως το BERT [17] και το T5 [11] έχουν δείξει εξαιρετική απόδοση σε πολλά προβλήματα κατανόησης φυσικής γλώσσας. Λόγω της φάσης προεκπαίδευσης σε μεγάλα σύνολα δεδομένων, μαθαίνουν να κατανοούν τη γλώσσα μέσω παραδειγμάτων και μετέπειτα μπορούν να προσαρμοστούν σε συγκεκριμένα προβλήματα όπως το DST μέσω μιας διαδικασίας γνωστής ως fine-tuning.

Τέτοια μοντέλα είναι συνεπώς κατάλληλα για διαλογικά συστήματα εξαιτίας της ικανότητάς τους να γενικεύουν.

0.1.2 Συνεισφορές

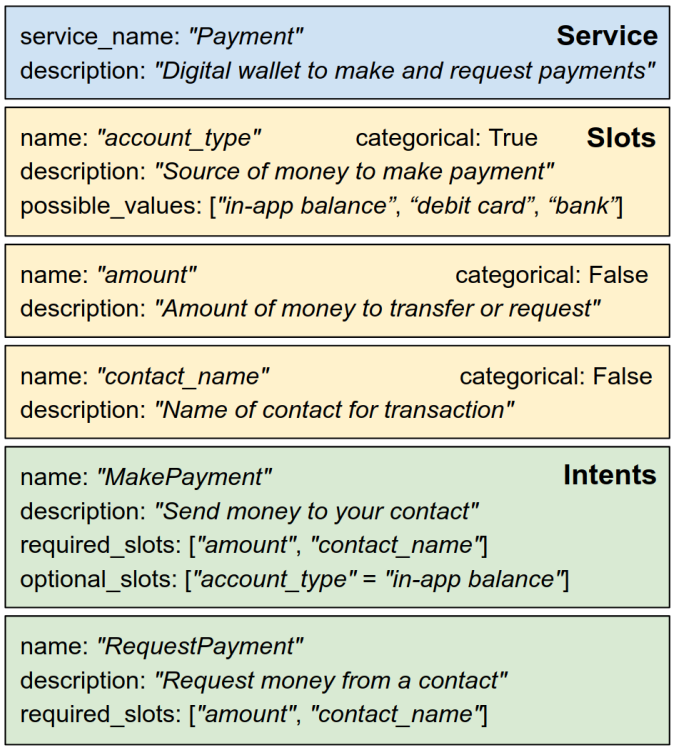
Σε αυτή τη διπλωματική εργασία, προτείνουμε ένα μοντέλο πολλαπλών εργασιών (multi-task) βασισμένο στο BERT που λύνει ταυτόχρονα τα τρία DST προβλήματα: πρόβλεψη πρόθεσης (intent prediction), πρόβλεψη ζητούμενων πεδίων (requested slot prediction) και ανάθεση τιμών πεδίων (slot filling). Επιπλέον, κατασκευάζουμε μια αποδοτική και φειδωλή συνοπτική αναπαράσταση για το διάλογο και το σχήμα που δείχνουμε ότι βελτιώνει σημαντικά την απόδοση ενώ είναι λιγότερο υπολογιστά απαιτητική. Το προτεινόμενο μοντέλο μας ξεπερνά κατά πολύ σε απόδοση το σύστημα αναφοράς (baseline) και η απόδοσή του είναι κοντά στο state-of-the-art. Αναλυτικά ablation studies αποκαλύπτουν την επίδραση κάθε στρατηγικής του μοντέλου μας στο πρόβλημα ανάθεσης τιμών πεδίων.

0.2 Το Schema-Guided Dialogue Σύνολο Δεδομένων

Σε αυτή την εργασία χρησιμοποιούμε το Schema-Guided Dialogue (SGD) Σύνολο Δεδομένων [1], ένα multi-domain task-oriented σύνολο δεδομένων μεγάλης κλίμακας που ακολουθεί το schema-guided πρότυπο. Το SGD περιέχει 21,106 διαλόγους σε 20 domains και 45 υπηρεσίες υπερβαίνοντας σε κλίμακα άλλα σύνολα δεδομένων. Συμπεριλαμβάνει επισημειώσεις που διευκολύνουν πολλαπλά task-oriented διαλογικά προβλήματα όπως η κατανόηση φυσικής γλώσσας (Natural Language Understanding - NLU), το DST και την παραγωγή απάντησης. Το σύνολο δεδομένων περιέχει περιγραφές σε φυσική γλώσσα των διαφόρων στοιχείων του σχήματος· ένα παράδειγμα σχήματος φαίνεται στο Σχήμα 1. Για να αξιολογηθεί η zero-shot [1] ικανότητα γενίκευσης σε καινούριες υπηρεσίες και domains, στο επίσημο σύνολο δοκιμών (test set) το 77% των διαλογικών γύρων περιέχουν τουλάχιστον μια υπηρεσία που δεν είναι παρούσα στο σύνολο εκπαίδευσης. Με αυτόν τον τρόπο, δίνεται το κίνητρο για την σχεδίαση ενός ενοποιημένου μοντέλου για όλες τις υπηρεσίες και το σχήμα δίνεται ως είσοδος.

Στο Schema-Guided DST track του 8ου Dialogue System Technology Challenge, οι συμμετέχοντες ανέπτυξαν zero-shot DST καθοδηγούμενα από το σχήμα μοντέλα βασισμένα στο σύνολο δεδομένων [18]. Τα DST προβλήματα αποτελούν την πρόβλεψη της ενεργής πρόθεσης του χρήστη (**intent prediction**), των πεδίων που ζητούνται από το χρήστη (**requested slot prediction**) και των τιμών που δίνονται για τα πεδία από το χρήστη έως τον συγκεκριμένο γύρο (**slot filling**). Επομένως, θεωρούμε ως κατάσταση διαλόγου (dialogue state) την ενεργή πρόθεση, τα ζητούμενα πεδία και τα ζευγάρια πεδίων-τιμών για ένα γύρο παρόλο που τα δύο πρώτα προβλήματα πιο συχνά θεωρούνται NLU πρόβλημα. Σε multi-domain διαλόγους, μια ξεχωριστή κατάσταση διαλόγου υπολογίζεται για κάθε υπηρεσία που υπάρχει στο διάλογο.

[1] με τον όρο zero-shot learning αναφερόμαστε στο πρόβλημα στο οποίο κατά τη διάρκεια αξιολόγησης κάποιου μοντέλου μηχανικής μάθησης υπάρχουν δεδομένα που δεν έχουν παρατηρηθεί στην εκπαίδευση



Σχήμα 1: *Ένα παράδειγμα σχήματος για την υπηρεσία Payment. Το σχήμα περίεχει μια λίστα από πεδία και προθέσεις. Τα πεδία είναι είτε κατηγορικά είτε μη-κατηγορικά και για τα κατηγορικά δίνεται λίστα με τις πιθανές τιμές. Επιπλέον, κάθε πρόθεση απαριθμεί τα απαιτούμενα και τα προαιρετικά πεδία που ο χρήστης πρέπει να δώσει για να αλληλεπιδράσει με τη συγκεκριμένη πρόθεση. Πηγή: [1]*

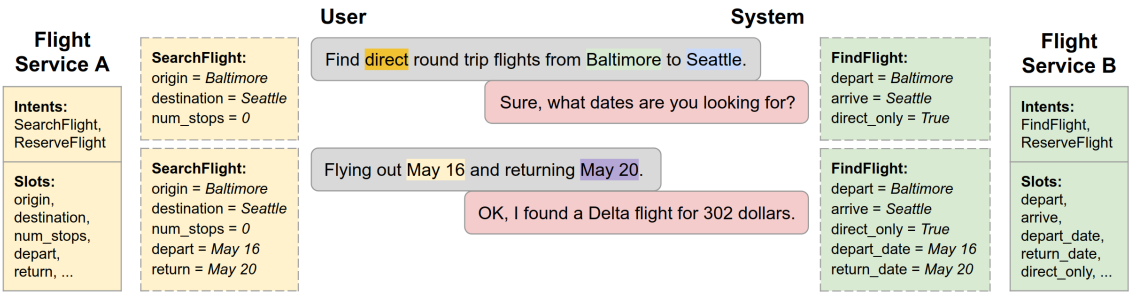
Όπως μπορούμε να δούμε στο Σχήμα 1, υπάρχει μια σχέση μεταξύ των υποστηριζόμενων προθέσεων και πεδίων. Κάθε πρόθεση απαριθμεί απαιτούμενα και προαιρετικά πεδία. Μπορούμε να κατηγοριοποιήσουμε ένα πεδίο ως **informable** ή **non-informable** ανάλογα με το αν ο χρήστης επιτρέπεται να δώσει τιμή για αυτό. Στις επόμενες ενότητες υποθέτουμε ότι ένα πεδίο είναι informable αν είναι είτε απαιτούμενο είτε προαιρετικό σε τουλάχιστον μία πρόθεση. Για το πρόβλημα ανάθεσης τιμών πεδίων θεωρούμε μόνο τα informable πεδία ως υποψήφια ενώ οποιοδήποτε πεδίο (informable ή όχι) μπορεί να ζητηθεί.

Οι μετρικές αξιολόγησης για το πρόβλημα του DST στο SGD σύνολο δεδομένων είναι οι ακόλουθες σύμφωνα με [1]:

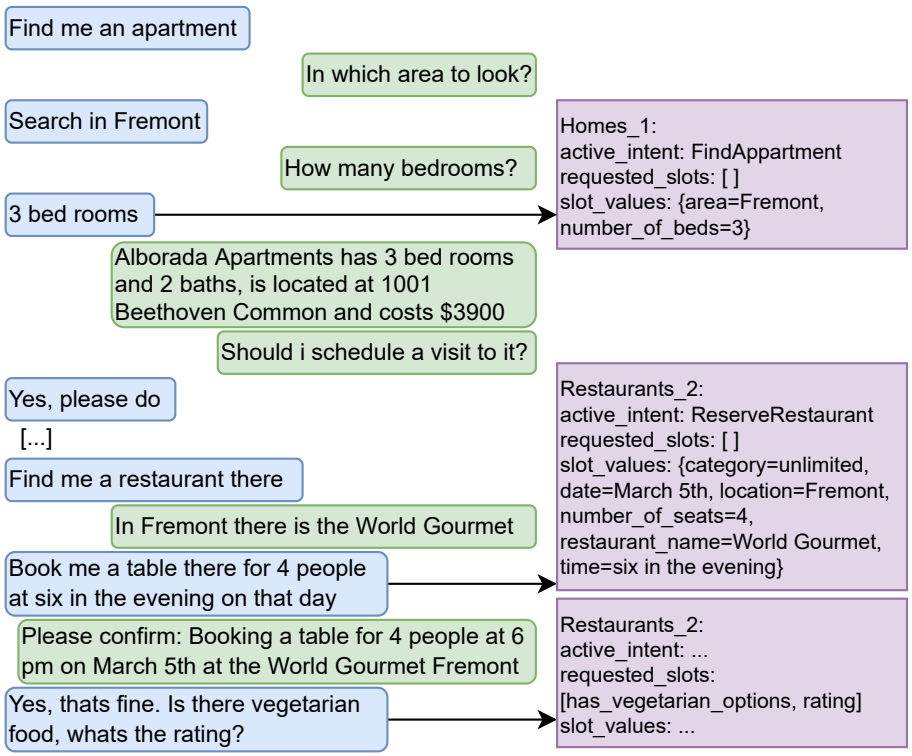
- **Active intent accuracy:** Το ποσοστό των γύρων που μιλάει ο χρήστης για τους οποίους έγινε σωστή πρόβλεψη για την ενεργή πρόθεση.
- **Requested slot F1:** Το macro-averaged F1 score για τα ζητούμενα πεδία στους γύρους του χρήστη. Γύροι χωρίς ζητούμενα πεδία στην πραγματική (ground-truth) και στην κατάσταση διαλόγου που προβλέφθηκε παραλείπονται.
- **Average Goal Accuracy:** Η μέση ακρίβεια σωστής πρόβλεψης των πεδίων στους σχετικούς γύρους. Πεδία για τα οποία δεν έχει ανατεθεί τιμή στη ground truth κατάσταση του διαλόγου παραλείπονται. Για τα μη-κατηγορικά πεδία, ένα fuzzy score αντιστοίχισης (fuzzy matching score) χρησιμοποιείται για να ανταμείψει εν μέρει σωστές απα-

ντήσεις.

- **Joint Goal Accuracy:** Η μέση ακρίβεια σωστής πρόβλεψης όλων των πεδίων σε έναν συγκεκριμένο γύρο. Για τα μη-κατηγορικά πεδία, ένα fuzzy matching score χρησιμοποιείται για να ανταμείψει εν μέρει σωστές απαντήσεις.



Σχήμα 2: *Ένας διάλογος από το Flight domain. Οι υπηρεσίες A και B μπορούν να χρησιμοποιηθούν ως διεπαφή στο domain. Παρόλο που προσφέρουν την ίδια λειτουργικότητα, υπάρχουν μικρές διαφοροποιήσεις στο σχήμα τους επειδή ενδεχομένως προέρχονται από διαφορετικούς σχεδιαστές API. Πηγή: [1]*



Σχήμα 3: *Απόσπασμα διαλόγου με επισημειώσεις για κάποιους γύρους*

0.3 Σχετική βιβλιογραφία

Το SGD-baseline [1] πραγματοποιεί fine-tuning στο μοντέλο BERT με βάση τους δύο τελευταίους γύρους και συνενώνει τις κωδικοποιημένες μονάδες (tokens) με τα embeddings

του σχήματος - τα κωδικοποιημένα με το BERT στοιχεία του σχήματος. Τα embeddings του σχήματος είναι υπολογισμένα πριν την εκπαίδευση. Στη συνέχεια, κεφαλές κατηγοριοποίησης (classification heads) παίρνουν ως είσοδο τους κωδικοποιημένους γύρους του διαλόγου και διαφορετικά embeddings του σχήματος και παράγουν τις πιθανότητες των κλάσεων για κάθε πρόβλημα.

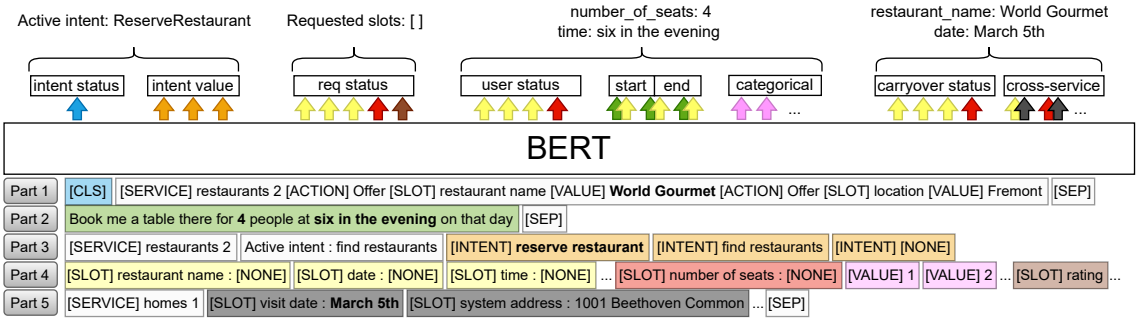
Κάποια από τα μοντέλα που έχουν προταθεί πραγματοποιούν fine-tuning σε έναν αριθμό από ξεχωριστά προεκπαιδευμένα μοντέλα για κάθε υποπρόβλημα με είσοδο τους δύο τελευταίους γύρους και την περιγραφή κάθε στοιχείου του σχήματος (πεδίου ή πρόθεσης) [19, 20]. Επιπροσθέτως, για να αντιμετωπίσουν το πρόβλημα της μακροπρόθεσμης εξάρτησης μεταξύ πεδίων (βλέπε Ενότητα 4.2.4) χρησιμοποιούν μηχανισμούς μεταφοράς πεδίων. Το SGP-DST [19] και το σύστημα SPPD [20] χρησιμοποιούν έναν αριθμό από προεκπαιδευμένα μοντέλα BERT και υιοθετούν την προσέγγιση πολλαπλών περασμάτων (multi-pass) και έτσι χρειάζονται πολλαπλά περάσματα από το BERT για κάθε γύρο του διαλόγου.

Οι state-of-the-art μέθοδοι [21, 22] δεν βασίζονται σε μηχανισμούς μεταφοράς πεδίων· εκπαιδεύονται με ολόκληρο το ιστορικό του διαλόγου. Τα πλεονεκτήματα τέτοιων προσεγγίσεων είναι ότι αποφεύγουν τη συσσώρευση σφαλμάτων και η απόδοση είναι καλύτερη από μεθόδους με μηχανισμούς μεταφοράς πεδίων. Το paDST [21] πραγματοποιεί fine-tuning σε έναν αριθμό από προεκπαιδευμένα μοντέλα με ολόκληρο το διάλογο και έχει παρατηρηθεί ότι μεγάλος αριθμός από χειροκίνητα κατασκευασμένα χαρακτηριστικά και επαύξηση δεδομένων μπορούν να βελτιώσουν πολύ την απόδοση στα κατηγορικά πεδία. Από την άλλη, το D3ST [22] πραγματοποιεί fine-tuning σε ένα μόνο T5 μοντέλο για όλα τα υποπροβλήματα με τη προσέγγιση ενός περάσματος (single-pass) δηλαδή όλες οι περιγραφές για το σχήμα ενώνονται και τροφοδοτούνται στο μοντέλο μαζί με το διάλογο.

0.4 Προτεινόμενο μοντέλο

Σε αυτήν την ενότητα, προτείνουμε ένα μοντέλο πολλαπλών εργασιών βασισμένο στο BERT που πραγματοποιεί ταυτόχρονα την πρόβλεψη πρόθεσης, την πρόβλεψη ζητούμενων πεδίων και την ανάθεση τιμών πεδίων. Στο προτεινόμενο μοντέλο, υιοθετούμε μηχανισμούς μεταφοράς πεδίων και κωδικοποιούμε μόνο τον αμέσως προηγούμενο γύρο του συστήματος και τον τρέχοντα γύρο του χρήστη όπως στα συστήματα αναφοράς (βλέπε Ενότητα 5.3 και 5.4). Επιπλέον, ο προηγούμενος γύρος του συστήματος αναπαρίσταται συνοπτικά αξιοποιώντας τις υποκείμενες διαλογικές ενέργειες του συστήματος (system actions). Για να επιτευχθεί μια πιο αποδοτική και φειδωλή αναπαράσταση της εισόδου, κωδικοποιούμε όλα τα στοιχεία του σχήματος μαζί χρησιμοποιώντας μόνο τα ονόματά τους και συμπεριλαμβάνουμε επιλεκτικά προηγούμενες καταστάσεις διαλόγου. Αυτή είναι η πιο σημαντική διαφορά σε σχέση με τα συστήματα αναφοράς. Το προτεινόμενο μοντέλο μας ξεπερνάει κατά πολύ σε απόδοση το σύστημα αναφοράς SGP-DST και επιτυγχάνει απόδοση κοντά στο state-of-the-art. Αναλυτικά ablation studies αποκαλύπτουν την επίδραση κάθε στρατηγικής του μοντέλου μας στο πρόβλημα της ανάθεσης τιμών πεδίων.

Η αρχιτεκτονική του μοντέλου πολλαπλών εργασιών φαίνεται στο Σχήμα 4. Ο γύρος του χρήστη, ο προηγούμενος γύρος του συστήματος, τα σχήματα και προηγούμενες DST πληροφορίες (βλέπε Μέρη 1 έως 5) κωδικοποιούνται μέσω BERT. Διαφορετικά κομμάτια



Σχήμα 4: *Οι είσοδοι για τις κεφαλές πρόβλεψης πρόθεσης, πρόβλεψης ζητούμενων πεδίων, ανάθεσης τιμών πεδίων και μεταφοράς πεδίων φαίνονται για το προτεινόμενο μοντέλο πολλαπλών εργασιών BERT (πάνω μέρος), μαζί με ένα παράδειγμα κωδικοποίησης του γύρου και του ιστορικού διαλόγου που είναι είσοδος στο μοντέλο BERT (κάτω μέρος). Προσέξτε τα χρώματα της εισόδου στις κεφαλές κατηγοριοποίησης (πάνω μέρος) που αντιστοιχούν στα διάφορα μέρη της ακολουθίας εισόδου (κάτω μέρος). Για αυτό το παράδειγμα, η υπηρεσία στους γύρους του συστήματος και του χρήστη είναι η Restaurants_2. Η προηγούμενη πρόθεση FindRestaurants αλλάζει στην ReserveRestaurant. Κανένα πεδίο δεν ζητείται από το χρήστη. Στον προηγούμενο γύρο του συστήματος, το σύστημα προσφέρει την τιμή "World Gourmet" για το πεδίο restaurant_name που ο χρήστης αποδέχεται (μεταφορά πεδίων in_sys_uttr). Ο χρήστης δίνει τις τιμές "six in the evening" και '4' για το μη-κατηγορικό πεδίο time και το κατηγορικό πεδίο number_of_seats. Η τιμή του date δεν λέγεται άμεσα αλλά υπονοείται ότι έχει αναφερθεί προηγουμένως (μεταφορά πεδίου in_cross_service_hist από μια προηγούμενη υπηρεσία (Homes_1)). Μέρος της εισόδου παραλείπεται.*

της κωδικοποιημένες ακολουθίας (βλέπε αντίστοιχιση με χρώματα στο σχήμα) δίνονται ως είσοδος σε εννιά κεφαλές κατηγοριοποίησης που λύνουν τα προβλήματα της πρόβλεψης πρόθεσης (2 κεφαλές), της πρόβλεψης ζητούμενων πεδίων, της ανάθεσης τιμών πεδίων (4 κεφαλές) και μεταφοράς πεδίων (2 κεφαλές).

0.4.1 Συμβολισμοί

Έστω n μία υπηρεσία του διαλόγου, $I(n)$ το σύνολο των προθέσεων στην υπηρεσία (που συμπεριλαμβάνει την ειδικη κενή πρόθεση `[NONE]`) και $S(n)$ το σύνολο των πεδίων στην υπηρεσία. Τα πεδία χωρίζονται σε κατηγορικά και μη-κατηγορικά. Έστω $S_{cat}(n) \subseteq S(n)$ το σύνολο των κατηγορικών πεδίων και $S_{noncat}(n) \subseteq S(n)$ το σύνολο των μη-κατηγορικών πεδίων. Για κάθε κατηγορικό πεδίο, ένα σύνολο πιθανών τιμών $V(s), s \in S_{cat}(n)$ είναι διαθέσιμο. Επίσης, κάθε πεδίο μπορεί να είναι informable ή όχι ανάλογα με το αν ο χρήστης επιτρέπεται να δώσει τιμή για αυτό. Το $S_{inf}(n) \subseteq S(n)$ υποδηλώνει τα informable πεδία της υπηρεσίας.

Υποθέτουμε ότι κατά το γύρο του χρήστη t σε ένα διάλογο με N υπηρεσίες θέλουμε να προβλέψουμε την κατάσταση του διαλόγου για την υπηρεσία n. Ουσιαστικά θέλουμε να προβλέψουμε την ενεργή πρόθεση $int(n)$ (intent prediction), τα ζητούμενα πεδία $req(n) \subseteq S(n)$ (requested slot prediction) και τις τιμές για τα πεδία που δίνονται από το χρήστη $usrSlotValue(s), s \in S_{inf}(n)$ (slot filling).

Για κάθε υπηρεσία $n', 1 \leq n' \leq N$, το $prevInt(n')$ υποδηλώνει την προηγούμενη ενεργή πρόθεση. Επιπλέον, για κάθε πεδίο $s \in S(n')$, το $prevUsrSlotValue(s)$ υποδηλώνει την τελευταία τιμή που δόθηκε από τον χρήστη για το s. Ακόμα, χρησιμοποιούμε το

$prevSysSlotValue(s)$ και το $sysUttrSlotValue(s)$ για να υποδηλώσουμε την τελευταία τιμή παρούσα σε μια διαλογική ενέργεια συστήματος, πριν το γύρο $t-1$ και στον γύρο (του συστήματος) $t-1$ αντίστοιχα. Για τα $prevSysSlotValue(s)$ και $sysUttrSlotValue(s)$ χρησιμοποιούμε μόνο διαλογικές ενέργειες του συστήματος που περιέχουν το πεδίο s και ακριβώς μια τιμή για το πεδίο. Σε περιπτώσεις που η πρόθεση ή η τιμή του πεδίου είναι άδεια χρησιμοποιούμε την τιμή `[NONE]`.

Χρησιμοποιούμε το S_{prev} για να υποδηλώσουμε το σύνολο των πεδίων $s \in S(n'), n' \neq n$ για τα οποία είτε το $prevUsrSlotValue(s)$ είτε το $prevSysSlotValue(s)$ δεν είναι `[NONE]` και το $prevSlotValue(s)$ για να υποδηλώσουμε την τελευταία τους τιμή. Αν το $prevUsrSlotValue(s)$ δεν είναι `[NONE]` τότε χρησιμοποιούμε αυτήν την τιμή αλλιώς χρησιμοποιούμε το $prevSysSlotValue(s)$.

Για κάθε πεδίο s χρησιμοποιούμε επιπρόσθετα δυαδικά χαρακτηριστικά $x_{bin}(s)$. Τα δυαδικά χαρακτηριστικά είναι τα ακόλουθα: 1) αν η υπηρεσία είναι καινούρια στο διάλογο 2) αν η υπηρεσία εναλλάσσεται (δεν ήταν παρούσα στην προηγούμενη κατάσταση του διαλόγου) 3) αν ακριβώς μια τιμή για το πεδίο υπάρχει στο γύρο διαλόγου συστήματος 4) αν ακριβώς μια τιμή για το πεδίο υπάρχει σε προηγούμενους γύρους συστήματος 5) αν το πεδίο είναι υποχρεωτικό σε τουλάχιστον μία πρόθεση 6) αν το πεδίο είναι προαιρετικό σε όλες τις προθέσεις. Παρόμοια χαρακτηριστικά έχουν χρησιμοποιηθεί από [19].

0.4.2 Αναπαράσταση εισόδου

Ένα παραδείγμα εισόδου φαίνεται στο Σχήμα 4. Στο Μέρος 1 κωδικοποιούμε τον αμέσως προηγούμενο γύρο του συστήματος ως μια λίστα από ενέργειες. Στο Μέρος 2 κωδικοποιούμε τον τρέχοντα γύρο του χρήστη. Στο Μέρος 3, η ενεργή υπηρεσία n, την προηγούμενη ενεργή πρόθεση $prevInt(n)$ και όλες τις υποψήφιες πρόθεσεις που ανήκουν στην υπηρεσία n απαριθμούνται. Το Μέρος 4 περιέχει τη λίστα με όλα τα πεδία $s \in S(n)$. Αν $s \in S_{inf}(n)$ προσθέτουμε το $prevUsrSlotValue(s)$ και αν $s \in S_{cat}(n) \cap S_{inf}(n)$ προσθέτουμε επιπλέον όλες τις τιμές στο $V(s)$. Το Μέρος 5 περιέχει όλες τις άλλες υπηρεσίες που έχουν εμφανιστεί νωρίτερα στο διάλογο. Για κάθε υπηρεσία απαριθμούμε τα ζευγάρια πεδίων-τιμών από προηγούμενες καταστάσεις διαλόγου ή διαλογικές ενέργειες συστήματος, $s \in S_{prev}$ και τις τιμές τους $prevSlotValue(s)$. Προσθέτουμε τη λέξη "system" πριν από πεδία που έχουν δοθεί από το σύστημα για να τα διαφοροποιήσουμε από πεδία που έχουν δοθεί από το χρήστη (δηλαδή που υπάρχουν σε προηγούμενες καταστάσεις του διαλόγου).

Για το σχήμα χρησιμοποιούμε μόνο τα ονόματα για τα πεδία και τις προθέσεις αντί για τις πλήρεις περιγραφές τους σε φυσική γλώσσα που χρησιμοποιούνται από άλλες εργασίες. Ένας αριθμός από προσαρμοσμένα (custom) tokens προστίθεται στο λεξιλόγιο του BERT που υποδεικνύουν προθέσεις, πεδία κλπ.

0.4.3 Πρόβλημα πρόβλεψης πρόθεσης

Κεφαλή intent status. Πραγματοποιούμε δυαδική κατηγοριοποίηση στην κωδικοποιημένη `[CLS]` αναπαράσταση για να προβλέψουμε την κατάσταση της πρόθεσης ως ενεργή ή όχι.

Κεφαλή intent value. Για κάθε πρόθεση $i \in I(n)$ πραγματοποιούμε δυαδική κατηγοριοποίηση στην κωδικοποιημένη του `[INTENT]` αναπαράσταση για να προβλέψουμε αν ο χρήστης εναλλάσσει σε αυτήν την πρόθεση.

Αν η κατάσταση της πρόθεσης είναι ενεργή τότε επιλέγουμε την πρόθεση με τη μεγαλύτερη πιθανότητα intent value. Διαφορετικά κρατάμε την προηγούμενη πρόθεση $prevInt(n)$.

0.4.4 Πρόβλημα πρόβλεψης ζητούμενων πεδίων

Κεφαλή requested status. Για κάθε πεδίο $s \in S(n)$ πραγματοποιούμε δυαδική κατηγοριοποίηση στην κωδικοποιημένη του `[SLOT]` αναπαράσταση στο Μέρος 4 για να αποφασίσουμε αν ζητείται στον τρέχοντα γύρο του χρήστη.

0.4.5 Πρόβλημα ανάθεσης τιμών πεδίων

Κεφαλή user status. Για κάθε πεδίο $s \in S_{inf}(n)$ βρίσκουμε την κατάσταση χρήστη user status χρησιμοποιώντας την κωδικοποιημένη του `[SLOT]` αναπαράσταση στο Μέρος 4 για να αποφασίσουμε αν η τιμή δίνεται στον τρέχοντα γύρο χρήστη. Οι πιθανές καταστάσεις χρήστη είναι: none, active και dontcare.

Κεφαλή categorical. Για τα κατηγορικά πεδία $s \in S_{inf}(n) \cap S_{cat}(n)$ πραγματοποιούμε δυαδική κατηγοριοποίηση για κάθε πιθανή τιμή $v \in V(s)$ στην κωδικοποιημένη της `[VALUE]` αναπαράσταση για να προβλέψουμε αν είναι παρούσα στο γύρο του χρήστη.

Κεφαλές start και end. Για τα μη-κατηγορικά πεδία $s \in S_{inf}(n) \cap S_{noncat}(n)$ βρίσκουμε την αρχή και το τέλος του αποσπάσματος (span) μέσα στο γύρο χρήστη πραγματοποιώντας κατηγοριοποίηση στην συνένωση κάθε token στο γύρο χρήστη και της κωδικοποιημένης `[SLOT]` αναπαράστασης.

Αν η κατάσταση χρήστη είναι active, η τιμή ή το απόσπασμα με τη μεγαλύτερη πιθανότητα επιλέγεται για το πεδίο. Αν η κατάσταση χρήση είναι dontcare, η ειδική τιμή dontcare ανατίθεται στο πεδίο.

0.4.6 Μεταφορά πεδίων

Ο χρήστης δεν δίνει πάντα ρητά την τιμή για το πεδίο αλλά μπορεί αντ'αυτού να αναφέρεται σε προηγούμενους γύρους. Για αυτό το λόγο, σχεδιάζουμε μηχανισμούς μεταφοράς πεδίο για να ανακτήσουμε τιμές για πεδία από την τρέχουσα ή από προηγούμενες υπηρεσίες.

Κεφαλή carryover status. Για κάθε πεδίο $s \in S_{inf}(n)$ προβλέπουμε την κατάσταση μεταφοράς (carryover status) χρησιμοποιώντας την κωδικοποιημένη του `[SLOT]` αναπαράσταση στο Μέρος 4 για να βρούμε την πηγή της τιμής του πεδίου. Για την κατάσταση μεταφοράς οι πιθανές τιμές είναι: none, in_sys_uttr (στο γύρο του συστήματος), in_service_hist (στο ιστορικό της υπηρεσίας) και in_cross_service_hist (στο ιστορικό άλλης υπηρεσίας).

Για το in_sys_uttr το πεδίο ενημερώνεται σύμφωνα με την τιμή που είναι παρούσα στον αμέσως προηγούμενο γύρο του συστήματος $sysUttrSlotValue(s)$. Για το in_service_hist το πεδίο ενημερώνεται σύμφωνα με την τιμή που είναι παρούσα σε προηγούμενες διαλογικές ενέργειες συστήματος της υπηρεσίας n, $prevSysSlotValue(s)$. Στις παραπάνω δύο περιπτώσεις, ο χρήστης αποδέχεται την τιμή που έχει δοθεί από το σύστημα και απλώς μεταφέρουμε την τιμή.

Κεφαλή cross-service. Για κάθε πεδίο $s' \in S_{prev}(n)$ πραγματοποιούμε δυαδική κατηγοριοποίηση στη συνένωση της κωδικοποιημένης της [SLOT] αναπαράστασης στο Μέρος 5 με την κωδικοποιημένη [SLOT] αναπαράσταση του s στο Μέρος 4 για να αποφασίσουμε αν πραγματοποιείται μεταφορά της τιμής από το πεδίο s' στο πεδίο s. Το πεδίο με τη μεγαλύτερη πιθανότητα s' χρησιμοποιείται ως πηγή για την τιμή s αν η κατάσταση μεταφοράς είναι in_cross_service_hist. Σε αυτήν την περίπτωση, αναθέτουμε την τιμή $prevSlotValue(s')$ στο πεδίο s.

Πρώτα ελέγχουμε την κατάσταση χρήστη και αν δεν είναι none ενημερώνουμε την τιμή ανάλογα με την έξοδο της. Αλλιώς, ελέγχουμε επίσης την κατάσταση μεταφοράς. Αν προβλέψει ότι πρέπει να γίνει μεταφορά, ενημερώνουμε την τιμή του πεδίου αναλόγως. Αν και η κατάσταση χρήστη και η κατάσταση μεταφοράς είναι none τότε η τιμή παραμένει η ίδια με την προηγούμενη κατάσταση διαλόγου, $prevUsrSlotValue(s)$.

0.4.7 Εκπαίδευση πολλαπλών εργασιών

Για τις κεφαλές κατηγοριοποίησης intent status, intent value, categorical, start, end και cross-service εξάγουμε τις πιθανότητες των κλάσεων με ένα feedforward νευρωνικό δίκτυο δύο επιπέδων. Για τις κεφαλές κατηγοριοποίησης requested status, user status και carryover status συνενώνουμε τα δυαδικά χειροποίητα χαρακτηριστικά $x_{bin}(s)$ μετά το πρώτο επίπεδο.

Εκπαιδεύουμε ταυτόχρονα όλες τις κεφαλές κατηγοριοποίησης, χρησιμοποιώντας το cross entropy loss για κάθε κεφαλή. Για το πρόβλημα της πρόβλεψης πρόθεσης το loss είναι $L_1 = w_1 L_{intstat} + w_2 L_{intval}$, για το πρόβλημα της πρόβλεψης ζητούμενων πεδίων το loss είναι $L_2 = L_{reqstat}$ και για το πρόβλημα της ανάθεσης τιμών πεδίων το loss είναι $L_3 = w_3 L_{usr} + w_4 L_{carry} + w_5 L_{cat} + w_6 L_{start} + w_7 L_{end} + w_8 L_{cross}$. Τελικά, το συνολικό loss ορίζεται ως $L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3$.

0.5 Πειράματα

0.5.1 Εξαγωγή ετικετών

Για να εξάγουμε τις ετικέτες (labels) για την κατάσταση χρήστη και μεταφοράς χρησιμοποιούμε τις διαλογικές ενέργειες χρήστη και ψάχνουμε προηγούμενους γύρους και καταστάσεις διαλόγου για να βρούμε την πηγή για το πεδίο. Θεωρούμε ένα πεδίο informable αν και μόνο αν είναι είτε απαιτούμενο είτε προαιρετικό σε μία τουλάχιστον πρόθεση. Για κάθε γύρο τρέχουμε το μοντέλο μόνο στις υπηρεσίες που εμπλέκονται (υπηρεσίες με τουλάχιστον μια αλλαγή στην κατάσταση του διαλόγου στο γύρο) σύμφωνα με τις πραγματικές (ground-truth) καταστάσεις διαλόγου και κατά την εκπαίδευση και κατά την αξιολόγηση για να είναι δίκαιη η σύγκριση με άλλες εργασίες. Η είσοδος του μοντέλου περιέχει πραγματικές προηγούμενες καταστάσεις διαλόγου κατά την εκπαίδευση και κατά την αξιολόγηση χρησιμοποιούνται αυτές που έχουν προβλεφθεί προηγουμένως.

Πίνακας 1: *Σύγκριση με άλλες εργασίες*

Σύστημα	Μοντέλο	Παράμετροι	JGA	Intent Acc	Req Slot F1
SGD-baseline [1]	$\text{BERT}_{\text{BASE}}$	**110 εκ.**	25.4	90.6	96.5
SGP-DST [19]	$6 \times \text{BERT}_{\text{BASE}}$	660 εκ.	72.2	91.9	99.0
paDST [21]	$3 \times \text{RoBERTa}_{\text{BASE}}$+ $\text{XLNet}_{\text{LARGE}}$	715 εκ.	**86.5**	94.8	98.5
D3ST [22] (Base)	T5_{BASE}	220 εκ.	72.9	97.2	98.9
D3ST [22] (Large)	T5_{LARGE}	770 εκ.	80.0	97.1	99.1
D3ST [22] (XXL)	T5_{XXL}	11 δισ.	86.4	**98.8**	**99.4**
Προτεινόμενο (διάμεσος εκτελέσεων)	$\text{BERT}_{\text{BASE}}$	**110 εκ.**	82.7	94.6	**99.4**
Προτεινόμενο (μέσος όρος 3 εκτελέσεων)	$\text{BERT}_{\text{BASE}}$	**110 εκ.**	82.5 ± 1.0	94.7 ± 0.5	**99.4** ± 0.1

Πίνακας 2: *Ablation study*

Σύστημα	JGA	Avg GA
Προτεινόμενο	**82.7**	**95.2**
χωρίς διαλογικές ενέργειες συστήματος	71.9	91.6
με περιγραφές πεδίων	78.3	94.1
χωρίς προηγούμενες καταστάσεις	79.8	94.0
χωρίς επαύξηση σχήματος	80.5	94.9
χωρίς επαύξηση σχήματος & word dropout	78.1	94.3
χωρίς δυαδικά χαρακτηριστικά	81.0	94.4

0.5.2 Εκπαίδευση

Χρησιμοποιούμε την υλοποιήση των BERT uncased μοντέλων που παρέχεται από το huggingface [2]. Για όλα μας τα πειράματα χρησιμοποιούμε μέγεθος για το batch 16 και dropout rate 0.3 για τις κεφαλές κατηγοριοποιήσεις. Χρησιμοποιούμε τον AdamW αλγόριθμο βελτιστοποίησης [23] με γραμμικό warmup διάρκειας το 10% των βημάτων εκπαίδευσης και ρυθμό εκπαίδευσης 2e-5. Επιλέγουμε το μοντέλο που έχει την βέλτιστη απόδοση με βάση τη μετρική JGA στο σύνολο ανάπτυξης (development set).

0.5.3 Προεπεξεργασία και επαύξηση

Προεπεξεργαζόμαστε τα στοιχεία του σχήματος και τις διαλογικές ενέργειες του συστήματος αφαιρώντας τις κάτω παύλες και χωρίζοντας τις λέξεις όταν είναι στις μορφές CamelCase και snake_case. Αντικαθιστούμε τυχαία ($p = 0.1$) τα tokens εισόδου στον γύρο του χρήστη με το token `[UNK]` (word dropout) και ανακατεύουμε τη σειρά εμφάνισης των στοιχείων του schema στα Μέρη 3-5 κατά τη διάρκεια της εκπαίδευσης όπως προτείνεται από [24]. Επιπλέον εφαρμόζουμε τυχαία ($p = 0.1$) επαύξηση δεδομένων (επαύξηση σχήματος) μέσω αντικατάστασης συνωνύμων και τυχαίας αναδιάταξης στις πρόθεσεις, πεδία και στις τιμές στα Μέρη 3-4 μέσω του [25].

Πίνακας 3: *Επίδραση των μηχανισμών μεταφοράς*

Σύστημα	JGA	Avg GA
Προτεινόμενο	**82.7**	**95.2**
χωρίς in_sys_uttr	62.8	87.0
χωρίς in_service_hist	76.4	92.7
χωρίς in_cross_service_hist	66.8	84.4
SGD-baseline [1]	25.4	56.0
χωρίς in_service_hist & in_cross_service_hist	61.6	81.9
κανένας μηχανισμός	36.5	68.5

0.6 Αποτελέσματα και συζήτηση

0.6.1 Σύγκριση με άλλες εργασίες

Στον πίνακα 1 συγκρίνουμε το δικό μας (προτεινόμενο) μοντέλο με τα SGD-baseline, SGP-DST, paDST και τρεις υλοποιήσεις του D3ST με μεταβλητό μέγεθος. Το SGD-baseline [1] πραγματοποιεί fine-tuning στο BERT με τους δύο τελευταίους γύρους ως είσοδο και χρησιμοποιεί υπολογισμένα εκ των προτέρων BERT embeddings για το σχήμα. Το SGP-DST [19] χρησιμοποιεί τους δύο τελευταίους γύρους και μηχανισμούς μεταφοράς για να ανακτήσει τιμές για πεδία που είχαν αναφερθεί σε προηγούμενους γύρους. Το paDST [21] και το D3ST [22] κωδικοποιούν ολόκληρο το ιστορικό διαλόγου μέχρι τον τρέχοντα γύρο και υπολογίζουν την κατάσταση διαλόγου από την αρχή. Αναφέρουμε τις μετρικές και των αριθμό παραμέτρων στα προεκπαιδεύμενα μοντέλα που γίνονται fine-tuned από κάθε μέθοδο.

Η μέθοδός μας ξεπερνάει ξεκάθαρα το SGP-DST σε όλα τα προβλήματα υποδεικνύοντας ότι οι στρατηγικές μας είναι αποτελεσματικές. Κάποια από τα μοντέλα που χρησιμοποιούν ολόκληρο το διάλογο ξεπερνούν σε απόδοση το μοντέλο μας, ιδίως όταν χρησιμοποιούν πολύ περισσότερες παραμέτρους (D3ST XXL) ή εφαρμόζουν περισσότερα χειροποίητα χαρακτηριστικά, ειδικούς κανόνες και επαύξηση διαλόγου μέσω back-translation (paDST). Συνολικά, η προτεινόμενη προσέγγιση επιτυγχάνει απόδοση κοντά στο state-of-the-art παρά το πολύ μικρότερο μοντέλο και την πιο σύντομη αναπαράσταση εισόδου.

0.6.2 Ablation study

Πραγματοποιούμε ένα ablation study (Πίνακας 2) για να δείξουμε τη συνεισφορά κάθε μίας από τις προτεινόμενες στρατηγικές για το πρόβλημα της ανάθεσης τιμών πεδίων. Η αντικατάσταση του γύρου συστήματος με ένα σύνολο από διαλογικές ενέργειες (χωρίς διαλογικές ενέργειες συστήματος) έχει τη μεγαλύτερη επίπτωση στην απόδοση (βλέπε ακολουθία εισόδου Μέρος 1 στο Σχήμα 4). Οι διαλογικές ενέργειες περιέχουν πολύ βασικές πληροφορίες όπως τα ονόματα των πεδίων και τις αντίστοιχες τιμές τους που βοηθάνε το μοντέλο μας να αναγνωρίσει ποια πεδία ζητούνται, προσφέρονται (Offer), επιβεβαιώνονται (Confirm) κλπ και να προβλέψει την κατάσταση χρήστη και μεταφοράς με μεγαλύτερη ακρίβεια. Η απόδοση πέφτει όταν χρησιμοποιούμε και τις περιγραφές για τα informable πεδία της τρέχουσας υπηρεσίας (με περιγραφές πεδίων, βλέπε Μέρη 3-4 της εισόδου). Αφαιρώντας την προηγούμενη

[2] https://huggingface.co/docs/transformers/model_doc/bert

πρόθεση και προηγούμενες τιμές για πεδία στα Μέρη 3-4 (χωρίς προηγούμενες καταστάσεις) παρατηρούμε ότι η απόδοση πέφτει αλλά επίσης η εκπαίδευση γίνεται πιο γρήγορη λόγω της μικρότερης ακολουθίας εισόδου. Παρατηρούμε επίσης βελτίωση όταν πραγματοποιούμε επαύξηση σχήματος και word dropout πιθανότητα επειδή αυτές οι στρατηγικές βοηθούν την αποφυγή υπερεκπαίδευσης (χωρίς επαύξηση σχήματος & word dropout). Τα χειροποίητα δυαδικά χαρακτηριστικά μπορούν να ωφελήσουν ελάχιστα το σύστημα (χωρίς δυαδικά χαρακτηριστικά).

0.6.3 Επίδραση των μηχανισμών μεταφοράς πεδίων

Στον Πίνακα 3 δείχνουμε την επίδραση των διαφόρων μηχανισμών μεταφοράς πεδίων. Για αυτά τα πειράματα το μοντέλο εκπαιδεύεται μία φορά και κατά τη διάρκεια της αξιολόγησης αντικαθιστούμε κάθε κατηγορία κατάστασης μηχανισμού με το "none". Όπως αναμενόταν, χωρίς το "in_sys_uttr" παρατηρείται η μεγαλύτερη επίπτωση στην απόδοση. Το "in_cross_service_hist" είναι επίσης σημαντικό λόγω του μεγάλου αριθμού multi-domain διαλόγων. Αφαιρώντας το "in_service_hist" υπάρχει λιγότερη επίδραση στην απόδοση. Χωρίς το "in_service_hist" και το "in_cross_service_hist" (χρησιμοποιώντας μόνο τους δύο τελευταίους γύρους) εξακολουθούμε να επιτυγχάνουμε μεγαλύτερη ακρίβεια από το SGD-baseline.

0.6.4 Συζήτηση

Δείξαμε ότι το προτεινόμενο σύστημα μπορεί να βελτιώσει δραστικά την απόδοση σε σχέση με το σύστημα SGP-DST που χρησιμοποιεί επίσης μηχανισμούς μεταφοράς πεδίων. Από τα πειραματικά αποτελέσματα συμπεραίνουμε ότι οι στρατηγικές για την δημιουργία μιας αποδοτικής ακολουθίας εισόδου είναι αποτελεσματικές. Παράλληλα, το σύστημα χρησιμοποιεί μόνο ένα BERT μοντέλο και η φειδωλή ακολουθία εισόδου επιτρέπει την επίλυση των τριών προβλημάτων με ένα μόνο πέρασμα από το BERT για κάθε γύρο. Αυτό κάνει το σύστημα περισσότερο υπολογιστικά αποδοτικό.

Σε σύγκριση με το state-of-the-art, η πιο αξιόλογη διαφορά είναι ότι αξιοποιούμε μηχανισμούς μεταφοράς πεδίων, όμως η κωδικοποίηση ολόκληρου του ιστορικού του διαλόγου έχει καλύτερη απόδοση. Στη δικιά μας προσέγγιση τα ονόματα των στοιχείων του σχήματος είναι πιο αποδοτικά από τις πλήρεις περιγραφές τις οποίες δεν καταφέραμε να ενσωματώσουμε με επιτυχία στο μοντέλο μας. Επιπλέον, το προεκπαιδευμένο μοντέλο μας (BERT) είναι μικρότερο. Το paDST χρησιμοποιεί μεγάλο αριθμό χειροποίητων χαρακτηριστικών και κανόνων που απαιτούν χειροκίνητη σχεδίαση και ενδεχομένως είναι υπερβολικά εξειδικευμένα για το σύνολο δεδομένων. Από την άλλη, το D3ST πετυχαίνει μεγαλύτερο JGA από το προτεινόμενο μοντέλο μας μόνο με τη μεγαλύτερη έκδοση του T5 που έχει 100 φορές περισσότερες παραμέτρους από το BERT. Συνολικά, το σύστημά μας είναι πιο υπολογιστικά αποδοτικό και έχει ικανότητα επέκτασης (scalable) σε μεγαλύτερα σχήματα και διαλόγους.

0.7 Συμπεράσματα

Σε αυτή τη διπλωματική εργασία, μελετήσαμε σε βάθος το αντικείμενο του schema-guided dialogue state tracking. Κάναμε μια εισαγωγή σε σημαντικές έννοιες της μηχανικές

μάθησης και αναφέραμε κάποιες πρόσφατες σημαντικές προόδους στη βαθιά μάθηση. Αναλύσαμε τα RNNs, τα LSTMs, το μηχανισμό προσοχής και τον transformer καθώς επίσης και δύο σημαντικές και ευρέως χρησιμοποιούμενες τεχνικές μάθησης: τη μεταφορά μάθησης και τη μάθηση πολλαπλών εργασιών. Έπειτα εστιάσαμε στο πεδίο της επεξεργασίας φυσικής γλώσσας. Πιο συγκεκριμένα, αφού κάναμε μια εισαγωγή σε παραδοσιακά γλωσσικά μοντέλα προχωρήσαμε στην παρουσίαση μοντέρνων γλωσσικών μοντέλων βασισμένων στον transformer όπως το BERT και το T5.

Κάναμε μια ανασκόπηση των διαλογικών συστημάτων εξετάζοντας μεθόδους, προκλήσεις και τρόπους αξιολόγησης των δύο κύριων κατηγοριών: διαλογικά συστήματα ανοιχτής συζήτησης (open-domain) και προσανατολισμένα σε συγκεκριμένο σκοπό (task-oriented). Στη συνέχεια επικεντρώσαμε το ενδιαφέρον μας σε στο dialogue state tracking και παρουσιάσαμε την τελευταία έρευνα που έχει γίνει στο συγκεκριμένο πεδίο . Παρατηρήσαμε ότι τα συστήματα καθοδηγούμενα από το σχήμα (schema-guided), που αποσκοπούν στο να διαχωρίσουν το μοντέλο από τις υποστηριζόμενες υπηρεσίες, έχουν κεντρίσει το ενδιαφέρον της ερευνητικής κοινότητας τελευταία.

Πρώτα μελετήσαμε το schema-guided dialogue state tracking και κατασκευάσαμε δύο συστήματα αναφοράς. Στη συνέχεια προτείναμε ένα καινοτόμο σύστημα πολλαπλών εργασιών για το schema-guided dialogue state tracking βασισμένο σε ένα μόνο BERT μοντέλο και μια αποδοτική και φειδωλή αναπαράσταση εισόδου. Το σύστημά μας αντιμετωπίζει τρία κρίσιμα προβλήματα DST ταυτόχρονα. Απόδοση κοντά στο state-of-the-art επιτυγχάνεται χρησιμοποιώντας πολύ μικρότερο μοντέλο και κωδικοποίηση εισόδου. Μεταξύ των διαφόρων προτεινόμενων βελτιώσεων στο μοντέλο δείχνουμε ότι η σύνοψη του αμέσως προηγούμενου γύρου του συστήματος με διαλογικές ενέργειες δίνει τη μεγαλύτερη αύξηση στην απόδοση. Στρατηγικές όπως η πρόσθεση προηγούμενων καταστάσεων διαλόγων, η επαύξηση δεδομένων και η προσθήκη χειροποίητων χαρακτηριστικών βελτιώνουν περαιτέρω την απόδοση.

Πιστεύουμε ότι αυτές οι στρατηγικές μπορούν να καθοδηγήσουν την σχεδίαση συστημάτων DST που χαρακτηρίζονται από ακρίβεια, επίδοση, ανεξαρτησία από την οντολογία και ικανότητα επέκτασης σε μεγάλους multi-domain διαλόγους, πράγμα που είναι σημαντικό σε εφαρμογές στον πραγματικό κόσμο.

0.8 Μελλοντικές προεκτάσεις

Στο μέλλον πιθανές μελλοντικές προεκτάσεις μπορούν να συμπεριλαμβάνουν:

- **Την αντικατάσταση του BERT από ισχυρότερα μοντέλα που έχουν υψηλότερη zero-shot απόδοση και ενδεχομένως μπορούν να βελτιώσουν την ακρίβεια σε υπηρεσίες που δεν έχουν δει κατά την εκπαίδευση.** Για παράδειγμα, παρόλο που το D3ST ακολουθεί μια απλή προσέγγιση παραγωγής ακολουθίας (seq2seq), έχει state-of-the-art απόδοση όταν χρησιμοποιεί τη μεγαλύτερη έκδοση του T5.

- **Περαιτέρω προεκπαίδευση των μοντέλων σε σχετικά προβλήματα.** Στο Κεφάλαιο 4 δείχνουμε ότι η προεκπαίδευση σε προβλήματα όπως η μηχανική κατανόηση κειμένων (machine reading comprehension - MRC), για την οποία περισσότερα σύνολα δεδομένων μεγάλης κλίμακας είναι διαθέσιμα, μπορεί να ωφελήσει το DST. Το

schema-guided DST είναι ακόμα πιο σχετικό με το MRC από τα παραδοσιακά προβλήματα DST καθώς δείχνουμε ότι η περιγραφή του πεδίου/πρόθεσης μπορεί να θεωρηθεί ως η ερώτηση στο πρόβλημα του MRC (βλέπε Κεφάλαιο 4). Παρόλα αυτά, η προσέγγισή μας απαντάει πολλαπλές 'ερωτήσεις' ταυτόχρονα και έτσι η προεκπαίδευση παραδοσιακών μοντέλων MRC που απαντούν μόνο μια ερώτηση για κάθε χωρίο κειμένου μπορεί να χρειάζεται να τροποποιηθεί.

- **Στρατηγικές για την ενσωμάτωση προηγούμενων γύρων και περισσότερων πληροφοριών σχετικών με το σχήμα, όπως οι περιγραφές των στοιχείων του, χωρίς την αύξηση του μήκους της εισόδου.** Για παράδειγμα, ένα αναδρομικό μοντέλο (π.χ. LSTM) που δουλεύει στο επίπεδο του γύρου διαλόγου θα μπορούσε να μάθει να κωδικοποιεί χρήσιμες πληροφορίες από προηγούμενους γύρους.

- **Περισσότερη επαύξηση δεδομένων για την αντιμετώπιση των προβλημάτων υπερεκπαίδευσης στις υπηρεσίες που το μοντέλο βλέπει κατά την εκπαίδευση.** Ο αριθμός αυτών των υπηρεσιών είναι περιορισμένος και επομένως το μοντέλο είναι ευάλωτο σε υπερεκπαίδευση και 'απομνημόνευσή' τους. Δείξαμε μια απλή τεχνική επαύξησης δεδομένων αλλά πιο εκλεπτυσμένες μπορούν επίσης να εξερευνηθούν. Ως παράδειγμα, μελλοντικές επεκτάσεις μπορούν να υλοποιούν back-translation (μετάφραση από τα αγγλικά σε μια δεύτερη γλώσσα και ξανά μετάφραση από τη δεύτερη γλώσσα στα αγγλικά) με μοντέλα transformer.

- **Ενσωμάτωση του καθοδηγούμενου από το σχήμα προτύπου και σε άλλα υποσυστήματα του διαλόγου ή ακόμα σε διαλόγους από άκρη σε άκρη (end-to-end).** Προκειμένου να κατασκευάσουμε ένα σύστημα που δουλεύει στον πραγματικό κόσμο είναι σημαντικό να εκπαιδεύσουμε ένα υποσύστημα γνωστό ως dialogue policy (βλέπε Κεφάλαιο 4), ωστόσο το πρόβλημα δεν έχει μελετηθεί αρκετά. Μια πιθανή κατεύθυνση είναι συστήματα από άκρη σε άκρη που μαθαίνουν παράλληλα να προβλέπουν την κατάσταση του διαλόγου αλλά και να αποκρίνονται στο χρήστη.

Chapter 1

Introduction

1.1 Motivation

Machines that can have natural conversations with humans have been a theme in utopian science fiction. However, due to the latest advances in artificial intelligence and particularly deep learning, intelligent dialogue systems are becoming a reality, achieving what was once just a product of imagination. Popular virtual assistants such as Siri, Cortana, Google Assistant, Alexa and others are able to perform tasks and/or entertain users through conversations.

An important category of conversational agents is task-oriented dialogue systems. The aim of such systems is to assist users in accomplishing daily activities like reserving a restaurant, booking tickets etc. A critical component of a task-oriented dialogue system is Dialogue State Tracking (DST) which tracks the user goal over multiple turns of dialogue. Based on a spoken utterance and the dialogue history, DST predicts the dialogue state which represents the user goal. The predicted dialogue state is then used by other components to retrieve elements from a database, perform the actions requested by the user and respond accordingly [15]. Most recently proposed systems have been multi-domain; they can handle multiple conversational domains in the same dialogue.

Motivated by the ever-increasing number of diverse services used by commercial task-oriented systems, the schema-guided paradigm was developed [1]. The goal of this paradigm is to allow DST models to work across new, unseen services relying on the knowledge acquired by the services they saw during training. The natural language description of the services' schemata guides the model in such new scenarios.

At the same time, recent pre-trained transformer language models such as BERT and T5 show outstanding performance on many natural language understanding problems. During the pre-training phase on large datasets, they learn to understand language by examples and then they can be adapted to specific tasks such as DST through a process known as fine-tuning. Such models are therefore suitable for dialogue systems because of their generalization ability.

1.2 Contributions

In this thesis, we propose a multi-task BERT-based model that jointly performs three DST tasks: intent prediction, requested slot prediction and slot filling. Furthermore, we construct an efficient and parsimonious abstracted representation of the dialogue

and schema that is shown to significantly improve performance while achieving greater computational efficiency. Our proposed model significantly outperforms the baseline system and achieves near state-of-the-art performance. Extensive ablation studies reveal the impact of each strategy of our model on the slot filling task.

1.3 Thesis outline

In Chapter 2 we provide the necessary machine learning background knowledge. We first introduce traditional machine learning approaches which we build on to explain deep learning. We describe how neural networks, recurrent neural networks, the attention mechanism and transformers work. We also discuss about two widely used learning techniques: transfer learning and multi-task learning.

In Chapter 3 we introduce Natural Language Processing (NLP), the most common NLP tasks and modern approaches to NLP: word embeddings and language modeling with transformers.

In Chapter 4 we discuss about dialogue systems focusing on task-oriented systems found in virtual assistants. We study how deep learning can be applied for the various dialogue system components.

In Chapter 5, after a literature review on the task of Schema-Guided Dialogue State Tracking, we introduce two baseline systems and we present our proposed system, a Multi-Task BERT-based model. We conduct extensive experiments and ablation studies to reveal the impact of our proposed strategies. We compare it to other models and we highlight the benefits of our approach.

In Chapter 6 we summarize our work and our main findings and we provide ideas for future work.

Chapter 2

Machine Learning

Machine learning (ML) is a subfield of Artificial Intelligence. It studies algorithms that can automatically improve (learn) with the help of data. Such algorithms are not explicitly programmed to perform their tasks but they instead leverage models that learn to make predictions from the training data. ML is a suitable solution to a problem if its complexity makes traditional algorithms hard to program. For example, autonomous driving and natural language understanding are two tasks that can be performed easily by humans but writing rule-based algorithms for them would be far from trivial. ML is also suitable when the need for adaptivity is important. Because ML models rely on data, the scale of which becomes larger and larger nowadays, they are more robust - able to generalize to new inputs [26].

ML has applications in numerous fields including computer vision, natural language processing, speech recognition, healthcare and robotics. It offers solutions that avoid limitations of traditional algorithms. The availability of large scale data and computational advancements have made ML increasingly popular and more accessible. Researchers design increasingly more sophisticated ML approaches which require less manual effort and are more effective.

2.1 Machine learning approaches

2.1.1 Supervised learning

In supervised learning, the training data examples are input-output pairs: (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) and we assume that there is an unknown function $y = f(x)$ which maps the inputs x_i (**features**) to the outputs y_i (**labels**). Models have to predict a function $\hat{f}$ that approximates f based on the training data [27].

Two common types of supervised learning are **classification** and **regression**. In classification, examples belong to categories or classes and the problem is to identify the correct class based on the features. For example, given an image, a classification task is to predict whether the animal in the image is a dog or a cat. In regression, the task is to find the relationship between the input variables (features) and output variables (labels). The difference with classification is that labels are not limited to a set of possible classes but are numerical values. An example of a regression task is predicting a stock price based on information like previous prices.

2.1.2 Unsupervised learning

Unsupervised learning algorithms learn patterns from unlabelled data. Instead of relating features to their associated labels, they learn the structure of the data relying solely on the features. Unsupervised tasks include *clustering* in which data have to be separated into groups (clusters) with similar features and *anomaly detection* which aims to identify rare items (outliers) in a set of data that may indicate an anomaly [28].

2.1.3 Reinforcement learning

In reinforcement learning, agents interact with an environment by taking actions based on the state of the environment. These agents try to maximize the value of the **reward function** which relates to how close the outcomes are to the desired result. In this way, they favor actions which are more likely to have desirable effects and avoid actions which may produce less desirable effects. Different from supervised learning, the ground-truth labels are not known, the only learning signal is the reward that they observe in the environment after their actions. Learning involves finding a balance between **exploration** (trying actions that have not been tested before) and **exploitation** (relying on behaviors that are known to be effective) [29].

2.2 Machine learning concepts

2.2.1 Loss function

The loss function maps the model's predictions to a real number representing the cost of those predictions. Therefore, a small value for the loss is better as it suggests that the model has learned from the data. ML methods find the optimal parameters for the function $\hat{f}$ by minimizing the total loss function of the examples on the training set. The total loss over the training set is called cost function although the terms loss and cost function are often used intechangeably.

Mean Squared Error (MSE) is a loss function used in regression problems.

$$MSELoss = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

Cross-entropy is a loss function commonly used in classification problems. For every example the loss is calculated by the formula:

$$CrossEntropyLoss = \sum_{i=1}^{C} y_i \cdot log\hat{y}_i \tag{2.2}$$

where C the number of classes.

Hinge loss is another loss function for classification problems. If the possible outputs are $t = \pm 1$ then for a single example it is defined as:

$$HingeLoss = max(0, 1 - t \cdot y) \tag{2.3}$$

2.2.2 Gradient descent

Gradient descent is an iterative optimization algorithm which finds the local minimum of a differentiable function based on its first order gradients. It is the most popular optimization algorithm in ML. Gradient descent is used to find the model parameters that minimize the loss function. The model starts with some (often random) initial parameters which are then adjusted in steps. The new parameters are found by taking steps in the opposite direction of the gradient of the cost function.

Formally, in each step n the parameters $\mathbf{p}_n$ are updated according to the following equation:

$$\mathbf{p}_n = \mathbf{p}_{n-1} - \eta \nabla J(\mathbf{p}_n) \tag{2.4}$$

where $\eta > 0$ is the learning rate, a parameter that controls the speed of training and $J(\cdot)$ the cost function. The learning rate should be neither too small nor too large. A small learning rate often leads to slow convergence making the training process longer. On the other hand, a large learning rate may not allow the model to arrive at the optimal parameters because it bounces back and forth near those parameters.

There are three variants of gradient descent based on the amount of data processed before taking a step. Processing the entire dataset once corresponds to a training **epoch** but during one epoch there may be multiple gradient descent steps.

1. Batch gradient descent: The cost function is calculated by processing all training examples and then parameters are updated. This gives the most accurate approximation for the gradient of the cost function and it therefore results in a stable convergence. However, it may converge to a suboptimal local minimum and requires the entire dataset to be loaded in memory.

2. Stochastic gradient descent (SGD): The cost function is calculated for each training example separately and then parameters are updated. The more frequent updates make training faster but because the gradient is approximated, noisy gradients can interfere with training stability.

3. Mini-batch gradient descent: This is the most popular method that combines the strengths of the first two variants. The dataset is split into training **batches** (the size of which is usually a power of 2 between 16 and 512) and the gradient is estimated based on these training examples before taking a step. This gives a more accurate gradient estimation than SGD and is more computationally efficient than both other variants because batches are small enough to fit in the GPU memory.

2.2.3 Bias-variance tradeoff

The **bias** error is caused by imposing too many assumptions for the function $\hat{f}$ and thus not being able to learn the relationships between the features and the labels. For example, a linear classifier often has a high bias when the data distribution is complex and a linear function is not enough to explain them. This phenomenon is called **underfitting**.

Underfitted models cannot find the required patterns in the data in order to make accurate predictions.

The **variance** error is caused by learning too much from the data and as a result modelling the random noise present in the dataset. Models with a high variance tend to fluctuate a lot even when small changes are observed in the data. This phenomenon is called **overfitting**. The ability of an overfitted model to generalize to new, unseen examples is often poor.

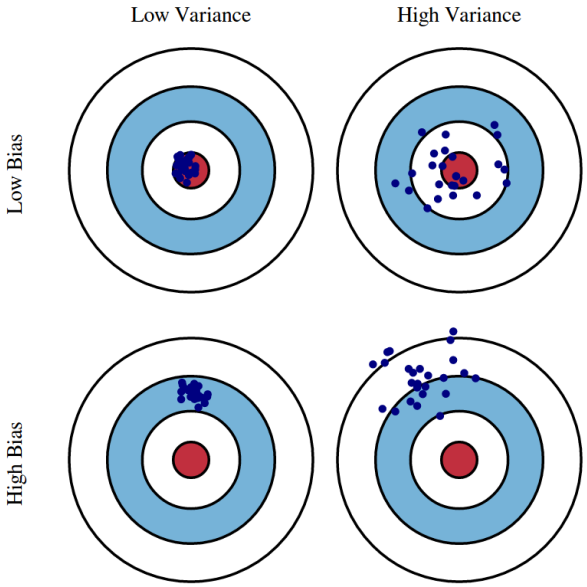


Figure 2.1: *Visual illustration of the bias and variance errors. The top left image is the perfect scenario with the right balance for the bias-variance tradeoff. Source:* `http://scott.fortmann-roe.com/docs/BiasVariance.html`

When the model is too simple and has few parameters it usually has a high bias and a low variance. On the other hand, models with too many parameters generally have a lower bias and a higher variance. The difference between the two error types is called bias-variance tradeoff. It is important to find the right balance between bias and variance so that the model can effectively learn from the training data and at the same time be capable of generalizing to new examples.

2.3 Machine Learning Methods

2.3.1 Decision trees

Decision trees are non-parametric models which can be used to predict values based on simple rules on the features. To reach a conclusion about the predicted value a sequence of tests of certain conditions is performed. Therefore, the decision tree is traversed until a leaf node which represents the final prediction. Decision trees can be used for both classification and regression tasks. They are very easy to interpret and visualize but their main disadvantage is that they are prone to overfitting and thus do not always generalize to unseen data.

2.3.2 Support-vector machines

Support-vector machines (SVMs) perform binary linear classification by maximizing the width of the gap between samples of the two classes. However, SVMs can even be used when the data examples are not linearly separable by using the kernel trick and mapping them to a higher dimension where they are linearly separable. There are many hyperplanes that divide data into two classes but SVMs find the one where the distance between the nearest points from each class and the hyperplane is maximized.

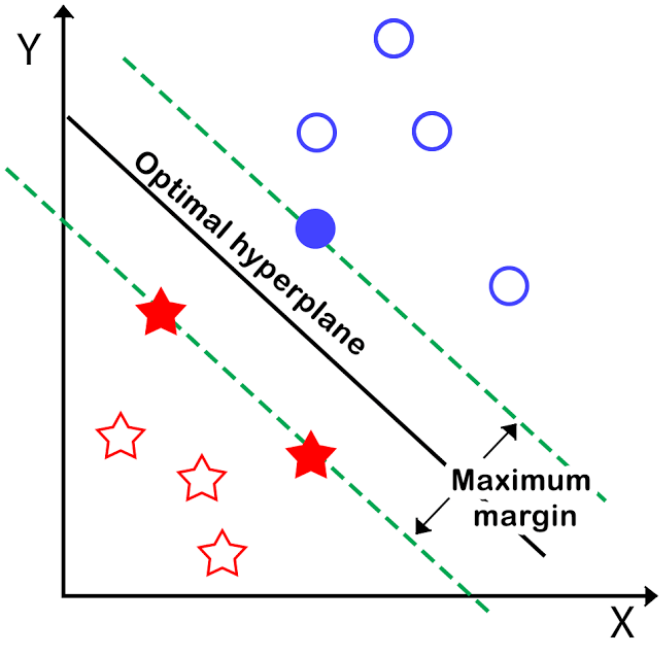


Figure 2.2: *SVM optimal hyperplane. Source:* https://medium.com/@cdabakoglu/what-is-support-vector-machine-svm-fd0e9e39514f

Assuming that the optimal hyperplane (see Figure 2.2) is described by:

$$\mathbf{w}^T\mathbf{x} - b = 0 \tag{2.5}$$

the SVM algorithm optimization problem involves minimizing $\|\mathbf{w}\|$ subject to $y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1$ for $i = 1, \ldots n$ where the i-th example is described by the feature vector $\mathbf{x_i}$ and the label $y_i \in \{1, -1\}$. The function that estimates the class for a new sample is:

$$\hat{f}(x_i) = sgn(\mathbf{w}^T\mathbf{x} - b) \tag{2.6}$$

If the examples are not linearly separable a loss function can be defined by incorporating the hinge loss (defined in Section 2.2.1) and thus allowing some examples to be misclassified:

$$Loss = \lambda\|\mathbf{w}\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right)\right)\right] \tag{2.7}$$

and by optimizing it we can find $\mathbf{w}$ and b.

2.3.3 Linear regression

For regression problems we can use linear regression when it is safe to assume that there is a linear relationship between features and labels. Assuming that there is only one input feature x, the output is given by:

$$y = mx + b \tag{2.8}$$

We can use the MSE loss function as defined in Section 2.2.1:

$$Loss = \frac{1}{N}\sum_{i=1}^{n}(y_i - (mx_i + b))^2 \tag{2.9}$$

and by optimizing it we can find m and b.

The above formulation describes **simple linear regression** where both the input x and the output y are scalars. However, we can easily generalize to multiple input variables (*multiple linear regression*) and multiple output variables (*general linear models*). Furthermore, generalized linear models are the extension of linear regression to classification problems where the labels are discrete rather than continuous. For example, **logistic regression** relies on the logistic function $f(x) = \frac{L}{1+e^{-k(x-x_0)}}$ to convert the output y (logit) to probabilities for each class.

2.4 Neural Networks and Deep Learning

2.4.1 Artificial Neural Networks

Artificial neural networks or simply neural networks are a class of models whose function is inspired by the animal brains. A neural network consists of connected **neurons** which transmit signals between them.

We can define a neuron as a function which takes a number of inputs, calculates their weighted sum and then passes it to a non-linear **activation function** to produce the final output. Formally a neuron with m inputs $x_1, x_2, \ldots, x_m$ produces the output:

$$y = \phi\left(\sum_{i=0}^{m} w_i x_i\right) \tag{2.10}$$

where $x_0 = 1$ is an additional fixed input, $w_0, w_1, \ldots, w_m$ are the weights corresponding to $x_0, x_1, \ldots, x_m$ and ϕ a non-linear activation function.

A neural network can be seen as a directed graph with the neurons represented as nodes and the weights between them represented as edges. It consists of input neurons which receive the inputs, output neurons which calculate the outputs (e.g. class probabilities in classification problems) and possibly hidden neurons between them.

Typically, by neural network we refer to a **feedforward network** whose neurons are organized in layers (see Figure 2.3). Each layer takes as input all neurons of the previous layer (fully-connected layers). Thus, the network consists of the input layer which takes as input the features, the hidden layer(s) which produce intermediate representations and

finally the output layer which calculates the output. Mathematically, the overall network computes the output vector $g(x)$ using the following equation:

$$g(x) = f^L\left(W^L f^{L-1}\left(W^{L-1}\cdots f^1\left(W^1 x\right)\cdots\right)\right) \tag{2.11}$$

where L is the number of layers, $W^l = (w^l_{jk})$ are the weights between layer l and $l-1$ and f^l is the activation function which is applied after the layer l.

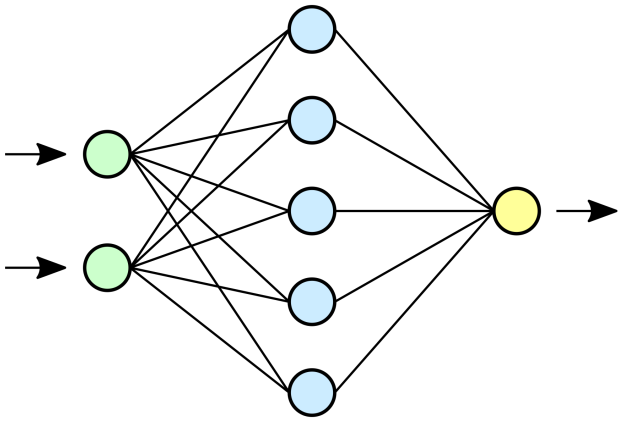


Figure 2.3: *A feedforward neural network with one hidden layer. Source:* https://commons.wikimedia.org/wiki/File:Neural_network.svg

2.4.2 Introduction to Deep Learning

Deep learning (DL) is a category of machine learning based on deep artificial neural networks. The main motivation behind DL is to limit the amount of feature engineering. Traditional ML approaches struggle to process input data of high dimensions like text or images and must rely on manual feature extraction in such scenarios. In constrast, deep networks with multiple layers are capable of processing raw data by automatically creating intermediate representations in the first layers and using them to make their predictions in the last layers. This technique enables the model to discover patterns in data on its own and is called **representation learning**.

2.4.3 Activation functions

In this section we will briefly mention some activation functions that are typically used in neural networks.

1. **Sigmoid** function. It maps a scalar x to a number in the range of $(0, 1)$ and is commonly used for output probabilities.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.12}$$

2. **Hyperbolic tangent (tanh)** function. It maps a scalar x to a number in the range

$(-1, 1)$.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.13}$$

3. **Rectified linear unit (ReLU)** function. It is commonly used as an activation functions for input or hidden layers.

$$ReLU(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{2.14}$$

4. **Gaussian Error Linear Unit (GELU)** function. An alternative to the ReLU function.

$$GELU(x) = \frac{1}{2}x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{2.15}$$

5. **Softmax** function. It takes as input a J-dimensional vector $\mathbf{x}$ and outputs positive numbers which sum to 1.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for } i = 1, \ldots, J \tag{2.16}$$

2.4.4 Learning through backpropagation

Backpropagation is an algorithm which enables the efficient calculation of the gradient of the cost function with respect to each weight in feedforward neural networks. These gradients are used by gradient descent to find the optimal weights as described in Section 2.2.2. It works by applying the chain rule starting from the end of the network (layers closer to the output) and working backwards. Backpropagation is a dynamic programming algorithm because redundant computations are avoided by leveraging intermediate computed terms.

Assume that the network is described by Equation 2.11 and for an input-output pair $(x_i, g(x_i))$ the cost function is $C(y_i, g(x_i))$ where y_i the ground-truth label. In this case, we want to compute the partial derivatives $\partial C/\partial w^l_{jk}$ with respect to the weights. Instead of computing these terms independently, backpropagation starts from the last (output) layer and works backwards finding the gradient with respect to the weighted input of each layer. These terms are calculated recursively and are used to find the partial derivatives. A detailed explanation of the algorithm can be found in [30].

2.4.5 Regularization

In Section 2.2.3 we discussed about underfitting and overfitting. Neural networks, especially deep neural networks, rarely underfit because they often have a lot of parameters, however overfitting is common. Techniques that help ML models avoid overfitting and generalize better are called regularization techniques. In this section we will describe the most widely used regularization techniques in (deep) neural networks.

- **L1 Regularization**. L1 regularization aims to penalize large values for the weights

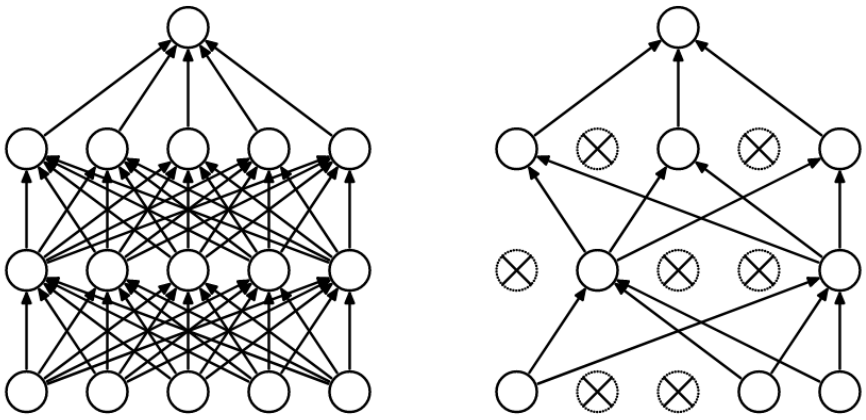
$\mathbf{w}$ of the network by adding the term $\|w\|$ to the loss function:

$$L' = L + \lambda\|\mathbf{w}\| \tag{2.17}$$

- **L2 Regularization or Weight Decay**. L2 regularization works similarly to L1 regularization but adds the term $\|\mathbf{w}\|^2$ to the loss function:

$$L' = L + \lambda\|\mathbf{w}\|^2 \tag{2.18}$$

- **Dropout**. Dropout [2] chooses a random subset of neurons during each training iteration and removes it. Because this random dropout of neurons is only performed during training, this method can be seen as an efficient averaging (ensemble) of different neural networks greatly improving generalization by forcing the neurons to learn representations independently of other neurons.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.4: *A neural network with 2 hidden layers before (a) and after (b) applying dropout. Source: [2]*

- **Early Stopping**. In early stopping we keep one small part of the training set (development or validation set) which is not fed to the model but is rather periodically used to estimate the generalization ability of the model as it is being trained. When we observe that the performance on the validation set starts getting worse we stop training as this may be an indicator of overfitting.

- **Data Augmentation**. The more training examples the network sees the less likely it is to overfit. Data augmentation creates more examples by artificially augmenting the dataset. For images, data augmentation techniques include scaling, flipping, rotating etc and for text swapping words, replacing with synonyms or deleting words are common options.

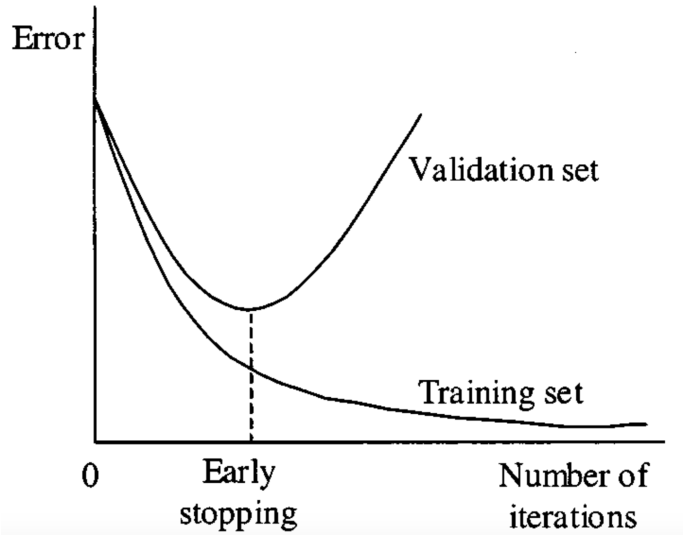


Figure 2.5: *An example of early stopping. Training stops when the validation set error starts increasing which indicates overfitting. Source:* https://www.researchgate.net/figure/Early-stopping-based-on-cross-validation_fig1_3302948

2.4.6 Recurrent neural networks

So far we have focused on feedforward networks which do not form cycles or loops. However, such networks are not suitable for problems in which the input features do not have a fixed size i.e. in sequence problems. Recurrent neural networks (RNNs) are types of neural networks which better model variable size sequences by keeping an internal hidden representation or state serving as a type of memory. Typically, for each input unit in the sequence they calculate the new hidden state based on the previous hidden state and the current input. This enables processing of sequences with variable lengths while keeping the number of parameters fixed (Figure 2.6).

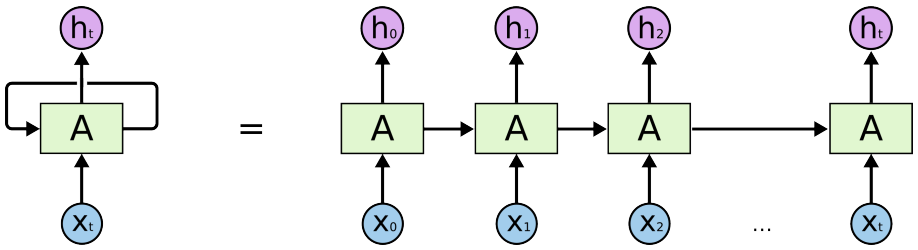


Figure 2.6: *A RNN model (left) and its unrolled structure (right). Source: [3]*

A type of a RNN network is the Elman network [31] and it works as follows: For each time step t, the input x_t and the previous hidden state h_{t-1} are used to derive the current hidden state h_t (Equation 2.19). Then, based on h_t the output y_t is given by Equation 2.20.

$$h_t = \sigma_h \left(W_h x_t + U_h h_{t-1} + b_h \right) \tag{2.19}$$

$$y_t = \sigma_y \left(W_y h_t + b_y \right) \tag{2.20}$$

where σ_h and σ_y are two activation functions and W_h, U_h, W_y, b_h, b_y are trainable param-

eters. Note that the training parameters are the same for every time step enabling the processing of potentially large input sequences without increasing the model size. The hidden state h_{t-1} is the only way to look back in previous parts of the sequence and it should therefore capture all the necessary information about the "past".

RNNs allow to perform various tasks on the processed sequence [4]. Some examples of RNN usages for different sizes of inputs and outputs (see Figure 2.7) include:

- **One to many**. The input is of fixed size and the output is a sequence. Example: creating image captions with input a fixed size image and output a sentence.
- **Many to one**. The input is a sequence and the output is of fixed size. Example: classifying a movie review as positive or negative.
- **Many to many (input and output with different sizes)**. Both the input and the outputs are sequences but they do not have the same size. These types of RNN models are commonly referred to as Sequence to Sequence (seq2seq) models [32]. Example: translation of a sentence from English to French.
- **Many to many (input and output with same sizes)**. Input and output sequences are of the same size. Alternatively, this can be seen as sequence labelling (assigning a label to each part of the sequence). Example: identifying named entities (persons, locations, organizations, etc) in a sentence (named entity recognition).

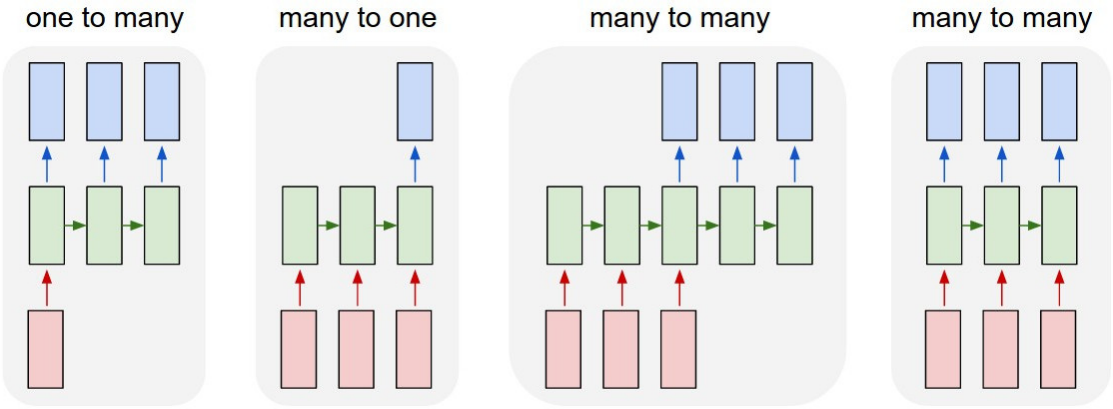


Figure 2.7: *Examples of RNN usages. Source: [4]*

Training of RNNs is possible with backpropagation through time (BPTT), an algorithm that extends standard backpropagation [33]. It works by unrolling the RNN model (see Figure 2.6) and accumulating errors across all time steps. However, it has been observed that for long sequences error gradients vanish (vanishing gradient problem) and consequently learning becomes slower. Modifications to the "vanilla" RNN model that have been proposed to eliminate the vanishing gradient problem include Long-Short Term Memory (LSTM) networks [34] and Gated Recurrent Units (GRUs) [35]. Based on [3] we will describe how LSTMs work.

LSTMs solve the vanishing gradient problem and enable the network to learn long-term relationships by adding the concept of the *cell state*. The cell state is not present in

the vanilla RNN and it allows for controlled addition or removal of information with gates as shown in Figure 2.8.

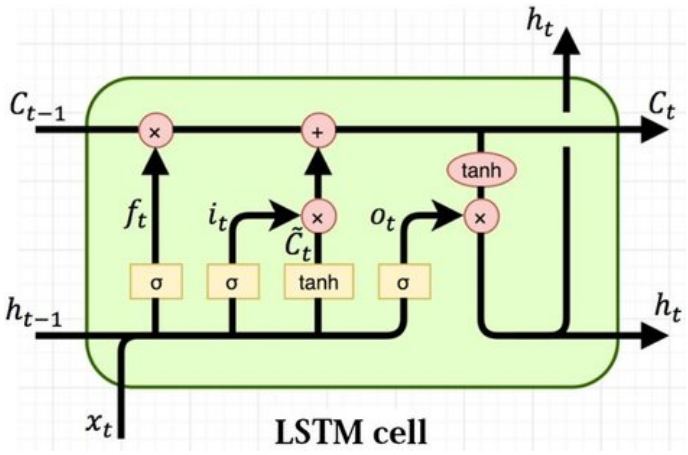


Figure 2.8: *A LSTM cell. The yellow rectangles represent the four neural network layers. Source:* https://www.researchgate.net/figure/Structure-of-the-LSTM-cell-and-equations-that-describe-the-gates-of-an-LSTM-cell_fig5_329362532

The following equations are used by the LSTM cell:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.21}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.22}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2.23}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.24}$$

$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right) \tag{2.25}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{2.26}$$

The function $tanh(\cdot)$ which outputs numbers in the range of $(-1, 1)$ is used as the activation function. Additionally, the sigmoid function $\sigma(\cdot)$ is used for gate layers outputting numbers between 0 and 1. The outputs of the gates are then multiplied with vectors to control which numbers of the vector to keep (close to 1) and which numbers to erase (close to 0).

- **Removing information from the cell state**. Based on the previous hidden state h_{t-1} and the current input x_t, the "forget gate layer" (Equation 2.21) controls which information to remove (forget) from the cell state.
- **Storing information to the cell state**. Equation 2.22 describes the "input gate layer" which decides which of the values in the cell state should be updated. The new candidate values $\tilde{C}_t$ are then created by another layer (Equation 2.23).
- **Updating the new cell state**. The cell state is updated based on f_t, i_t and $\tilde{C}_t$ (Equation 2.24).

- **Output**. The "output gate layer" (Equation 2.25) allows specific parts of the cell state to the output (Equation 2.26).

A limitation of RNNs is that only information from the past (previous time steps) is accessible. However, in some problems future information is also important. Bidirectional RNNs (Bi-RNNs) were proposed [36] to address this issue. Bi-RNNs process the sequence both in the forward direction as the standard unidirectional RNN and the backward direction (starting from the last token and working towards the first one). The backward mode is usually implemented with a separate hidden layer. Therefore, for every time step two hidden states are produced $\overrightarrow{h_t}$, $\overleftarrow{h_t}$, one for each direction and are concatenated to produce the final hidden state h_t. The vanilla RNN cells can be replaced by LSTM or GRU cells resulting in bidirectional LSTMs (Bi-LSTMs) and bidirectional GRUs (Bi-GRUs).

2.4.7 Attention mechanism

In the previous section we mentioned that RNNs (usually LSTMs) can be used for seq2seq problems like machine translation. These seq2seq models are usually composed of the **encoder** (an LSTM which processes the input sequence) and the **decoder** (another LSTM which generates the output sequence) [32]. The encoder produces the **context vector** which is usually its last hidden state and passes it to the decoder as its first hidden state. However, especially when the input sequence is large, it is difficult for the encoder to learn to capture all of its information to a fixed-sized context vector and as a result it may forget parts of the sequence.

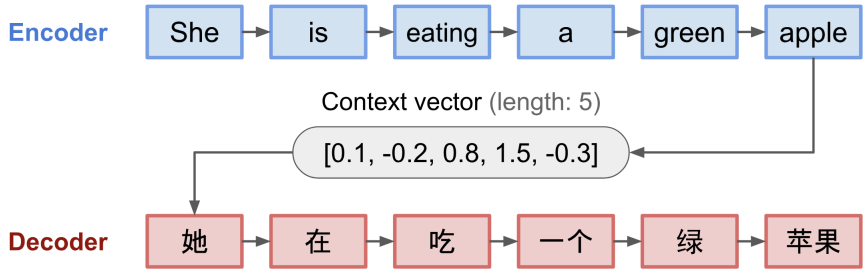


Figure 2.9: *An encoder-decoder model translating a sentence from English to Chinese. Source: [5]*

The attention mechanism was proposed to solve this problem [6]. Like the name suggests, it allows to pay attention to important parts of the input mimicking cognitive attention. Figure 2.10 shows how the original attention mechanism works. We suppose that the input sequence $[x_1, x_2, \ldots, x_T]$ is encoded with a Bi-RNN model and we want to generate the output sequence $[y_1, y_2, \ldots, y_{T'}]$. The decoder takes as input its previous hidden state s_{t-1}, the previous output y_{t-1} and a weighted sum of all of the encoder hidden states to produce s_t. By allowing the decoder to access all hidden states instead of only the last one, the model can reason more effectively for the entire input sequence. Furthermore, the attention mechanism allows different decoder steps to concentrate on different parts of the input sequence.

In the paper, the output conditional probabilities are defined as:

$$p(y_t \mid y_1, \ldots, y_{t-1}, \mathbf{x}) = g(y_{t-1}, s_t, c_t) \tag{2.27}$$

where the hidden state s_t is given by:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \tag{2.28}$$

The context vector c_t is the weighted sum of the encoder hidden states:

$$c_t = \sum_{j=1}^{T} a_{tj} h_j \tag{2.29}$$

and the weights are given by:

$$a_{tj} = \frac{\exp\left(e_{tj}\right)}{\sum_{k=1}^{T} \exp\left(e_{tk}\right)} \tag{2.30}$$

$$e_{tj} = \text{score}\left(s_{t-1}, h_j\right) \tag{2.31}$$

The function score$(\cdot, \cdot)$ (alignment model) which estimates the degree of alignment/similarity of two vectors is a feedforward network in the original paper: $\text{score}(s_{t-1}, h_t) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a [s_{t-1}; h_t])$. Other works study different variants to the original attention mechanism [37, 38]. Although, this mechanism was at first proposed to solve seq2seq text problems like machine translation, later works extended it to the computer vision field [39].

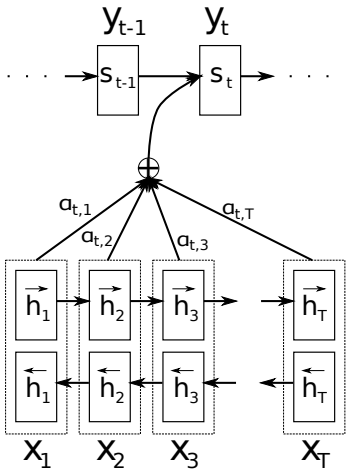


Figure 2.10: *Attention mechanism. Source: [6]*

2.4.8 The Transformer

In 2017, the paper "Attention is All you Need" [7] introduced the transformer, a seq2seq model which led to a revolution in the field of deep learning. In the previous section, we highlighted the importance of the attention mechanism as a way to concentrate on specific parts of the sequence. This helped RNNs improve their performance by eliminating

the problems arising from using only the last hidden state. However, the transformer proved that the sequential processing is not needed at all provided that attention is used. Transformer models process sequential data like RNNs but they remove the need for recurrent computations and this allows for more parallelization and therefore lower training times. Larger datasets can now be used and thus transformer models often outperform RNN-based models.

Attention. The paper relies on an attention type called "Scaled Dot-Product Attention". Suppose that the input consists of vectors which represent words (words can be represented as vectors called word embeddings which will be expained in detail in the next chapter). For each input vector x_i we calculate three vectors: the Query vector q_i, the Key vector k_i and the Value vector v_i:

$$q_i = x_i W_Q \tag{2.32}$$

$$k_i = x_i W_K \tag{2.33}$$

$$v_i = x_i W_V \tag{2.34}$$

where W_Q, W_K and W_V are learnable weight matrices. The attention score a_{ij} from token x_i to token x_j is computed as the dot-product between the query of x_i and the key of x_j:

$$a_{ij} = q_i \cdot k_j \tag{2.35}$$

We define the matrices Q, K and V where the i-th row is the vector q_i, k_i and v_i respectively. Then the attention output is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.36}$$

where the division by $\sqrt{d_k}$ helps with numerical stability.

Furthermore, multi-head attention is introduced allowing each attention head to attend to a different representation subspace:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O \\ \text{where head}_\text{i} &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.37}$$

Multi-Head attention is shown in Figure 2.11.

Positional encoding. The transformer model has no way of knowing the positions of the tokens (words) because, unlike in RNNs, recurrence is absent. To this end, the authors of the paper opted for adding a positional encoding to the embeddings based on sine and cosine functions:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_\text{model}}\right) \tag{2.38}$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_\text{model}}\right) \tag{2.39}$$

where pos is the position, i is the dimension and d_model is the embeddings dimension.

Encoder-decoder architecture. The encoder consists of $N = 6$ layers. In every encoder layer, the input is passed by a multi-head self-attention layer and by position-wise feed-

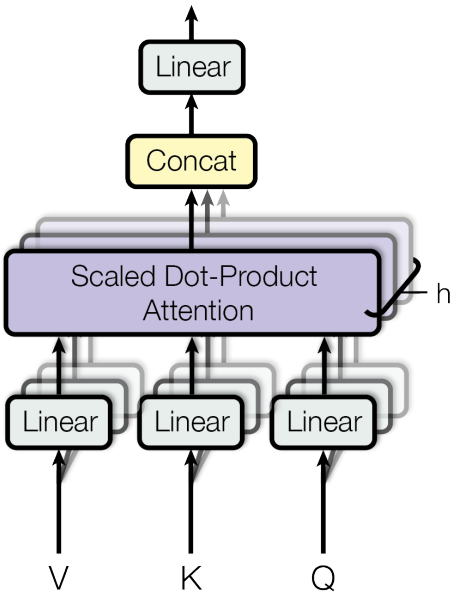


Figure 2.11: *Multi-head attention. Source: [7]*

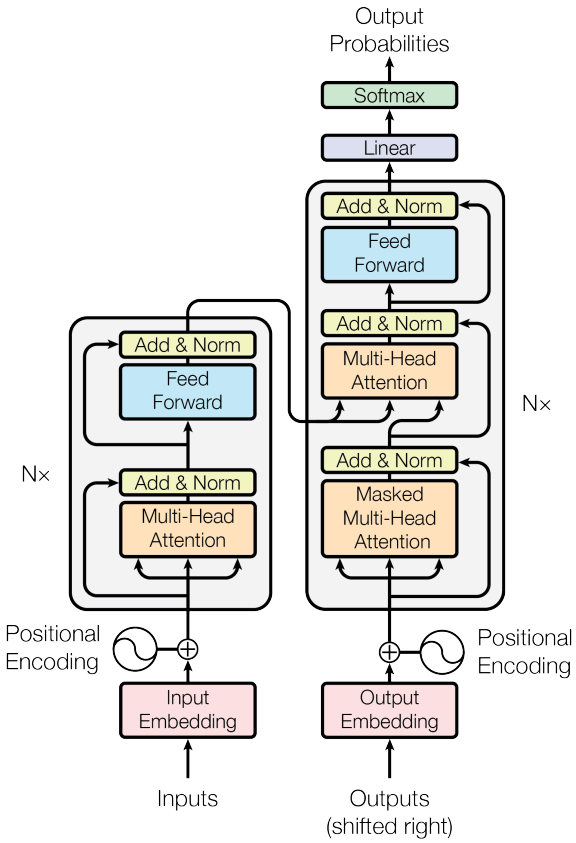


Figure 2.12: *The encoder-decoder transformer architecture. Source: [7]*

forward layers. The decoder also consists of $N = 6$ layers. The difference here is that before the feedforward layer there is an additional multi-head cross-attention layer that

enables the output to attend to the encoded input. The overall architecture is illustrated in Figure 2.12.

2.5 Transfer Learning

Transfer learning (TL) is a learning paradigm which enables a model to use the knowledge acquired during training for one task to solve another task that is related but different. For example, we could use the knowledge from a model that has been trained to classify images as dogs or cats to classify other animals.

We will first introduce some concepts and then formally define TL. The definition follows [40].

- The **feature space** $\mathcal{X}$ contains all possible input feature vectors.
- The **label space** $\mathcal{Y}$ contains all possible output label vectors.
- A **domain** $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is a tuple consisting of the feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over the feature space (only from specific samples $x_i \in \mathcal{X}$)
- A **task** $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$ is a tuple consisting of the label space $\mathcal{Y}$ and the conditional probability distribution $P(Y|X)$ which has to be learned.

The goal of TL is to learn $P(Y_T|X_T)$ in $\mathcal{D}_T$ by leveraging information learned from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$ (either the source domain or the source task is different from the target domain/task).

2.6 Multi-Task Learning

Apart from transfer learning, another learning paradigm that is widely used is multi-task learning (MTL) [41]. In multi-task learning, we train a model to perform multiple tasks at once. This is typically managed by optimizing two or more loss functions. The loss functions may correspond to either tasks that we want to solve or auxiliary tasks that help improve performance on the main task(s).

MTL motivates the model to exploit commonalities between the different tasks. It introduces an inductive bias as the model learns to bias representations that are useful for performing multiple tasks. As such, it acts as a form of regularization preventing overfitting.

MTL is implemented in deep learning with either hard or soft parameter sharing:

- In **hard parameter sharing** the hidden layers are shared and each task has its own task-specific output layers.
- In **soft parameter sharing** the tasks have separate layers and they are trained to keep their parameters similar.

Chapter 3

Natural Language Processing

Natural Language Processing (NLP) studies computational methods that process and analyze natural language data. It is a field in the intersection of linguistics, computer science and artificial intelligence. The term computational linguistics is often used as a synonym to NLP as it extends linguistics to take advantage of computational approaches.

Although humans communicate with natural language every day, it is not trivial for a computer to be programmed to understand language as deeply as humans do. Natural language has rules that are too high level and abstract which makes writing computer programs that are capable of understanding them difficult. Examples of characteristics that indicate the challenging nature of natural language include ambiguities, words of different meanings (homonyms) and coreference. NLP is therefore an interesting and challenging research area.

3.1 NLP tasks

Typical applications of NLP include [14]:

- **Part-of-speech tagging (POS tagging)**: It identifies whether each word in a sentence is a noun, verb, adjective, adverb, etc.
- **Named entity recognition (NER)**: It identifies and classifies named entities in a sentence. Named entities may include persons, cities, countries, time/date expressions etc.
- **Machine translation (MT)**: Translation between two natural languages e.g. from English to Chinese.
- **Constituency parsing**: It includes assigning a structure to a sentence called constituency-based parse tree.
- **Dependency parsing**: A dependency-based parse tree is assigned to the sentence which differs from constituency-based parse tree because it only displays relationships between words.
- **Coreference resolution**: It identifies referring expressions and the referent (the entity which they refer to).
- **Question answering**: It involves answering questions based on information or knowledge.

- **Dialogue systems**: Systems or agents which have conversations with a user. Such conversational agents are mainly divided to task-oriented dialogue agents which aim to assist the user to accomplish tasks and open-domain dialogue agents or chatbots which mostly aim to mimic humans trying to entertain the user with unstructured conversations. Chapter 4 comprehensively studies dialogue systems.

- **Automatic speech recognition (ASR)**: Speech-to-text (STT) translation.

- **Speech synthesis**: Text-to-speech (TTS) translation.

3.2 N-gram Language Models

Language models (LM) assign probabilities to sequences of words and can predict the next word based on the preceding words. Problems like speech recognition, machine translation and POS tagging can benefit from language modeling. In this section we will introduce a simple way to perform language modeling: n-gram language models.

The task of language modeling is to find the probability:

$$P(w_{1:m}) = P(w_1, w_2, \ldots, w_m) \tag{3.1}$$

for a sequence with m words. By applying the chain rule of probability to Equation 3.1:

$$P(w_{1:m}) = P(w_1)P(w_2|w_1)\ldots P(w_m|w_{1:m-1}) = \prod_{k=1}^{m} P(w_k \mid w_{1:k-1}) \tag{3.2}$$

N-grams approximate the conditional probabilities by making the assumption that the next word only depends on the $n-1$ previous words (Markov property):

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1}) \tag{3.3}$$

Then the conditional probabilities can be computed using the frequencies of the n-grams in the dataset.

3.3 Distributional hypothesis - word embeddings

In linguistics, the distributional hypothesis [42] refers to the property that words with similar meanings usually appear in similar contexts. Firth [43] summarized and popularized this hypothesis by the famous "you shall know a word by the company it keeps". According to this hypothesis, the meaning of words can be learned from their distribution in texts. Words can be represented as vectors such that words with similar meanings have a small distance. Linear algebra can be used as a tool to perform tasks on these vectors. The advantage of models using distributional semantics is that they replace manual feature engineering and they instead extract representations directly from text.

3.3.1 Tf-idf

Term frequency-inverse document frequency (tf-idf) is a statistic which indicates the importance of a word in a document. Tf-idf is a product of two statistics: term frequency and inverse document frequency.

Term frequency:

$$\mathrm{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3.4}$$

where $f_{t,d}$ is the number of appearances of the word t in the document d. Term frequency is normalized by the total number of words in the document.

Inverse document frequency:

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{3.5}$$

where D is a collection of documents and $N = |D|$. The denominator is the number of documents where the word t appears at least once. This statistic expresses how rare the word is across the documents. Rare words have a larger inverse document frequency as they are more "important" than common words like "the" or "and".

Term frequency–inverse document frequency (tf-idf):

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D) \tag{3.6}$$

Words that have high term frequency and low document frequency have high tf-idf. These words appear multiple times in a document but are not common in other documents.

3.3.2 Co-occurence matrix

The co-occurrence matrix is constructed by measuring how often words occur together. Two popular choices are the term-document matrix and the term-term matrix. They can be used to derive sparse vectors for each word or document. Tf-idf is one way to weigh the words in a co-occurrence matrix and thus improve the discriminating ability of the vectors. Nevertheless, such methods produce vectors that are high-dimensional (typically $|V|$-dimensional where V is the set of all possible words). Sparse vectors are not perfect representations when used e.g. by neural networks because learning tends to be difficult.

3.3.3 Word2vec

From the previous section it becomes clear that lower-dimensional vectors with dimensions that better capture linguistic properties are a better solution for the representation of words. These dense vectors which are usually called **embeddings** can be learned from large datasets and then used for other tasks making them an example of transfer learning. Embeddings can be classified as **static embeddings** like word2vec [8] or GloVe [44] and **contextualized embeddings** like ELMo [45] or BERT [17]. Static embeddings represent words as fixed vectors while contextualized embeddings are dynamic and depend on the

context. In this section we will introduce the word2vec algorithm.

The goal of word2vec is to learn high-quality dense distributed representations from huge datasets and large vocabularies in a computationally efficient way. The authors of the word2vec paper observed that previous architectures used a neural network with a non-linear hidden layer which made training on large datasets impossible. Instead, they choose a shallow network with a projection layer that projects the sparse input vectors to vectors with a smaller dimension and an output layer. The training objective is to reconstruct sentences based on the context and two architectures to achieve that were proposed: continuous bag-of-words (CBOW) and continuous skip-gram (Figure 3.1).

In the CBOW architecture, the model predicts the word in the middle based on the previous and next words. Every word is projected with the projection layer (the same weights are used for all words) and then the vectors are averaged. This means that the order of the words is irrelevant (bag-of-words).

In the continuous skip-gram architecture, the model predicts words found in the context based on the word in the middle. More distant words are sampled less because they tend to become less related to the middle word.

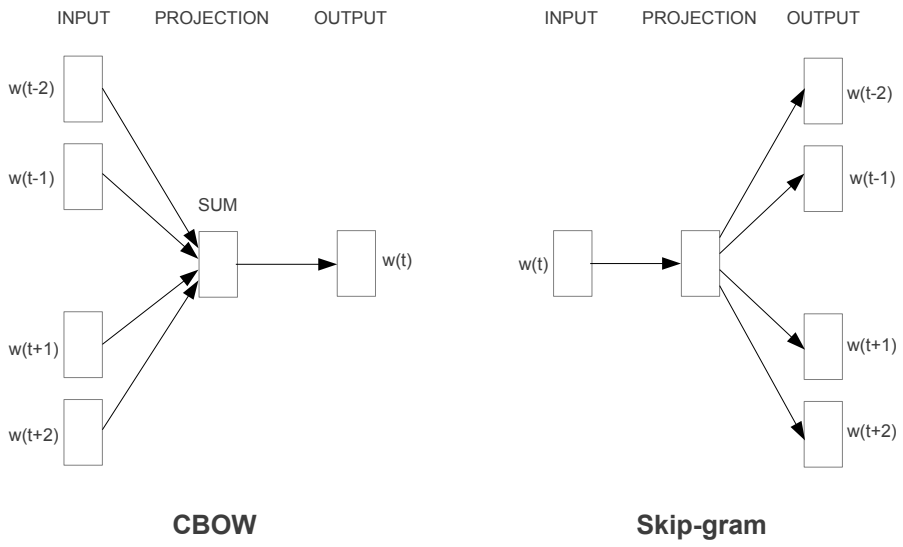


Figure 3.1: *The two word2vec model architectures. Source: [8]*

Syntactic and semantic word similarities are preserved by the embeddings. It is interesting that even simple metrics such as cosine similarity and vector arithmetics can be directly applied. For example, the vector `man-woman+queen` is close to the vector `king`. More examples of related words are shown in Figure 3.2. Although the reasons of the effectiveness of word2vec and related algorithms are not clear, a possible explanation is that the training objective causes the projected word representations of similar words to be close, confirming the distributional hypothesis. Embeddings derived from algorithms like word2vec are usually the default choice for encoding words in sequence architectures like those described in the previous chapter.

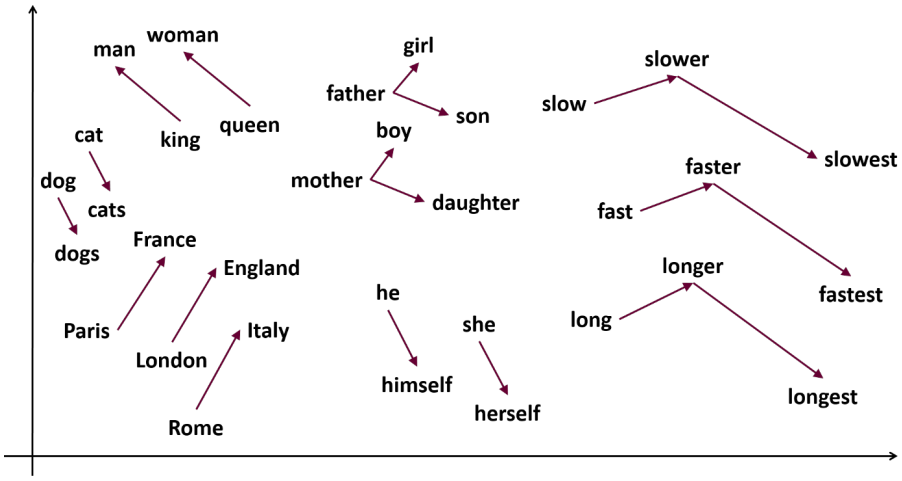


Figure 3.2: *Visualization of embeddings that capture the similarities between words. Source:* https://medium.com/@nuripurswani/word2vec-for-talent-acquisition-ab20a23e01d8

3.4 BERT

3.4.1 Architecture

BERT [9] is a multi-layer bidirectional transformer encoder. The authors of the paper report results for two BERT variants with different model sizes: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. $\text{BERT}_{\text{BASE}}$ has $L = 12$ transformer encoder blocks, the hidden size (size of the feedforward networks) is $H = 768$, it has $A = 12$ self-attention heads and a total of 110M parameters. For the larger version $\text{BERT}_{\text{LARGE}}$ the respective parameters are $L = 24$, $H = 1024$, $A = 16$ and a total of 340M parameters.

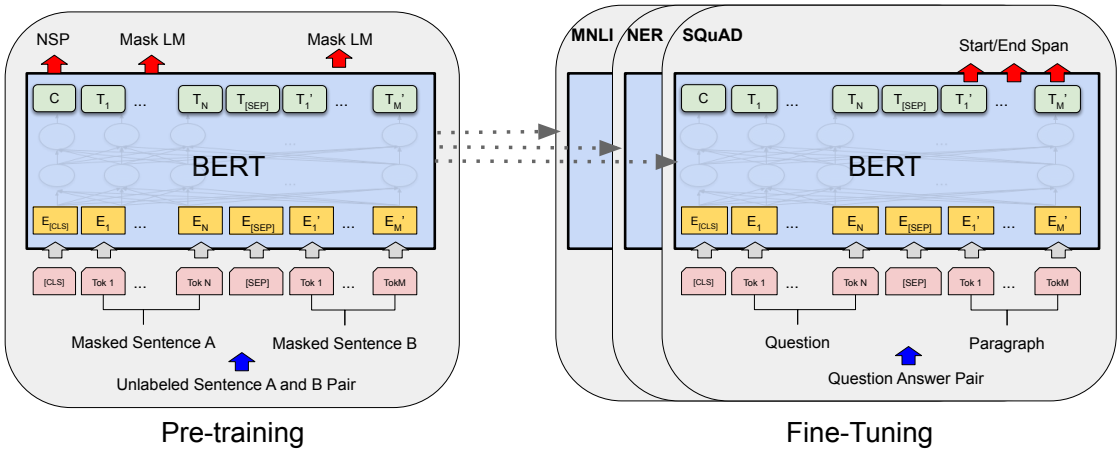


Figure 3.3: *BERT is first pre-trained on large corpora and then fine-tuned on task-specific datasets. Source: [9]*

3.4.2 Inputs and outputs

The input is a sequence which may be one or two sentences depending on the task. The input tokens are represented as embeddings (WordPiece embeddings are used). To

perform tasks such as classification two special tokens are introduced: `[CLS]` and `[SEP]`. `[CLS]` is always the first token of the sequence. For sequences consisting of two sentences, they are separated with the `[SEP]` token and additional learnable segment embeddings are added to the token and position embeddings.

Each token is encoded with BERT to a representation (embedding) of size H (768 for $\text{BERT}_{\text{BASE}}$). The special `[CLS]` token is used as a representation for the entire sequence and is useful in sequence classification tasks. BERT embeddings are contextual meaning that the same word will have different embeddings based on its context (the sentence that is found in).

3.4.3 Pre-training

BERT is pre-trained on two large datasets: BookCorpus (800M words) and the English Wikipedia (2,500M words). BERT's pre-training includes two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Pre-training requires a lot of computational power and time. After pre-training, the model can be fine-tuned (further trained) on downstream supervised tasks like question answering.

During MLM, 15% of the input tokens are masked and the model's task is to predict them. The tokens which are selected to be masked are:

- Replaced with the `[MASK]` token 80% of the time
- Replaced with a random token 10% of the time
- Left unchanged 10% of the time

NSP is used to teach the model to understand the relationship between two sentences. The input is constructed by choosing two sentences such that the second one is the actual next sentence 50% of the time and a random sentence 50% of the time. BERT's task is to predict whether the second sentence is actually the next one.

3.4.4 Fine-tuning

Pre-trained BERT models can be used for downstream tasks by adding classification heads. Classification heads are neural networks that take as input the BERT embeddings and output class probabilities. The weights of the BERT model can be fixed during downstream training and in this case BERT is used as a feature extractor. However, further training the entire BERT model along with the classification head (fine-tuning) usually leads to better performance. Figure 3.4 shows some possible ways to use BERT on specific tasks.

3.4.5 RoBERTa

RoBERTa [46] is a model that improves on the original BERT's hyperparameter and training choices. The main modifications introduced by RoBERTa are:

- It is pre-trained longer with a bigger batch size on a larger dataset.

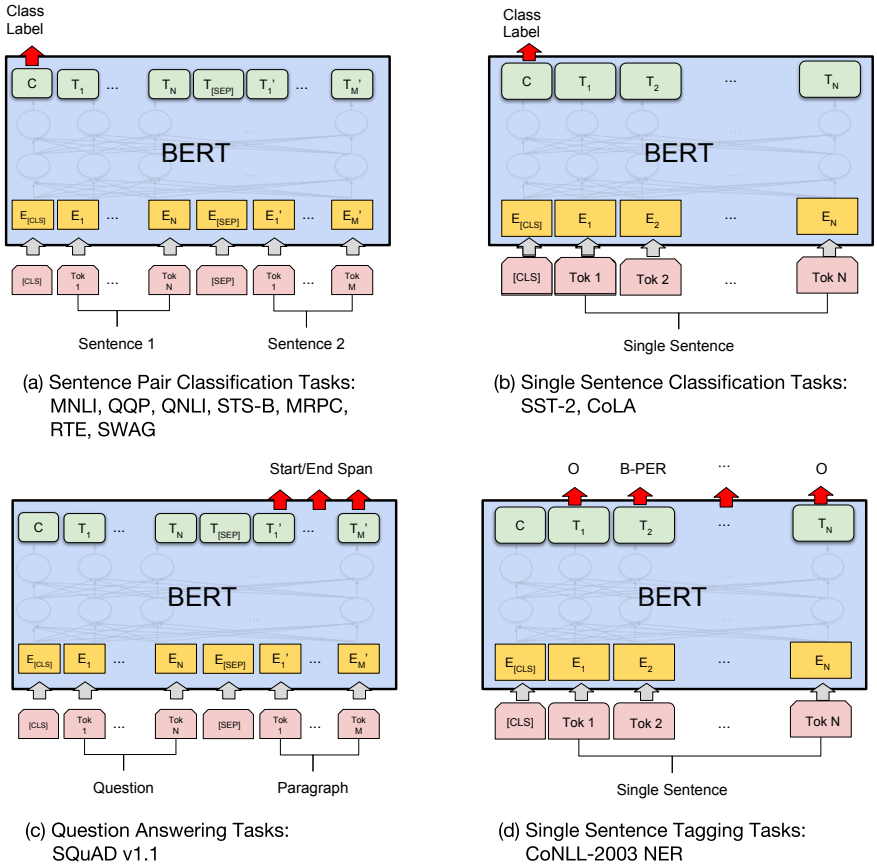


Figure 3.4: *BERT input sequences and classification heads for different tasks. Source: [9]*

- It does not use the next sentence prediction pre-training objective as the authors concluded that similar or even slightly better results can be achieved without it.
- The sequences used for pre-training are longer.
- The masking is dynamic: The input tokens that should be masked are chosen before feeding the input to the model rather than during the data preprocessing stage like in BERT. This means that in different epochs, the same sequence will be masked in a different way.

3.5 XLNet

BERT is trained with an autoencoding (AE) language modeling objective. This means that it does not rely on Equation 3.2 to explicitly model the probability distribution of the words in a sentence. Instead, BERT aims to reconstruct a corrupted sentence and thus it is possible to leverage bidirectional context. The disadvantage of this approach is that the words that have to be predicted are assumed to be independent.

On the other hand, autoregressive (AR) language modeling directly estimates the probability distribution of Equation 3.2. The conditional probability can be estimated with either a forward or a backward pass of the sentence. Although AR models can be used

for tasks like natural language generation, they fall short of performing natural language understanding when deep bidirectional context is required.

XLNet [10] is a transformer model that combines the benefits of both AE and AR transformer models. It is a generalized autoregressive method and performs AR language modeling with a newly introduced pre-training objective: Permutation Language Modeling (PLM). PLM allows the model to consider bidirectional contexts by permuting the word order in the sentence making it more suitable for natural language understanding tasks than traditional AR models. Furthermore, as it does not rely on data corruption (using `[MASK]` tokens), it avoids BERT's pretrain-finetune discrepancy and unlike BERT, words to be predicted are not assumed independent. XLNet borrows ideas from Transformer-XL [47] which improves on the vanilla transformer allowing better performance on long sequences.

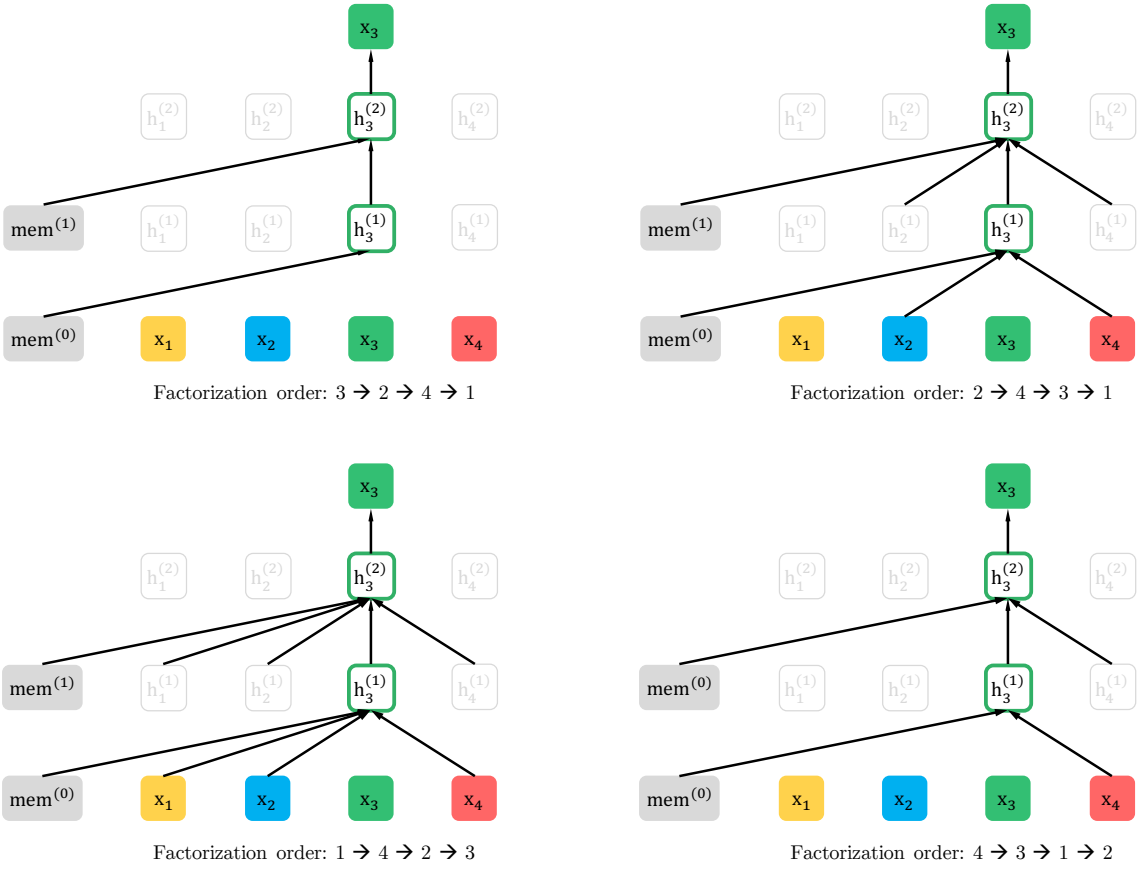


Figure 3.5: *Illustration of XLNet's permutation language modeling objective for predicting x_3 with different factorization orders. Source: [10]*

3.6 T5

Text-to-Text Transfer Transformer (T5) [11] is a model that leverages a unified text to text formulation for all language problems. The authors of the paper introduced a new dataset, the Colossal Clean Crawled Corpus (C4), which is two orders of magnitude larger than Wikipedia, to accommodate the model's pre-training. C4 is a cleaned version of

Common Crawl, a large dataset of crawled websites [1].

T5 relies on natural language description of the downstream tasks instead of specialized classification heads as used by BERT (and related) models. The model is trained to generate the predictions as text. For example, for the translation task from English to German, the model may take as input the sequence "translate English to German: That is good." and should output "Das ist gut." More examples of downstream tasks are shown in Figure 3.6.

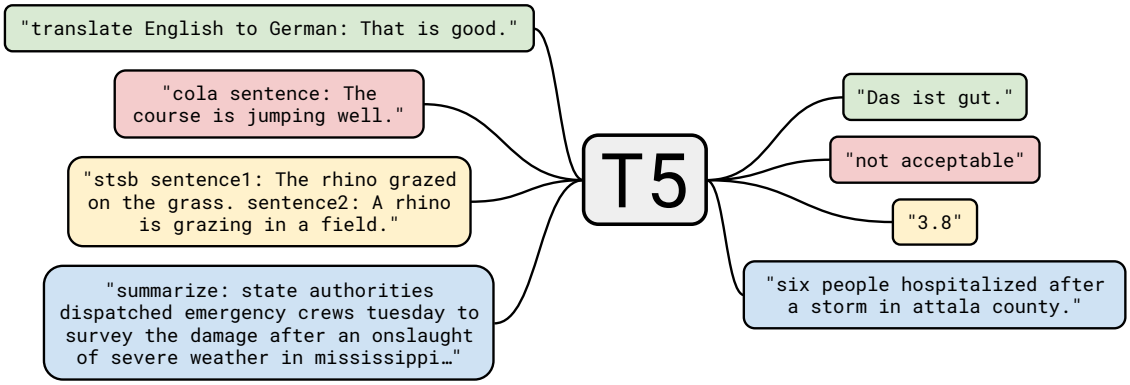


Figure 3.6: *T5 model on various downstream tasks. Source: [11]*

After a survey that compares many different approaches to transformer architectures and pre-training objectives, the authors found that the best results can be obtained with an architecture close to the vanilla encoder-decoder transformer and a denoising pre-training objective (similar to BERT). The pre-training and fine-tuning phases are shown in Figure 3.7.

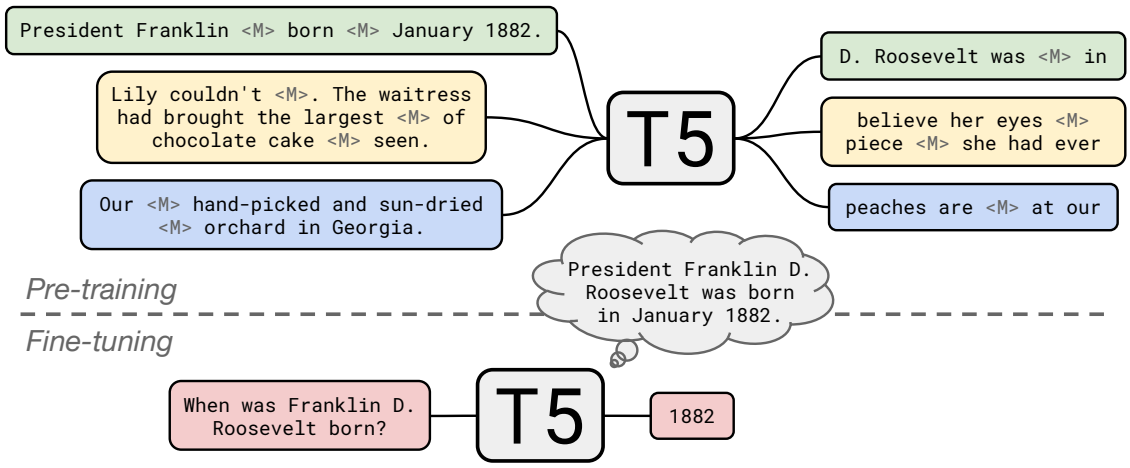


Figure 3.7: *T5 pre-training and fine-tuning. Source: [12]*

[1] https://commoncrawl.org/

Chapter 4

Dialogue Systems

4.1 Introduction to dialogue systems

Dialogue systems are designed to communicate with users in natural language. One of the longest running challenges in artificial intelligence is developing intelligent agents that can have conversations with the goal of entertaining humans, helping them accomplish daily tasks or both. Therefore, dialogue systems generally fall into two categories [14, 48, 49]. Systems that aim to not only have engaging and long conversations but also satisfy emotional needs of humans are called open-domain dialogue systems [13]. By contrast, task-oriented dialogue systems [15] aim to help users achieve more domain-specific tasks like booking flight tickets or finding a restaurant to eat. In this section we will introduce general methods and challenges in dialogue systems and in Section 4.2 we will discuss about a very important component that mainly task-oriented systems use: Dialogue State Tracking (DST).

4.1.1 Open-domain dialogue systems

Early dialogue systems include Eliza [50], Parry [51] and Alice [52]. Such systems are rule-based and work well on specific environments. Unlike traditional systems, most modern systems are data-driven i.e. trained with human-human conversations without relying on handcrafted rules. They are typically implemented with end-to-end architectures that directly produce the response at each turn based on the dialogue context. The response is produced by open-dialogue systems by either retrieval or generation methods.

Retrieval methods

Retrieval methods choose the best answer from a candidate set present in the corpus. In Figure 4.1 the architecture of a retrieval-based system is shown. Based on the dialogue context C and the input utterance X, retrieval algorithms are employed to retrieve the best candidates from a set of input-output pairs which are collected offline. Matching models consequently find the best output (next system utterance) Y by ranking the candidate responses. To this end, traditional information retrieval algorithms are effective [53] but more recently deep architectures are preferred. For example, BERT [17] can be used to separately encode the candidate responses and the current dialogue and the similarity between them can be calculated with the dot-product operation. Most sophisticated approaches (e.g. [54]) can act as an alternative to make the pairwise comparisons.

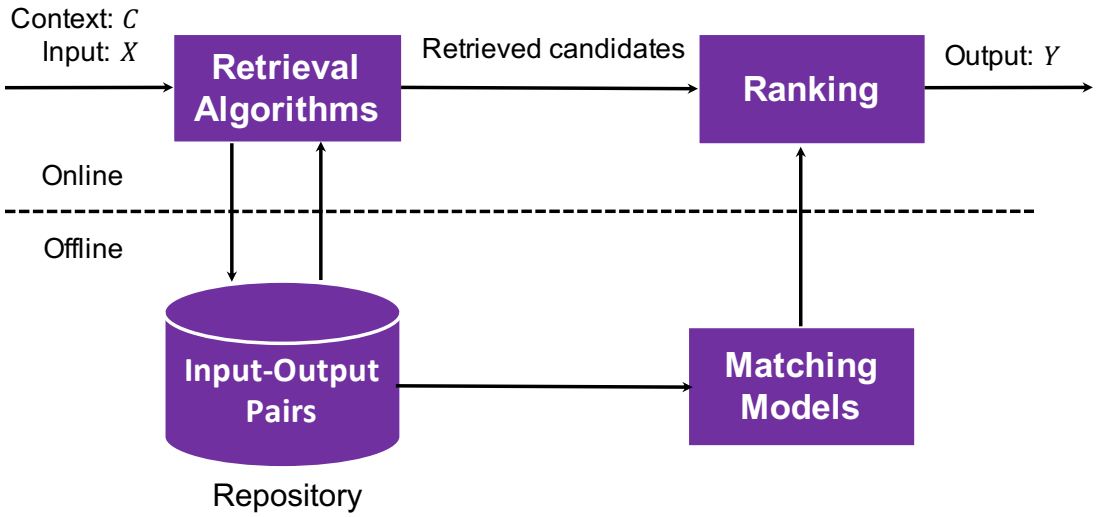


Figure 4.1: *Retrieval-based architecture. Source: [13]*

Generative methods

Alternatively, the problem can be formulated as a generation problem. In this case, seq2seq encoder-decoder models as discussed in Chapter 2 can be used (see Figure 4.2). Recently the best performing architectures are based on transformers [55, 56, 57]. Dialogue generation can also be performed with conditional variational autoencoders (CVAEs) [58] or generative adversarial networks (GANs) [59].

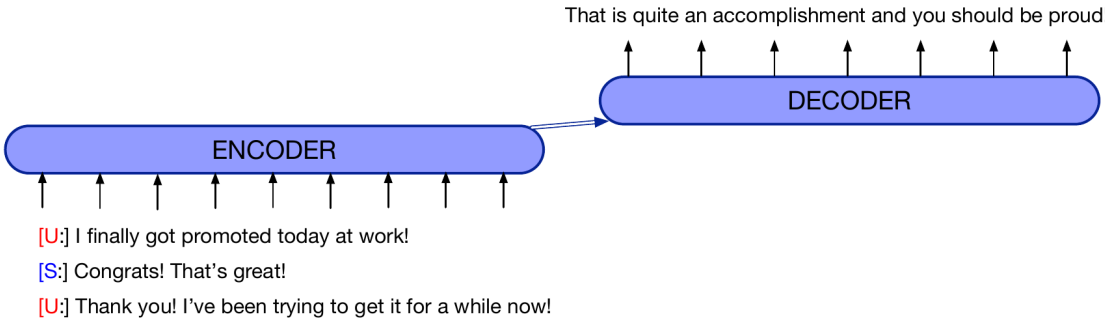


Figure 4.2: *Encoder-decoder based architecture. Source: [14]*

Hybrid methods

Retrieval-based methods produce responses with high quality as they are written by humans but the they are limited to the data present in the corpus. On the other hand, generation-based methods can produce unseen responses but are often not fluent or not related to the dialogue. A lot of hybrid methods are proposed to benefit from the strengths of both approaches. For example in [60] a candidate (prototype) response is retrieved and then edited to match the context. This method results in relevant and diverse responses.

Challenges

One of the most common problems in dialogue generation is bland responses. Systems based on seq2seq models prefer to respond with generic answers like "I don't know". The reason is that they are trained to maximize the conditional probability $P(S|T)$ of the response T based on the context S and short answers that do not convey much information can satisfy many possible contexts. Instead, [61] proposed to maximize the mutual information between T and S: $\frac{P(T,S)}{P(T)P(S)}$. Methods based on GANs [62] have also been used where the generator tries to produce an answer which the discriminator cannot tell whether it originates from a human or not. In addition, [63] proposed a model that mitigates the issue by introducing latent variables that capture high-level semantic information of the response.

Another issue is that dialogue generation models often produce inconsistent responses. For example, when asked "How old are you?", the system may respond with different answers across turns. To address this problem, we can encode personas that reflect the identity of the conversational agent [64] into representations called speaker embeddings. Speaker embeddings capture information about the speaker such as their accent or their preferred conversational topics. Datasets like PERSONA-CHAT [65] have helped research on agents with consistent personality.

In some cases, a desired property of conversational systems is to ground the dialogue in the real world. The dialogue can be ground in the personalities of the speaker [65], in visual information [66] or knowledge [67]. Grounded models can respond based on facts and may prove more useful in the real world. The implementation of such systems involves encoding the external real-world information along with the dialogue context.

Evaluation

The quality of an open-domain dialogue system can be evaluated either by humans or with automatic metrics. Humans can be asked to evaluate the engagingness, the repetitiveness, how much sense the answers make etc. Although human evaluation is the most accurate metric, it is expensive and time-consuming and therefore automatic metrics are often used instead. Word-overlap metrics like BLEU [68], ROUGE [69] and METEOR [70] calculate the similarity between the ground-truth and the generated sequences. More recently neural metrics [71] have been proposed to evaluate dialogue generation by learning to predict human-like scores.

4.1.2 Task-oriented dialogue systems

Task-oriented dialogue (TOD) systems are more goal-oriented and domain-dependent. The domain knowledge is often represented by a structured ontology and the dependence on such ontologies is a major difference between task-oriented and open-domain dialogue systems. TOD methods can be classified as **pipeline** and **end-to-end**. In pipeline methods the system leverages several components (modules): **Natural Language Understanding (NLU), Dialogue State Tracking (DST), Dialogue Policy** and **Natural Language Gener-**

ation (NLG). A high-level illustration of the pipeline architecture is shown in 4.3. On the other hand, end-to-end methods use a single model similarly to open-domain dialogue systems. Pipeline methods are widely used because they are tend to be more accurate but end-to-end methods are easier to build because they require fewer annotations.

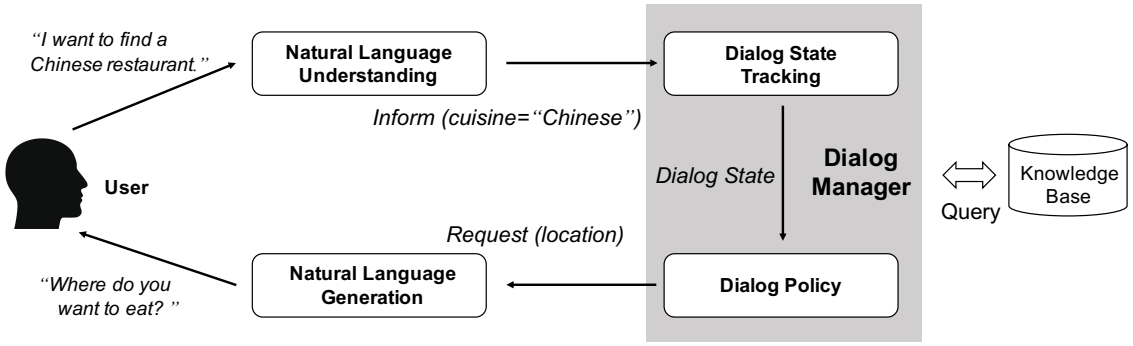


Figure 4.3: *A traditional pipeline TOD system. Source: [15]*

Natural Language Understanding

The first component of pipeline methods is NLU. Based on a single user utterance, NLU generates a structured representation which usually consists of intents and slots. **Intents** are functions requested by the user such as *ReserveRestaurant* and *TransferMoney* and **slots** are elements mentioned in the utterance such as *restaurant_name* and *account_type*. NLU thus involves two tasks: intent prediction and slot filling. Intent prediction finds the active intent in the user utterance from the list of intents present in the domain ontology. Slot filling finds the slot-value pairs by tagging the user utterance tokens: each token may be tagged as either part of one of the slots or not belonging to any slot. RNNs were the traditional choice for NLU [72, 73, 74] while more recently BERT is becoming more popular [75]. Although, intents and slots are the most common way to represent the ontology, other representations that capture more complex dialogues can be used as well.

Dialogue State Tracking

DST predicts the user goal for each dialogue turn given the entire dialogue. The user goal is captured by the **dialogue state** which can be seen as an abstracted representation of the dialogue. Early works make the assumption of discrete predefined dialogue states and model the task as a Markov Decision Process (MDP) [76]. Most recent works represent the dialogue state as slot-value pairs [77] and the values are found with classification. Although NLU used to be required in order to produce a semantic user utterance representation that the DST component could understand, nowadays a separate NLU component is rarely used. For a more comprehensive review of DST methods see Section 4.2.

Dialogue Policy

Given the dialogue state, the dialogue policy predicts the next **system actions**. System actions are semantic representations of the information that should be present in the system utterance e.g. `Request (location)` expresses that the system should request from the user their desired restaurant location. The framework of MDPs is suitable for the policy component and reinforcement learning (RL) is an ideal solution to the problem. Typically, it is first trained with supervised learning offline and it is then fine-tuned with RL. Note that unlike other components which are trained individually, RL fine-tuning requires the entire system. Model-free RL frameworks [78] rely on the interaction with human users. Alternatively, user simulators can replace real users [79] or model-based RL algorithms [80] can integrate planning and alleviate the need for interactive training with humans.

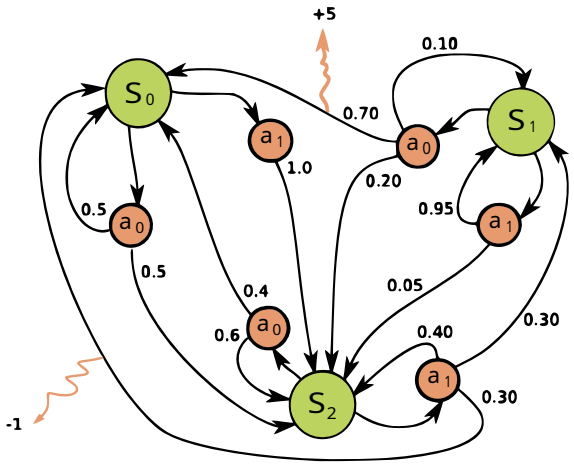


Figure 4.4: *An example of a MDP. Green circles correspond to states, orange circles correspond to actions and orange arrows to rewards. Source:* https://en.wikipedia.org/wiki/Markov_decision_process#/media/File:Markov_Decision_Process.svg

Natural Language Generation

The last component of TOD systems, NLG, generates the system utterance based on the predicted system actions. This can be seen as a seq2seq task. Semantically Conditioned LSTMs (SC-LSTMs) [81] are LSTMs that take into account dialogue act information to produce better responses. SC-GPT [82] pre-trains GPT-2 [83] on a large NLG dataset and fine-tunes on task-oriented generation tasks with few samples (few-shot learning) achieving better performance.

End-to-end TOD systems

Despite the widespread use of pipeline methods and their proven stability, they have several drawbacks including a complex system design, the requirement for more annotated

data and the need for individual training of components which may lead to error accumulation. Inspired by work in open-domain dialogues, end-to-end methods reformulate TOD as a seq2seq task and directly generate the next system utterance based on the dialogue history. [84] proposed a network that consists of several components but is end-to-end differentiable and can therefore be trained as a single model. [85] used end-to-end memory networks to reason over multiple turns. Sequicity [86] uses a two-stage seq2seq model to first generate the dialogue state and then the final system utterance. Furthermore, SimpleTOD [87] keeps the component-based approach and employs transfer learning from large open domain datasets by using a pre-trained model such as GPT-2. The pre-trained model solves all subtasks in a unified seq2seq approach.

Evaluation

Like in open-domain systems human evaluation is the most accurate metric for the quality of the system. The evaluation metrics measure whether the task was completed successfully, the user satisfaction, the number of required turns etc. However, in practice dialogue systems are more easily evaluated with automatic metrics. For NLU and DST, slot and intent classification metrics are used e.g. intent accuracy and slot F1. Dialogue policy uses inform rate, match rate and task success rate [15]. For NLG similar metrics as in open-domain dialogues apply e.g. BLEU. These metrics are well-defined but they do not necessarily represent the quality of the system in the real world. As a middle ground, user simulation can be used to automatically evaluate the system although the research on the field is still ongoing.

4.2 Dialogue state tracking

Dialogue State Tracking (DST) is an essential component of task-oriented dialogue systems. As previously described, it tracks the user goals over multiple turns of dialogue. The dialogue state is then used by dialogue policy to predict the next system actions and/or issue queries to knowledge bases. In most modern systems, DST takes the natural language utterances directly as input without the need for intermediate NLU representations for user utterances. However, many works leverage the produced system actions instead of system utterances for past system turns [77, 88]. Being the first component of such systems makes DST's accuracy critical as with wrong dialogue state predictions next components cannot function properly. In this section we will review recent developments in the topic of DST modeling. For this section the goal of DST is to estimate the dialogue state which consists of slot-value pairs unless mentioned otherwise.

4.2.1 Datasets

DST datasets require dialogues with annotations on the dialogue state level per user turn. Some datasets contain single-domain dialogues such as DSTC [89], DSTC2 [90], CamRest [91] and WOZ [77]. However, most recent datasets span over multiple domains and provide a significantly more challenging testbed for DST systems. In multi-domain

datasets, the user may interact with the system over more than one domains. For example, a user may ask the system to search for a flight to Athens and then request attractions in the area. Therefore, slots may take values from previous domains especially in turns where domains are switched. Popular multi-domain datasets include MultiWOZ 2.0-2.4 [92, 16, 93, 94, 95] and the Schema-Guided Dialogue (SGD) dataset [1].

Agent: I have two restaurants. They are Pizza Hut Cherry Hinton and Restaurant Alimentum. User: What type of food do each of them serve? **restaurant.name**: *Pizza Hut Cherry Hinton, Restaurant Alimentum*
User: I would like to visit a museum or a nice nightclub in the north. **attraction.type**: *museum, nightclub*
User: I would also like a reservation at a Jamaican restaurant in that area for seven people at 12:45, if there is none Chinese would also be good. **restaurant.food**: *Jamaican (preferred), Chinese*
User: I would prefer one in the cheap range, a moderately priced one is fine if a cheap one isn't there. **restaurant.pricerange**: *cheap (preferred), moderate*

Figure 4.5: *An example dialogue with dialogue state annotations from MultiWOZ 2.1. The slots span two domains: restaurant and attraction. Source: [16]*

4.2.2 Discriminative and generative DST

Early methods [77, 96] use classification to find the values for slots assuming that every slot has a predefined list of possible values. NBT [77] is a system that relies on natural language utterances instead of the output of a separate NLU component. However, discriminative approaches fail to scale to slots with many possible values. For example, it is impossible to list the candidate values for a slot like *restaurant_name* because they are too many and constantly changing. Therefore, during inference possibly unknown slot values may appear and the system is not capable of handling them.

The Global-Locally Self-Attentive Dialogue State Tracker (GLAD) [88] uses global modules with shared parameters across slots as well as local modules for each slot. This approach dramatically outperforms prior methods because it generalizes better to rare slot-value pairs. Other works completely remove the need for slot-specific parameters and are thus scalable to large sets of slots [97, 98]. However, such models can still not handle slot-value pairs that are not present in the training dataset.

Sequicity [86] formulates slot filling as a two-stage sequence generation task taking advantage of simple seq2seq architectures. In the first stage it generates the dialogue state (belief state) and in the second stage the final system response. To handle out of vocabulary (OOV) words it utilizes an extension to the traditional seq2seq framework, CopyNet [99], which allows copying of such OOV words. Another approach [100] is to use the pointer network (PtrNet [101]). TRAnsferable Dialogue statE generator (TRADE) [102] uses a state generator with an utterance copying mechanism to decode the values for each slot separately achieving great zero-shot and few-shot performance as slots are independent from the model design. COMER [103] decreases the computational complexity by hierarchically generating the dialogue state: first the slots are decided and then for each slot in the dialogue state its value is decoded. TripPy [104] uses three copy mechanisms to retrieve the value for a slot: 1) span extraction from the user utterance 2) copying from system inform memory 3) copying from a different slot from the dialogue state.

Hybrid approaches [105] can select whether to use classification or generation to find the value for the slots combining the benefits of both strategies. This is very practical in real-world dialogue systems because there are slots which can better be modeled as categorical slots. For example, boolean slots like *has_vegetarian_options* in the restaurant domain has only two possible values: *true* and *false*. Other types of categorical slots may include numerical slots (where the number of possible values is limited) or other slots with a small number of possible values like *price_range* which can be *cheap*, *moderate* and *expensive*. The SGD dataset and recent MultiWOZ versions divide slots to categorical and non-categorical and only provide possible values for categorical slots.

Some of the methods proposed until now rely on BERT to generate the word embeddings and therefore capitalize on transfer learning from huge datasets. Text-to-text transformers like GPT-2 [83], T5 [11], BART [106] etc can also be used to build an end-to-end generative dialogue state tracker without RNNs. SimpleTOD [87] treats all task-oriented components as seq2seq tasks and employs a unified approach with a single causal language model like GPT-2. MinTL [107] introduces a novel approach called Levenshtein belief spans which model the differences between old and new dialogue states and apply pre-trained plug-and-play sequence generation models like T5 and BART. UBAR [108] fine-tunes GPT-2 with input sequence all dialogue information: utterances, belief states, database results and system acts. In this way, it jointly performs DST and response generation. PPTOD [109] introduces a multi-task pre-training dialogue objective to plug-and-play models. A prompt that contains natural language instructions for each TOD task is used and allows to use a single model for all tasks.

Most recently, it has been found that the masked span prediction pre-training objective performs better than autoregressive language modeling and that pre-training with seemingly irrelevant tasks like text summarization can help [110]. Pre-trained transformer language models have become dominant in DST modeling because of the transfer learning capabilities and can be used for slot filling to find slot values with span extraction, classification, generation or any combination of the above.

4.2.3 DST as machine reading comprehension

Machine reading comprehension (MRC) is the task of automatically answering a question based on a text passage [111]. The availability of large scale datasets and well performing methods for MRC motivated us to study ways of applying transfer learning from MRC to DST. DST can be seen as an MRC task if we consider the dialogue context as the passage and "What is the value for slot x?" as the question [112]. It is also possible to manually create the questions for each slot to better represent its meaning [113] e.g. for the slot *food* of the *restaurant* domain, the question could be "What type of food does the user want to eat?". By encoding with BERT the pair of two sequences: the dialogue and the question and then finding the answer (slot value) [113] achieves great zero-shot and few-shot performance when pre-trained on MRC datasets. For categorical slots the question (slot) and each one of the possible answers (values) can be encoded separately and the value can be found by classification with the softmax function. On the other hand, non-categorical slots can be found via span extraction.

4.2.4 Schema integration in DST

Nowadays, commercial task-oriented systems use an ever-increasing number of diverse services, i.e. interfaces to dialogue domains. This motivated the development of the schema-guided paradigm [1], in which each service is defined by a structured ontology called **schema**. The schema usually contains the list of the supported intents and slots as well as natural language description for the various schema elements. For example, in the SGD dataset the schema for the service *Restaurants_1* has a slot with name *party_size* and description "Party size for a reservation". Many recently published datasets follow this paradigm including: SGD [1], MultiWOZ 2.1 [16] and STAR [114].

An important goal of schema-based approaches is scalability and generalization, i.e., to build systems that are capable of handling completely new domains and services. To decouple the schema from the architecture, the schema element names and descriptions should be used as an additional input to DST models. The models can therefore generalize to services that they have not seen before by relating semantically similar schema information from the training set. This enables the ability to scale to services with different schemata but similar functionality or even to completely new services of distant domains without the need for further training.

The challenge of zero-shot generalization makes pre-trained transformer models such as BERT, XLNet, GPT-2 and T5 the most popular choice for schema-guided DST. Models that adopt the MRC formulation for the problem as described in the previous section are generally capable of performing zero-shot generalization. Many models leverage the slot descriptions present in schema-guided datasets replacing the plain slot names [115, 116].

Schema-guided DST approaches fall into two categories:

- **Single-pass approaches**: For each dialogue turn the utterances are passed by the encoder only once. For example, this can be accomplished by pre-computing the schema embeddings with an encoder like BERT before training. The encoded

dialogue is then concatenated with the schema embeddings [1] or it attends to them [117, 118] to generate the DST predictions.

- **Multi-pass approaches**: For each dialogue turn the utterances are concatenated with each schema element description and are encoded separately. This allows deep self-attention between the schema and the dialogue and significantly improves performance on unseen domains [119]. However, it is less computationally efficient and it does not scale well to schemata with many slots.

A possible solution to the efficiency problem is to limit the number of encoded utterances for each turn making the input sequence smaller. However, there are cases (e.g. the slot value update is delayed) when the system should search the dialogue history to retrieve the slot value. DST systems can be designed to explicitly model **slot carryover** [120] as a binary decision to copy the value from slots mentioned previously in the dialogue. To this end, the system keeps track of slot-value pairs from previous dialogue states or system actions. In multi-domain dialogues the source slot may belong to a previous domain in the dialogue (cross-domain or cross-service slot carryover).

Various schema-based slot carryover mechanisms have been proposed. The SPPD system [20] predicts whether a slot carryover should take place and chooses the source slot from a candidate source set. The slots in this set are found with a separate BERT model that is trained to identify slots between which carryovers are made. SGP-DST [19] performs DST with a number of fine-tuned BERT models including one for in-domain slot carryover and one for cross-domain slot carryover with input the slots and pairs of slot-target slots respectively. Such methods make multi-pass approaches more computationally feasible but usually perform worse than models that encode the entire dialogue history and do not rely on slot carryover [22, 21].

Chapter 5

Multi-Task Schema-Guided Dialogue State Tracking

5.1 The Schema-Guided Dialogue Dataset

In this work we use the Schema-Guided Dialogue (SGD) dataset [1], a large scale multi-domain task-oriented dataset following the schema-guided paradigm. SGD contains 21,106 dialogues across 20 domains and 45 services exceeding other datasets in scale. The included annotations facilitate multiple task-oriented dialogue tasks such as NLU, DST and response generation. The dataset contains natural language descriptions for the various schema elements; an example schema is shown in Figure 5.1. To evaluate the zero-shot [1] generalization ability to unseen services and domains, in the standard test set 77% of the dialogue turns contain at least one service not present in the train set. In this way, building a unified model for all services and using the schema as input is encouraged.

In the Schema-Guided DST track of the 8th Dialogue System Technology Challenge, the participants developed zero-shot schema-guided DST models based on the dataset [18]. The DST's tasks are to predict the active user intent (**intent prediction**), the slots that are requested by the user (**requested slot prediction**) and the values for slots given by the user until the turn (**slot filling**). Therefore, we consider the dialogue state as the active intent, the requested slots and slot-value pairs for a turn although the first two tasks are more often considered NLU tasks. In multi-domain dialogues, a separate dialogue state is calculated for each service present in the dialogue.

As can be seen in Figure 5.1, there is a relationship between the supported intents and slots. Every intent lists a number of required slots and optional slots. We can categorize a slot as **informable** or **non-informable** depending on whether the user is allowed to give a value for it. In the next sections we assume that a slot is informable if it is either required or optional in at least one intent. We only consider informable slots as candidates for the slot filling task although any slot (informable or not) can be requested.

The evaluation metrics for the task of DST on the SGD dataset are the following according to [1]:

- **Active intent accuracy:** The percentage of user turns for which the correct active intent has been predicted.

[1] zero-shot learning refers to the problem in which during evaluation of a machine learning model there are data examples not observed in training

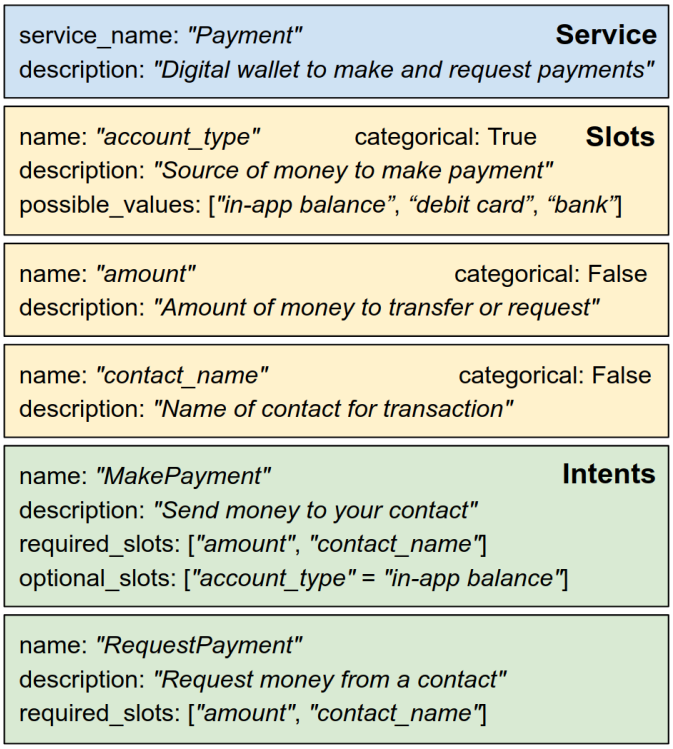


Figure 5.1: *An example schema for a Payment service. The schema contains a list of slots and intents. Slots are either categorical or non-categorical and a list of possible values is provided for categorical slots. Furthermore, each intent lists the required and optional slots that the user should provide when interacting with the particular intent. Source: [1]*

- **Requested slot F1:** The macro-averaged F1 score for requested slots over the user turns. Turns with no requested slot in ground truth and prediction are skipped.
- **Average Goal Accuracy:** The average accuracy of correctly predicting the slots over eligible turns. Slots with no assigned value in the ground truth dialogue state are skipped. For non-categorical slots, a fuzzy matching score is used to reward partial matches.
- **Joint Goal Accuracy:** The average accuracy of correctly predicting all slots for a turn. For non-categorical slots, a fuzzy matching score is used to reward partial matches.

5.2 Related work

The SGD-baseline [1] fine-tunes a BERT model on the last two utterances and concatenates the encoded tokens with schema embeddings - BERT-encoding of the schema elements. The schema embeddings are pre-computed before training. Then classification heads take as input the encoded dialogue utterances and different schema embeddings to output the class probabilities for each task.

Some of the proposed systems fine-tune a separate pre-trained model for each subtask with input the last two utterances and each schema element description (slot or intent)

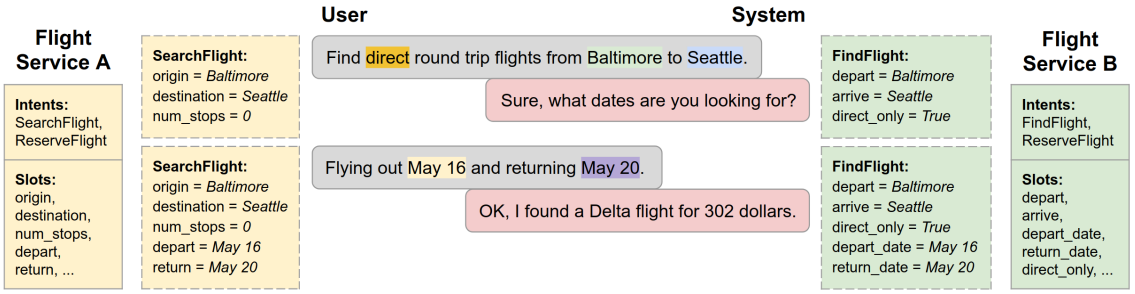


Figure 5.2: *A dialogue from the Flight domain. Services A and B can be used as an interface to the domain. Although they offer the same functionality, there are slight differences in their schema because they may come from different API designers. Source: [1]*

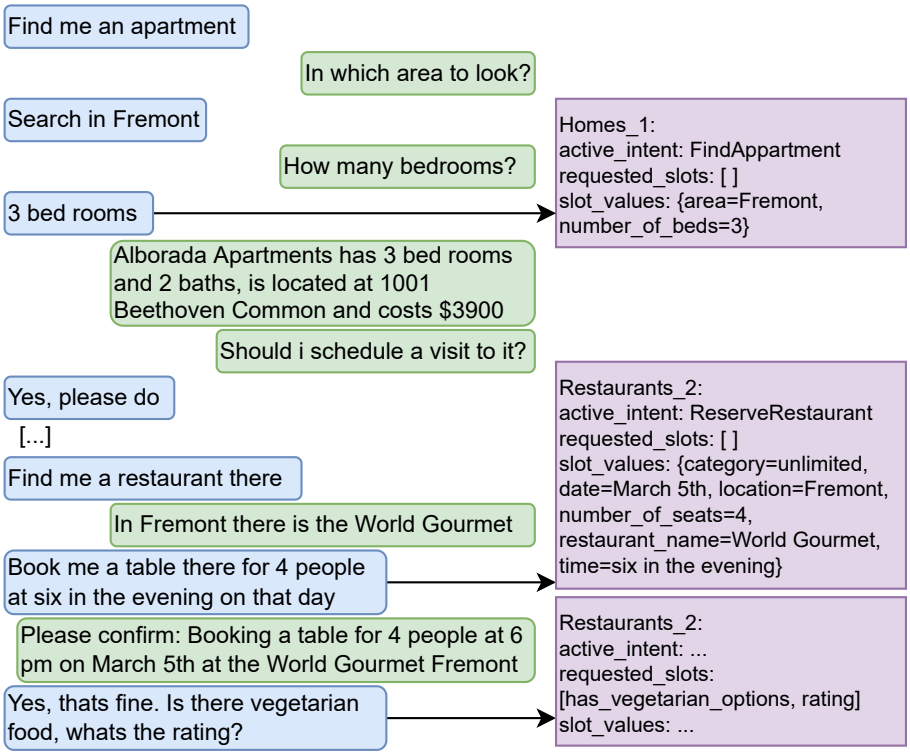


Figure 5.3: *Dialogue fragment with DST annotations for some user turns*

[19, 20]. Furthermore, to address the issue of long-range slot dependency (see Section 4.2.4) they use slot carryover mechanisms. Both SGP-DST [19] and the SPPD system [20] use a number of pre-trained BERT models and adopt a multi-pass approach needing to perform multiple BERT passes per dialogue turn.

State-of-the art methods [21, 22] do not rely on slot carryover mechanisms; they are trained with the entire dialogue history. The benefits of such approaches are that error accumulation is avoided and performance is better than slot carryover methods. paDST [21] fine-tunes a number of pre-trained models with the full dialogue and it is observed that a large number of handcrafted features and data augmentation can greatly improve performance on categorical slots. On the other hand, D3ST [22] fine-tunes a single T5 model for all subtasks with a single-pass approach i.e. all schema descriptions are concatenated and fed to the model along with the dialogue.

5.3 Baseline system 1: Multiple task-specific BERT modules and comparison of Encoder architectures

In this section, we introduce a baseline system with six NLU modules which perform predictions about each user turn. The tasks of intent prediction and requested slot prediction are solved with classification. For the slot filling task we perform classification (categorical slots), span extraction (non-categorical slots) and implement carryover mechanisms (see Section 4.2.4) to solve the long-range slot dependency problem. All the predictions are combined with a rule-based DST Inference module to produce the dialogue state for the specific turn. This approach is similar to [19] because we use a number of separately trained modules but the modules for slot filling are slightly different. We additionally compare two Encoder architectures: the Fusion Encoder and the Cross Encoder for encoding the dialogue and schema. The main goals of this section are to:

- Introduce a first baseline system and give examples of the tasks.
- Evaluate how the system performs when slot filling is seen as a two-stage prediction problem with separately trained modules and slot carryover mechanisms.
- Compare the two Encoder architectures.

5.3.1 Encoder architectures

From the dialogue we choose to encode only the last two utterances: the preceding system utterance and the current user utterance. For the schema we use the natural language description of each schema element present in the dataset. Both architectures are based on BERT and produce contextual representations for the dialogue and each schema element.

More specifically, the Encoder produces the u_{CLS} token acting as a representation for the entire sequence (utterances-schema element pair) and the encoded system and user utterance tokens u_{t1}-u_{t_N} which are used for span extraction (see Section 5.3.2). Each one of the six modules uses a separate Encoder and classification head(s). Some modules use additional binary features denoted by *bin* which are concatenated with u_{CLS} before feeding the classification head.

Fusion Encoder

The Fusion Encoder (Figure 5.4) encodes the utterances and the schema element descriptions separately with two BERT models. The Utterance BERT is fine-tuned while the Schema BERT is fixed. The encoded tokens from Utterance and Schema BERT are then projected to a lower dimension with the Utterance and Schema Projection respectively. The projected representations are given as input to a Transformer Encoder that is trained from scratch. This architecture is inspired by [119] but we also use the projections to make the training of the Transformer Encoder easier.

The advantage of the Fusion Encoder is that for each turn of the dialogue the expensive BERT pass is performed only once. The embeddings of the Schema BERT are pre-computed before training. The Transformer Encoder enables self-attention between the dialogue and the schema but it can use a smaller number of layers than BERT and its input is of lower dimension.

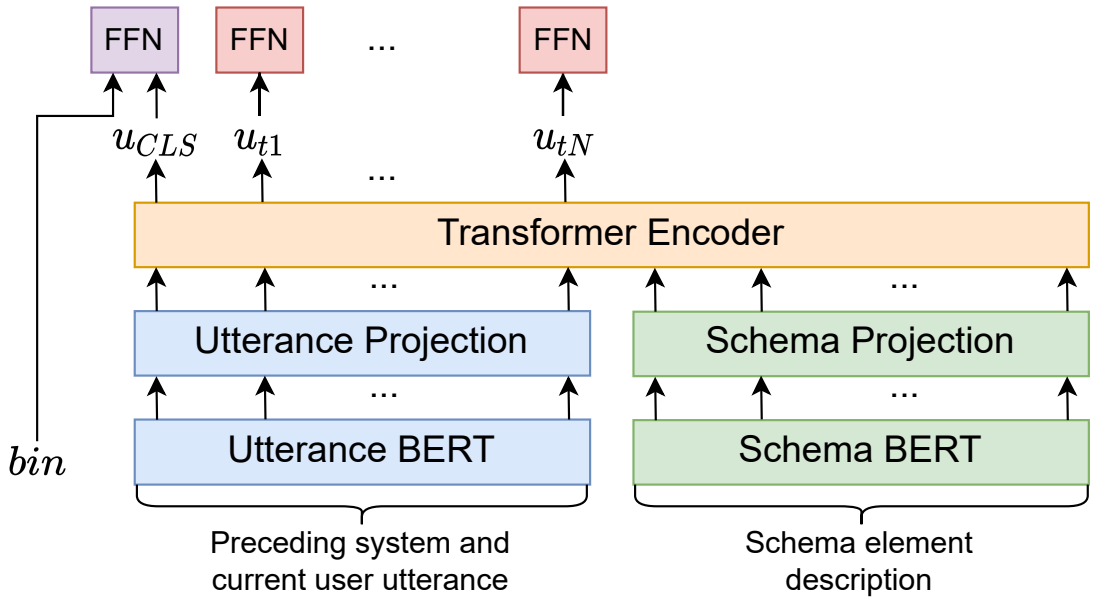


Figure 5.4: *Fusion Encoder*

Cross Encoder

The Cross Encoder (Figure 5.5) uses a single BERT model and takes as input the concatenation of the utterances and the schema element descriptions. The `[SEP]` token separates the utterances and the schema. This is a more common choice for the Encoder architecture.

The advantage of the Cross Encoder is that early deep self-attention between the utterances and the schema is enabled. Therefore, the final contextual embeddings are better conditioned on the schema than the ones from Fusion Encoder. However, it is now necessary to perform multiple BERT passes (as many as the schema elements) making this Encoder less computationally efficient.

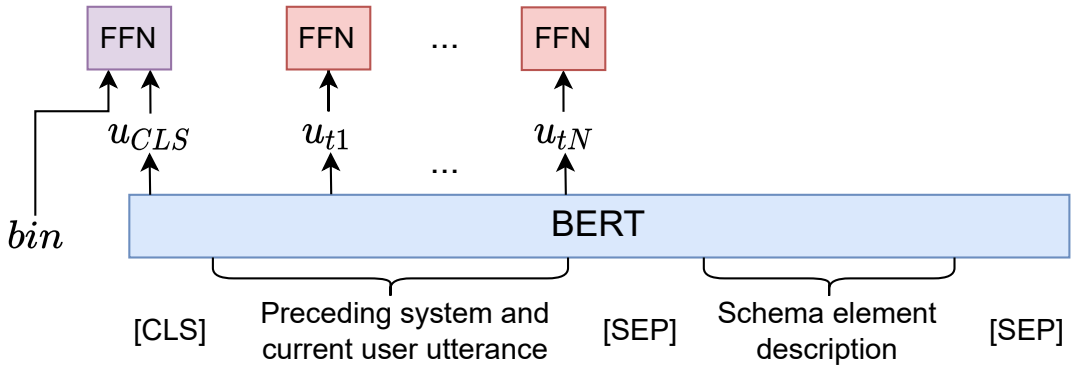


Figure 5.5: *Cross Encoder*

5.3.2 Modules

Intent module

The utterances and each one of the possible intent descriptions are encoded to classify whether the intent is active or not in the current turn. Because in some turns there is no active intent, we add an additional NONE intent with description "No active intent". We choose the intent with the highest probability as the active one.

This module uses one binary feature which indicates whether the intent was active in the previous turn. This helps the model in situations where only the last system and user utterances are not enough to predict the intent.

Figure 5.6: *Intent prediction. The active intent is "FindApartment" which has the following description: "Find an apartment in a city for a given number of bedrooms".*

Requested Slot module

The utterances and each one of the slot descriptions are encoded to predict whether the slot was requested by the user. The module predicts a probability for each slot and every slot with probability higher than 0.5 is considered requested.

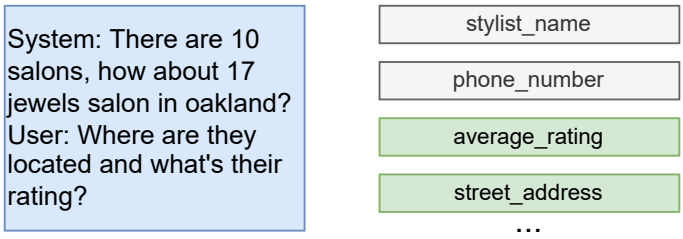


Figure 5.7: *Requested slot prediction. The user requests the value for two slots: "average_rating" and "street_address".*

Slot Filling Status module

The utterances and each one of the informable slot descriptions are encoded to predict the slot filling status of the slot. Based on the predicted status we decide whether we should update the slot value in the current turn and if so how. The possible values for the status are: 1) none 2) active 3) dontcare 4) in_service_carryover 5) cross_service_carryover. The exact slot value update mechanism is explained in Section 5.3.2.

For this module, we use three binary features: 1) whether the slot has a value in previous system history actions of the service, 2) whether the service that the slot belongs to has been switched to in the current turn and thus not found in the previous user turn, 3) whether the service is completely new in the dialogue. These binary features help the model decide when it should perform in_service_carryover or cross_service_carryover. They are similar to the ones proposed by [19].

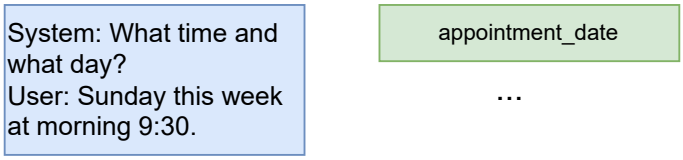


Figure 5.8: *The Slot Filling Status module identifies that in this particular example only one slot is "active" ("appointment_date"), all the other slots in the service take the "none" status.*

Non-categorical Slot module

The utterances and each one of the non-categorical informable slot descriptions are encoded to find the span in the user utterance corresponding to the slot value. For each token in the user utterance two probabilities are calculated: the probability that the token is the start of the span and the probability that it is the end of the span.

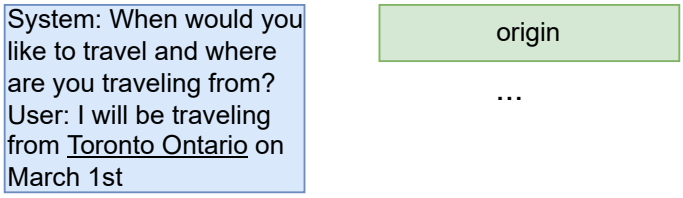


Figure 5.9: *Non-categorical slot filling. For the slot "origin" the slot filling status is "active": the user has given the value "Toronto Ontario".*

Categorical Slot module

The utterances and each possible pair of a categorical informable slot description and a candidate value for that slot are encoded to predict whether the value is given by the user. We then select the value with the highest probability.

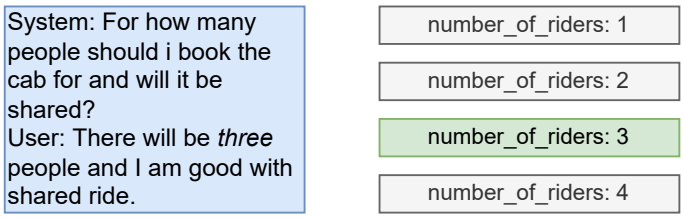


Figure 5.10: *Categorical slot filling. For the slot number_of_riders the slot filling status is active: the user has given the value 3.*

Cross-service Carryover module

The utterances and two slot descriptions are encoded to find the probability that a cross-service carryover is performed between the first slot (source slot) and the second slot (target slot). We use these probabilities to decide how to update the slot value as explained in greater detail in section 5.3.2.

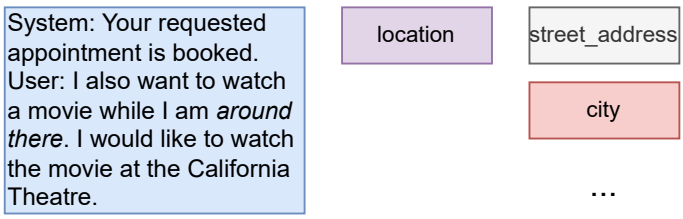


Figure 5.11: *In this example, the target slot "location" has the status of "cross_service_carryover". This means that the Cross-service Carryover module must find the appropriate source slot ("city"). The full example is illustrated in Figure 5.12.*

DST Inference module

The DST Inference module combines the results of the six aforementioned NLU modules and produces the dialogue state. For the tasks of **intent prediction** and **requested slot prediction**, the active intent and the requested slots are found with the Intent module and the Requested Slot module respectively whose functionality is straightforward.

For **slot filling**, the slot value update mechanism works in two stages. In the first stage the status of a slot is predicted (Slot Filling Status module). In the second stage, depending on the predicted status another module may be activated to find the corresponding value if necessary:

- If the status is predicted to be *none*, then the slot keeps the value of the previous dialogue state, if any, otherwise it is assumed to be not assigned.
- If the status is predicted to be *active*, then the slot is updated and the value is found in the current user turn. In this case, depending on the type of the slot we activate the appropriate module (Non-categorical Slot module or Categorical Slot module).
- If the status is predicted to be *dontcare* then the special value `dontcare` is assigned to the slot, indicating that the user does not have a preference.

- If the status is is predicted to be *in_service_carryover*, then the slot is updated but the user does not explicitly state the value of the slot in the current user utterance. The value is either found in the preceding system utterance or in earlier system turns of the current service. It can be extracted by the most recent system action that contains one single value for the specific slot. Therefore, the DST module has to find the most recent system action with the corresponding slot field matching the slot in question. Subsequently, the value from the selected system action is copied to the target slot.

- If the status is predicted to be *cross_service_carryover*, then the slot is also carried over from earlier utterances. However, in this case the source slot is different than the target slot and it comes from a different service than the current turn service. We activate the Cross-service Carryover module to find the source slot. The candidate source slots are all of the slots belonging to previous services that have appeared in the dialogue state or system actions before the current turn as long as the source and target slots have the same value type (e.g. both are numerical slots). Finally, the most recent value for the source slot is copied to the target slot.

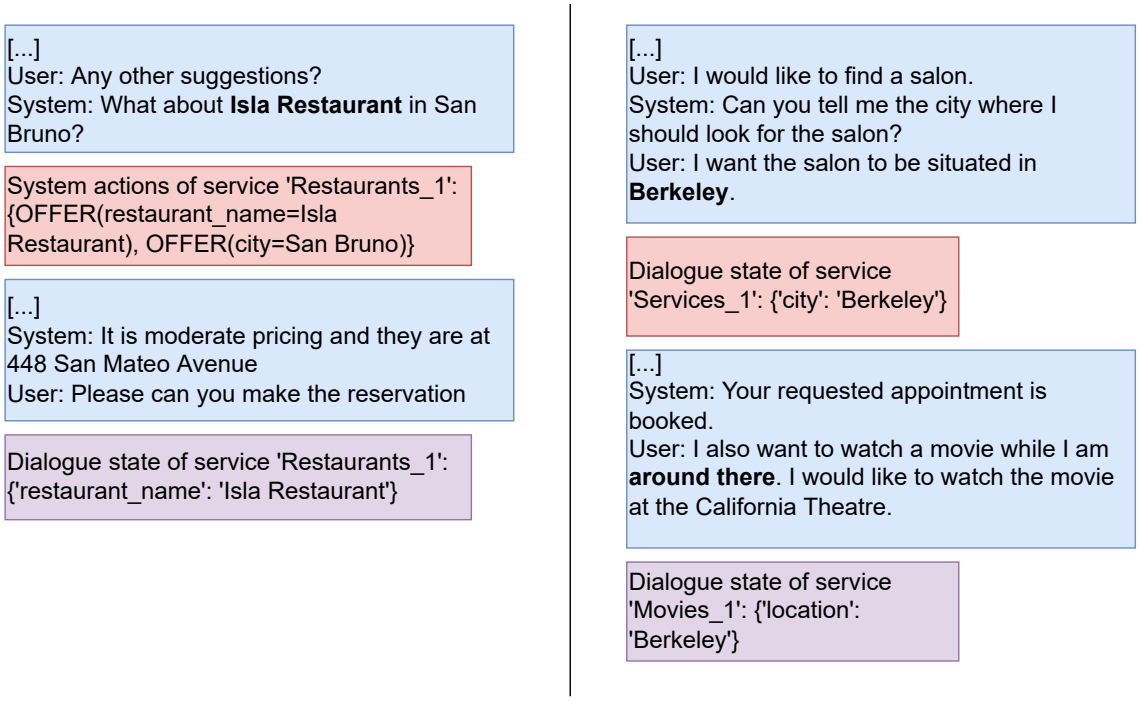


Figure 5.12: *Examples of in_service_carryover (left) and cross_service_carryover (right).*

5.3.3 Experimental Setup

We use the huggingface [2] implementation of the bert-base-uncased models. We use a batch size of 64 and a dropout rate of 0.3 for the classification heads. We use the AdamW

[2] https://huggingface.co/docs/transformers/model_doc/bert

Table 5.1: *Baseline system 1 results*

System	JGA	Intent Acc	Req Slot F1
SGD-baseline [1]	25.4	90.6	96.5
SGP-DST [19]	**72.2**	91.9	**99.0**
Baseline system 1 w. Fusion Encoder	37.5	92.5	97.2
Baseline system 1 w. Cross Encoder	55.1	**93.3**	98.5

optimizer [23] with a constant learning rate of 2e-5. We train each module separately for a total of 10 epochs and perform early stopping based on the validation loss.

The Transformer Encoder has an input dimension of 128 (the Projection layers transform the BERT outputs from dimension 768 to 128), a hidden dimension of 512 and consists of 4 layers. It is trained from scratch unlike BERT which uses the pre-trained checkpoints.

5.3.4 Results and Discussion

In Table 5.1, we compare Baseline system 1 with the Fusion Encoder and Cross Encoder, SGD-baseline and SGP-DST. Cross Encoder outperforms Fusion Encoder in all tasks indicating that performing early concatenation of the dialogue and the schema is better. However, the differences between the Cross Encoder and the Fusion Encoder are much larger in the slot filling task, because it is more challenging. The SGD-baseline performs worse in all three tasks because the schema elements are encoded before training and no self-attention is performed between them and the encoded utterances. Additionally, it does not use slot carryover mechanisms or hand-crafted binary features.

In the task of intent prediction both our Encoders outperform SGP-DST. In SGP-DST the binary feature indicating whether the candidate intent is found in the previous dialogue state is provided with an additional context feature embedding added to BERT's token, segment and position embeddings. We instead directly provide this feature to the classification head which possibly leads to our higher intent accuracy.

In the task of requested slot prediction SGP-DST slightly outperforms Baseline system 1 with Cross Encoder possibly because we do not use any hand-crafted features unlike SGP-DST.

In the task of slot filling, our system outperforms the SGD-baseline indicating that the slot carryover mechanisms and the Encoder architectures are effective but perform much worse than SGP-DST. The most major difference between our methods is that we assume that the slot carryover and the user giving a value for the slot in the current utterance are mutually exclusive, however this is not always the case. In Section 5.4 we modify our architecture trying to reach SGP-DST's performance on the slot filling task.

5.4 Baseline system 2: Unified slot BERT module and encoding of system actions

In this section, we focus on the slot filling task. We merge the slot-specific modules except for the Cross-service Carryover module into one single module and apply multi-task learning. We also modify the slot carryover mechanisms. Additionally, we replace the system utterance with a representation derived from the system actions which we directly encode along with the user utterance and the slot. Because of its superior performance we keep the Cross Encoder architecture for the rest of the experiments. The main goals of this section are to:

- Introduce a method to encode the underlying system actions from the preceding system utterance.
- Improve the previous architecture's slot filling performance by introducing a unified module for the slots and modifying the slot carryover mechanisms.

5.4.1 Input representation for slots

For the slot-related tasks (requested slot prediction and slot filling) we use a single BERT module. It takes as input the concatenation of the following:

- Preceding system utterance represented as the underlying system actions. For example the utterance "In Fremont there is the World Gourmet" is represented by the sequence `[ACTION] Offer [SLOT] restaurant name [VALUE] World Gourmet [ACTION] Offer [SLOT] location [VALUE] Fremont`. A number of additional custom tokens are added to BERT's vocabulary.
- Current user utterance.
- Slot name and description.
- List of possible slot values (only if the slot is categorical).

5.4.2 Unified slot module

After it encodes the sequence, the module uses a total of six classification heads to predict:

- The **user status**: 1) none 2) active 3) dontcare (1 head).
- The **carryover status**: 1) none 2) in_sys_uttr 3) in_service_hist 4) in_cross_service_hist (1 head). Compared to the Baseline system 1 we now separate the user status (whether the user explicitly gives the value) and the carryover status because their function is not mutually exclusive. For in service carryover we separate into two cases: in_sys_uttr (the value is present in a system action of the preceding system utterance) and in_service_hist (the value is present in a previous system action in

the dialogue). Nevertheless, the slot carryover mechanisms work as described in Section 5.3.2.

- The **requested status**: 1) none 2) active (1 head).
- For non-categorical slots the **span** in the user utterance corresponding to the slot (2 heads) and for categorical slots its **value** with classification over the list of possible values (1 head).

The user status, carryover status and requested status heads take as input the encoded `[CLS]` token, the non-categorical span (start and end) heads take as input the encoded user utterance tokens and the categorical head takes as input the max pooled representation of the encoded tokens corresponding to each value. See Figure 5.13 for an example.

For every slot s we employ additional binary features $x_{bin}(s)$ (slightly different from the ones in the previous architecture). They are provided as an additional input to the user status, carryover status and requested status classification heads. The binary features used are the following: 1) whether the service is new in the dialogue 2) whether the service switched (it was not present in the previous dialogue state) 3) whether exactly one value for the slot is found in the system utterance 4) whether exactly one value for the slot is found in previous system utterances 5) whether the slot is required in at least one intent 6) whether the slot is optional in all intents. Similar binary features have been used by [19].

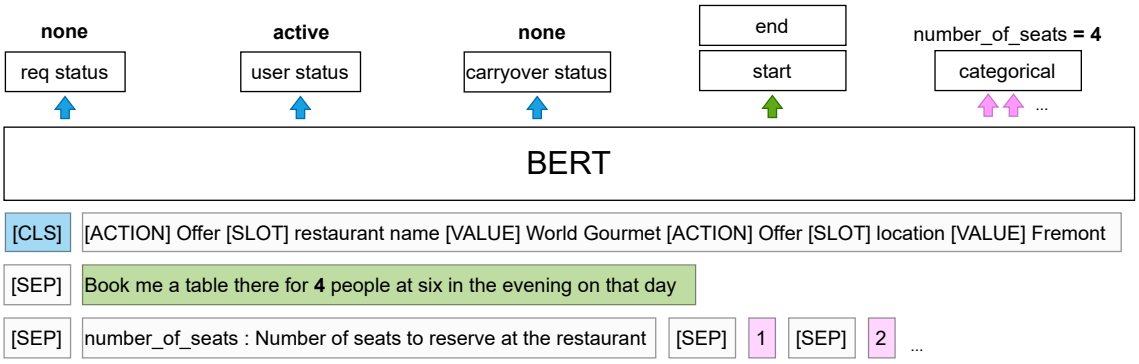


Figure 5.13: *The unified slot module (top) along with an example input sequence corresponding to the categorical slot "number_of_seats" (bottom). In this example the slot is not requested (requested status = none) and the user gives a value for the slot (user status = active). The categorical head picks the given value from the list of possible values (number_of_seats = 4). Note that the start and end heads are not activated in this example; they are only activated for non-categorical slots.*

5.4.3 Experimental setup

We fine-tune BERT and jointly train the classification heads. Depending on the slot type only the relevant classification heads contribute to the loss. We create separate batches for categorical and non-categorical slots and randomly shuffle them. All other training and model hyperparameters are the same as in 5.3.3.

Table 5.2: *Baseline system 2 results on the slot filling task*

System	JGA	Avg GA
SGD-baseline [1]	25.4	56.1
SGP-DST [19]	**73.8**	**92.0**
Baseline system 1 w. Cross Encoder	55.1	84.7
Baseline system 2 w. Cross Encoder	**73.8**	91.8

5.4.4 Results and discussion

In Table 5.2, we compare Baseline system 1, Baseline system 2, SGD-baseline and SGP-DST on the slot filling task. Baseline system 2 outperforms Baseline system 1 and matches SGP-DST's performance. Compared to Baseline system 1, the most important changes are that we separated the user status and the carryover status and we included system actions for the system utterance. Although SGP-DST also considers system actions, it does so with handcrafted features.

The difference with SGP-DST is that they fine-tune a total of six BERT models while we fine-tune only three: the Intent module, the Unified Slot module and the Cross-service Carryover module. A more unified approach not only reduces training time but also makes multi-task training possible. For example, unlike SGP-DST, we encode categorical and non-categorical slots with the same model. However, it is still not easy to combine all three DST tasks with this formulation and we need to explore alternatives.

5.5 Proposed model

In this section, we propose a single multi-task BERT-based model that jointly performs intent prediction, requested slot prediction and slot filling. In the proposed model, we adopt slot carryover mechanisms and encode only the preceding system utterance and the current user utterance like in the previous baseline systems. Furthermore, the preceding system utterance is abstracted and represented as its underlying system actions. To achieve a more efficient and parsimonious input representation, we encode all of the schema elements together using only their names and we selectively include past dialogue states. This is the most important change compared to the baseline systems. Our proposed model significantly outperforms the baseline SGP-DST system and achieves near state-of-the-art performance. Extensive ablation studies reveal the impact of each strategy of our model on the slot filling task.

The multi-task model architecture is shown in Figure 5.14. The user utterance, previous system utterance, schema(ta) and past DST information (see Part 1 to 5) are encoded via BERT. Different pieces of the encoded sequence (see matching color coding in figure) are given as input to nine classification heads that perform the tasks of intent prediction (2 heads), requested slot prediction, slot filling (4 heads) and slot carryover (2 heads).

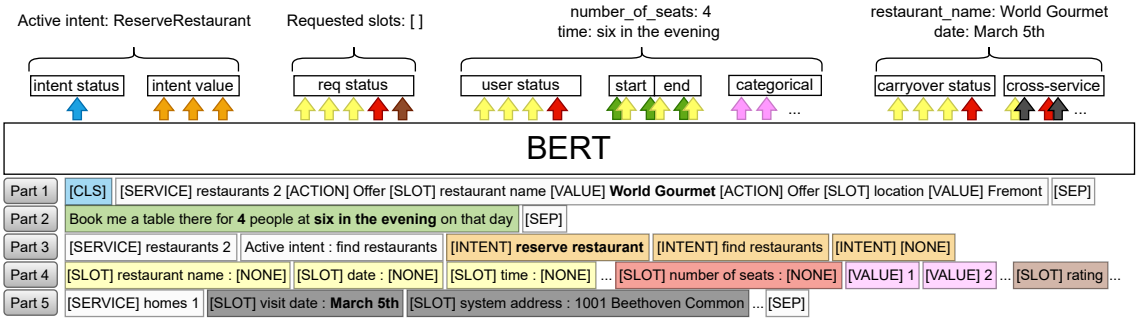


Figure 5.14: *The inputs to the intent prediction, requested slot prediction, slot filling and slot carryover heads are shown for our proposed multi-task BERT model (top), along with an example encoding of the utterance and dialogue history that is the input to the base BERT model (bottom). Note the color coding of the input to the classification heads (top) that matches the various parts of the input sequence (bottom). For this example, the service in the system and the user utterance is Restaurants_2. The previous intent FindRestaurants changes to ReserveRestaurant. No slots are requested by the user. In the preceding system utterance, the system offers the value "World Gourmet" for the slot restaurant_name which the user accepts (slot carryover in_sys_uttr). The user gives the values "six in the evening" and "4" for the non-categorical slot time and the categorical slot number_of_seats. The date value is not uttered but it is implied that it has been mentioned before (slot carryover in_cross_service_hist from a previous service (Homes_1)). Part of the input is truncated for illustration purposes.*

5.5.1 Notation

Let n be a dialogue service, $I(n)$ the set of intents in the service (including the special `[NONE]` intent) and $S(n)$ the set of slots in the service. Slots are divided to categorical and non-categorical slots. Let $S_{cat}(n) \subseteq S(n)$ be the set of categorical slots and $S_{noncat}(n) \subseteq S(n)$ the set of non-categorical slots. For every categorical slot, a set of possible values $V(s), s \in S_{cat}(n)$ are available. Furthermore, every slot may be informable or not depending on whether the user is allowed to give a value for it. We denote the set of the service informable slots as $S_{inf}(n) \subseteq S(n)$.

Assume that at user turn t of a dialogue with N services we want to predict the dialogue state for service n. Essentially we have to predict the active intent $int(n)$ (intent prediction), the requested slots $req(n) \subseteq S(n)$ (requested slot prediction) and the values for the slots given by the user $usrSlotValue(s), s \in S_{inf}(n)$ (slot filling).

For every service $n', 1 \leq n' \leq N$, we denote its previous active intent as $prevInt(n')$. Also, for every slot $s \in S(n')$ we denote the last value given by the user for s as $prevUsrSlotValue(s)$. Furthermore, we use $prevSysSlotValue(s)$ and $sysUttrSlotValue(s)$ to denote the last value present in a system action, before turn $t-1$ and at (system) turn $t-1$ respectively. For $prevSysSlotValue(s)$ and $sysUttrSlotValue(s)$ we only use system actions that contain the slot s and exactly one value for the slot. In cases where the intent or the slot value is empty we use the `[NONE]` value.

We use S_{prev} to denote the set of slots $s \in S(n'), n' \neq n$ that $prevUsrSlotValue(s)$ or $prevSysSlotValue(s)$ is not `[NONE]` and $prevSlotValue(s)$ to denote their previous value. If $prevUsrSlotValue(s)$ is not `[NONE]` we use that value otherwise we use $prevSysSlotValue(s)$.

5.5.2 Input representation

An example input can be seen in Figure 5.14. In Part 1 we encode the preceding system utterance as a list of actions. In Part 2 we encode the current user utterance. In Part 3, the active service n, the previous active intent $prevInt(n)$ and all candidate intents belonging to service n are enumerated. Part 4 contains the list of all slots $s \in S(n)$. If $s \in S_{inf}(n)$ we append $prevUsrSlotValue(s)$ and if $s \in S_{cat}(n) \cap S_{inf}(n)$ we also append all values in $V(s)$. Part 5 contains all other services found earlier in the dialogue. For every service we enumerate slot-value pairs from previous dialogue states or system actions, $s \in S_{prev}$ and their values $prevSlotValue(s)$. We prepend the word "system" before slots given by the system to differentiate them from slots given by the user (present in previous dialogue states).

For the schema we only use the names for the slots and intents instead of their full natural language descriptions used by other works. A number of custom tokens are introduced to the BERT vocabulary to indicate intents, slots etc.

5.5.3 Intent prediction task

Intent status head. We perform binary classification on the encoded `[CLS]` representation to predict the intent status as active or none.
Intent value head. For every intent $i \in I(n)$ we perform binary classification on its encoded `[INTENT]` representation to predict if the user switches to that intent.

If the intent status is active we choose the intent with the highest intent value probability. Otherwise, we keep the previous intent $prevInt(n)$.

5.5.4 Requested slot prediction task

Requested status head. For every slot $s \in S(n)$ we perform binary classification on its encoded `[SLOT]` representation in Part 4 to decide whether it is requested in the current user utterance.

5.5.5 Slot filling task

User status head. For every slot $s \in S_{inf}(n)$ we find the user status using its encoded `[SLOT]` representation in Part 4 to decide whether a value is given in the current user utterance. The user status classes are none, active and dontcare.
Categorical head. For the categorical slots $s \in S_{inf}(n) \cap S_{cat}(n)$ we perform binary classification for every possible value $v \in V(s)$ on its encoded `[VALUE]` representation to predict whether it is present in the user utterance.
Start and **end heads**. For the non-categorical slots $s \in S_{inf}(n) \cap S_{noncat}(n)$ we find the start and end span index distribution in the user utterance by performing classification on the concatenation of every user utterance token with the encoded `[SLOT]` representation.

If the user status is active, the value or the span with the highest probability is chosen for the slot. If the user status is dontcare, the special dontcare value is assigned to the slot.

5.5.6 Slot carryover

The user does not always explicitly give the value for the slot but they may instead refer to previous utterances. Therefore, we design slot carryover mechanisms to retrieve values for slots from the current or previous services.

Carryover status head. For every slot $s \in S_{inf}(n)$ we predict the carryover status using its encoded `[SLOT]` representation in Part 4 to find the source of the slot value. The carryover status classes are none, in_sys_uttr, in_service_hist and in_cross_service_hist.

For in_sys_uttr the slot is updated according to the value present in the preceding system utterance $sysUttrSlotValue(s)$. For in_service_hist the slot is updated according to the value present in past system actions of service n, $prevSysSlotValue(s)$. In the above two cases, the user accepts the value given by the system and we simply carry that value over.

Cross-service head. For every slot $s' \in S_{prev}(n)$ we perform binary classification on the concatentation of its encoded `[SLOT]` representation in Part 5 with the encoded `[SLOT]` representation of s in Part 4 to decide whether we should carry the value over from slot s' to slot s. The highest probability slot s' is used as the source for the value s if the predicted carryover status is in_cross_service_hist. In this case, we assign the value $prevSlotValue(s')$ to slot s.

We first check the user status and if it is not none we update the slot value according to its output. Otherwise, we also check the carryover status. If it predicts that a carryover should take place, we update the slot value accordingly. If both user and carryover status are none the value remains the same as in the previous dialogue state, $prevUsrSlotValue(s)$.

5.5.7 Multi-task training

For the intent status, intent value, categorical, start, end and cross-service classification heads we derive the class probabilities with a two-layer feed-forward neural network. For the requested status, user status and carryover status classification heads we concatenate the slot binary features $x_{bin}(s)$ after the first layer.

We jointly optimize all classification heads, using the cross entropy loss for each head. For the intent prediction task the loss is $L_1 = w_1 L_{intstat} + w_2 L_{intval}$, for the requested slot prediction $L_2 = L_{reqstat}$ and for the slot filling task $L_3 = w_3 L_{usr} + w_4 L_{carry} + w_5 L_{cat} + w_6 L_{start} + w_7 L_{end} + w_8 L_{cross}$. Finally, the total loss is defined as $L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3$.

5.6 Experimental Setup

5.6.1 Label Acquisition

In order to acquire labels for the user and carryover status, we use the user actions and search previous turns and dialogue states to find the source for the slot. We consider a slot as informable if and only if it is either required or optional in at least one intent. For every turn we run the model only for the involved services (services with at least one

Table 5.3: *Comparison to other works*

System	Model	Params	JGA	Intent Acc	Req Slot F1
SGD-baseline [1]	$\text{BERT}_{\text{BASE}}$	**110M**	25.4	90.6	96.5
SGP-DST [19]	$6 \times \text{BERT}_{\text{BASE}}$	660M	72.2	91.9	99.0
paDST [21]	$3 \times \text{RoBERTa}_{\text{BASE}}$+ $\text{XLNet}_{\text{LARGE}}$	715M	**86.5**	94.8	98.5
D3ST [22] (Base)	T5_{BASE}	220M	72.9	97.2	98.9
D3ST [22] (Large)	T5_{LARGE}	770M	80.0	97.1	99.1
D3ST [22] (XXL)	T5_{XXL}	11B	86.4	**98.8**	**99.4**
Ours (median result)	$\text{BERT}_{\text{BASE}}$	**110M**	82.7	94.6	**99.4**
Ours (avg 3 runs)	$\text{BERT}_{\text{BASE}}$	**110M**	82.5 ± 1.0	94.7 ± 0.5	**99.4** ± 0.1

change in the dialogue state in the turn) according to the ground truth dialogue states during both training and evaluation for fair comparison to other works. The input to the model contains ground-truth previous dialogue states during training and during evaluation the previously predicted ones are used.

5.6.2 Training Setup

We use the huggingface [3] implementation of the BERT uncased models. For all our experiments we use a batch size of 16 and a dropout rate of 0.3 for the classification heads. We use the AdamW optimizer [23] with a linear warmup of 10% of the training steps and learning rate 2e-5. We train for a total of about 55k steps and evaluate on the development set every 4k steps. We choose the model that performs best based on the JGA metric on the development set.

5.6.3 Preprocessing and augmentation

We preprocess the schema elements and the system actions by removing underscores and splitting the words when on CamelCase and snake_case style. We randomly ($p = 0.1$) replace the input tokens in the user utterance with the `[UNK]` token (word dropout) and shuffle the order of the schema elements in Parts 3-5 during training as proposed by [24]. We also apply random ($p = 0.1$) data augmentation through synonym replacement and random swap to the intents, slots and values in Parts 3-4 (schema augm.) via [25].

5.7 Results and Discussion

5.7.1 Comparison to other works

In Table 5.3 we compare our model to SGD-baseline, SGP-DST, paDST and three D3ST implementations of variable size. The SGD baseline [1] fine-tunes BERT with the last two utterances as input and uses precomputed BERT embeddings for the schema. SGP-DST [19] uses the last two utterances and slot carryover mechanisms to retrieve values for slots which were mentioned in previous utterances. paDST [21] and D3ST [22] encode the entire dialogue history until the current turn and calculate the dialogue state from

[3] https://huggingface.co/docs/transformers/model_doc/bert

Table 5.4: *Ablation study*

System	JGA	Avg GA
Ours	**82.7**	**95.2**
w/o system actions	71.9	91.6
w. slot descriptions	78.3	94.1
w/o previous state	79.8	94.0
w/o schema augm.	80.5	94.9
w/o schema augm. & word dropout	78.1	94.3
w/o binary features	81.0	94.4

Table 5.5: *Effect of carryover mechanisms*

System	JGA	Avg GA
Ours	**82.7**	**95.2**
w/o in_sys_uttr	62.8	87.0
w/o in_service_hist	76.4	92.7
w/o in_cross_service_hist	66.8	84.4
SGD-baseline [1]	25.4	56.0
w/o in_service_hist & in_cross_service_hist	61.6	81.9
w/o all	36.5	68.5

scratch. We report the metrics and the number of parameters in the pretrained model(s) fine-tuned by each method.

Our method clearly outperforms SGP-DST in all tasks indicating that our strategies are effective. Some of the entire-dialogue models outperform our model, especially when they use much more parameters (D3ST XXL) or apply more handcrafted features, special rules and dialogue augmentation through back-translation (paDST). Overall, the proposed approach achieves near state-of-the-art performance despite using a much smaller model size and a shorter input representation.

5.7.2 Ablation study

We perform an ablation study (Table 5.4) to show the contribution of each of the proposed strategies on the slot filling task. Replacing the system utterance with a set of system actions (w/o system actions) has the biggest effect on performance (see input sequence Part 1 in Figure 5.14). The system actions contain key information including the slot names and their respective values, helping our model identify which slots are requested, offered, confirmed etc. and predict the user and carryover status most accurately. Performance drops when we additionally include the slot descriptions for the informable slots of the current service (w. slot descriptions, see Parts 3-4 of input). By removing previous intent and slot values in Parts 3-4 (w/o previous state) we observe a performance drop but also a training speedup because of the smaller input sequence. We also observe an improvement by performing schema augmentation and word dropout possibly because these strategies help to avoid overfitting (w/o schema augm. & word dropout). The hand-crafted binary features can slightly benefit the system (w/o binary

features).

5.7.3 Effect of slot carryover mechanisms

In Table 5.5 we show the effect of the various slot carryover mechanisms. For these experiments the model is trained once and during evaluation we replace each carryover status class with "none". As expected, dropping "in_sys_uttr" has the biggest impact on performance. "in_cross_service_hist" is also important because of the large number of multi-domain dialogues. By removing "in_service_hist", performance is less affected. Without "in_service_hist" and "in_cross_service_hist" (by only considering the last two utterances) we still achieve a higher accuracy than the SGD-baseline.

5.7.4 Discussion

The proposed system is shown to significantly improve performance over the SGP-DST system that also uses slot carryover mechanisms. The experimental results indicate that the strategies to create an efficient input sequence are effective. Furthermore, the system only uses a single BERT model and the parsimonious input sequence enables solving the three tasks with just one BERT-pass per turn. This leads to more computational efficiency.

Compared to the state-of-the-art, the most important difference is that we utilize slot carryover mechanisms, however encoding the entire dialogue history performs better. In our setting the names of the schema elements perform better than the full schema descriptions which we did not manage to effectively incorporate in our model. Additionally, our pre-trained model (BERT) is smaller. paDST uses a large number of handcrafted features and rules which require a lot of feature engineering and may be too dataset-specific. On the other hand, D3ST achieves a higher JGA than ours only with the largest version of T5 which has 100 times more parameters than BERT. Overall, our system is computationally efficient and scalable to large schemata and dialogues.

Chapter 6

Conclusions

6.1 Conclusions

In this thesis, we studied in depth the topic of schema-guided dialogue state tracking. We first introduced important machine learning concepts and provided some recent deep learning breakthroughs. We analyzed vanilla RNNs, LSTMs, attention and the transformer as well as two important and widely used learning techniques: transfer learning and multi-task learning. We then focused on the field of natural language processing. Specifically, after we introduced traditional language models we moved on to modern transformer-based language models such as BERT and T5.

We provided an overview of dialogue systems examining methods, challenges and evaluation of the two main categories: open-domain and task-oriented dialogue systems. We then focused our interest in dialogue state tracking and we presented the latest research on the field. We noticed that schema-guided systems, which aim to separate the model and its supported services, are gaining a lot of interest lately.

We first studied schema-guided dialogue state tracking and introduced two baseline systems. We then proposed a novel multi-task system for schema-guided dialogue state tracking based on a single BERT model and an efficient and parsimonious input representation. Our system reasons for three critical DST tasks simultaneously. Close to state-of-the-art performance is achieved, using a significantly smaller model and input encoding. Among the various proposed enhancements to the model we show that abstracting the preceding system utterance with system actions gives the biggest performance boost. Strategies like appending previous dialogue states, data augmentation and adding hand-crafted features further improve performance.

We believe that these strategies can guide the design of accurate, efficient and ontology-independent task-oriented DST capable of scaling to large multi-domain dialogues, important in real world applications.

6.2 Future work

In the future, extensions to the work can include:

- **Replacing BERT with stronger models which have better zero-shot performance and therefore potentially improve accuracy on unseen services.** For example, although D3ST [22] works with a very simple seq2seq approach, it has state-of-the-art performance when it uses the largest version of T5.

- **Further pre-training models on related tasks.** In Chapter 4 we showed that pre-training on tasks like machine reading comprehension, for which more large-scale datasets are available, can benefit DST. Schema-guided DST is even more related to MRC than traditional DST tasks as we showed that the slot/intent description can be seen as the MRC's question. However, our approach answers multiple "questions" at the same time and therefore pre-training from traditional MRC models that only answer one question for each passage may need to be adapted.

- **Strategies to incorporate earlier utterances and more information about the schema, such as the descriptions of its elements, without increasing the input size.** For example, a turn-level recurrent model (e.g. LSTM) could be used to learn to encode useful information from previous utterances.

- **More data augmentation to avoid the problems of overfitting to seen services.** The number of services seen during training are limited and therefore it is very easy for the model to overfit and "memorize" them. We showed a simple data augmentation technique but more sophisticated ones may perform even better. As an example, future work can implement back-translation with transformer models.

- **Incorporating the schema-guided paradigm in other dialogue components such as the dialogue policy or even to end-to-end dialogues.** In order to build a real-world system it is also important to train a dialogue policy component, however this problem is less studied. One possible direction could be end-to-end systems that jointly learn to predict the dialogue state and respond.

Bibliography

[1] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta και Pranav Khaitan. *Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset*, 2020.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever και Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[3] *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[4] *The Unreasonable Effectiveness of Recurrent Neural Networks*. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[5] Lilian Weng. *Attention? Attention!* *lilianweng.github.io*, 2018.

[6] Dzmitry Bahdanau, Kyunghyun Cho και Yoshua Bengio. *Neural machine translation by jointly learning to align and translate*. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser και Illia Polosukhin. *Attention is all you need*. *Advances in neural information processing systems*, 30, 2017.

[8] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*, 2013.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. *CoRR*, abs/1810.04805, 2018.

[10] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov και Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. *CoRR*, abs/1906.08237, 2019.

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li και Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. *CoRR*, abs/1910.10683, 2019.

[12] Adam Roberts, Colin Raffel και Noam Shazeer. *How Much Knowledge Can You Pack Into the Parameters of a Language Model?* *CoRR*, abs/2002.08910, 2020.

[13] Minlie Huang, Xiaoyan Zhu και Jianfeng Gao. *Challenges in Building Intelligent Open-domain Dialog Systems*. *CoRR*, abs/1905.05709, 2019.

[14] Daniel Jurafsky και James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009.

[15] Zheng Zhang, Ryuichi Takanobu, Minlie Huang και Xiaoyan Zhu. *Recent Advances and Challenges in Task-oriented Dialog System*. *CoRR*, abs/2003.07490, 2020.

[16] Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao και Dilek Hakkani-Tür. *MultiWOZ 2.1: Multi-Domain Dialogue State Corrections and State Tracking Baselines*. *CoRR*, abs/1907.01669, 2019.

[17] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2018.

[18] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta και Pranav Khaitan. *Schema-Guided Dialogue State Tracking Task at DSTC8*. *CoRR*, abs/2002.01359, 2020.

[19] Yu-Ping Ruan, Zhen-Hua Ling, Jia-Chen Gu και Quan Liu. *Fine-Tuning BERT for Schema-Guided Zero-Shot Dialogue State Tracking*. *CoRR*, abs/2002.00181, 2020.

[20] Miao Li, Haoqi Xiong και Yunbo Cao. *The SPPD System for Schema Guided Dialogue State Tracking Challenge*. *CoRR*, abs/2006.09035, 2020.

[21] Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou και Jianping Shen. *An End-to-End Dialogue State Tracking System with Machine Reading Comprehension and Wide & Deep Classification*. *CoRR*, abs/1912.09297, 2019.

[22] Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran και Yonghui Wu. *Description-Driven Task-Oriented Dialog Modeling*. *CoRR*, abs/2201.08904, 2022.

[23] Ilya Loshchilov και Frank Hutter. *Fixing Weight Decay Regularization in Adam*. *CoRR*, abs/1711.05101, 2017.

[24] Sungdong Kim, Sohee Yang, Gyuwan Kim και Sang-Woo Lee. *Efficient Dialogue State Tracking by Selectively Overwriting Memory*. *CoRR*, abs/1911.03906, 2019.

[25] Jason Wei και Kai Zou. *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, σελίδες 6383–6389, Hong Kong, China, 2019. Association for Computational Linguistics.

[26] Shai Shalev-Shwartz και Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[27] Stuart Russell και Peter Norvig. *Artificial intelligence: a modern approach*. 2002.

[28] Geoffrey Hinton και Terrence J Sejnowski. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[29] Richard S Sutton και Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[30] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[31] Jeffrey L Elman. *Finding structure in time*. *Cognitive science*, 14(2):179–211, 1990.

[32] Ilya Sutskever, Oriol Vinyals και Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. *CoRR*, abs/1409.3215, 2014.

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

[34] Sepp Hochreiter και Jürgen Schmidhuber. *Long short-term memory*. *Neural computation*, 9(8):1735–1780, 1997.

[35] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau και Yoshua Bengio. *On the properties of neural machine translation: Encoder-decoder approaches*. *arXiv preprint arXiv:1409.1259*, 2014.

[36] Mike Schuster και Kuldip K Paliwal. *Bidirectional recurrent neural networks*. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[37] Minh-Thang Luong, Hieu Pham και Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. *CoRR*, abs/1508.04025, 2015.

[38] Denny Britz, Anna Goldie, Minh-Thang Luong και Quoc V. Le. *Massive Exploration of Neural Machine Translation Architectures*. *CoRR*, abs/1703.03906, 2017.

[39] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel και Yoshua Bengio. *Show, attend and tell: Neural image caption generation with visual attention*. *International conference on machine learning*, σελίδες 2048–2057. PMLR, 2015.

[40] Sebastian Ruder. *Transfer Learning - Machine Learning's Next Frontier*. http://ruder.io/transfer-learning/, 2017.

[41] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. *CoRR*, abs/1706.05098, 2017.

[42] Zellig S. Harris. *Distributional Structure*. WORD, 10(2-3):146–162, 1954.

[43] J. R. Firth. *A synopsis of linguistic theory 1930-55*. 1952-59:1–32, 1957.

[44] Jeffrey Pennington, Richard Socher και Christopher Manning. *GloVe: Global Vectors for Word Representation*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.

[45] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee και Luke Zettlemoyer. *Deep contextualized word representations*, 2018.

[46] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer και Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. *CoRR*, abs/1907.11692, 2019.

[47] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le και Ruslan Salakhutdinov. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. *CoRR*, abs/1901.02860, 2019.

[48] Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, Vinay Adiga και Erik Cambria. *Recent Advances in Deep Learning Based Dialogue Systems: A Systematic Survey*. *CoRR*, abs/2105.04387, 2021.

[49] Jianfeng Gao, Michel Galley και Lihong Li. *Neural Approaches to Conversational AI*. *arXiv:1809.08267 [cs]*, 2019. arXiv: 1809.08267.

[50] Joseph Weizenbaum. *ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine*. *Commun. ACM*, 9(1):36–45, 1966.

[51] Kenneth Mark Colby, Sylvia Weber και Franklin Dennis Hilf. *Artificial paranoia*. *Artificial Intelligence*, 2(1):1–25, 1971.

[52] Richard S Wallace. *The anatomy of ALICE*. *Parsing the turing test*, σελίδες 181–210. Springer, 2009.

[53] Tie Yan Liu και others. *Learning to rank for information retrieval*. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[54] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux και Jason Weston. *Real-time Inference in Multi-sentence Tasks with Deep Pretrained Transformers*. *CoRR*, abs/1905.01969, 2019.

[55] Alec Radford και Karthik Narasimhan. *Improving Language Understanding by Generative Pre-Training*. 2018.

[56] Thomas Wolf, Victor Sanh, Julien Chaumond και Clement Delangue. *TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents*. *CoRR*, abs/1901.08149, 2019.

[57] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu και Bill Dolan. *DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation*. *CoRR*, abs/1911.00536, 2019.

[58] Tiancheng Zhao, Ran Zhao και Maxine Eskénazi. *Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders*. *CoRR*, abs/1703.10960, 2017.

[59] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter και Dan Jurafsky. *Adversarial learning for neural dialogue generation*. *arXiv preprint arXiv:1701.06547*, 2017.

[60] Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li και Ming Zhou. *Response generation by context-aware prototype editing*. *Proceedings of the AAAI Conference on Artificial Intelligence*, τόμος 33, σελίδες 7281–7288, 2019.

[61] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao και Bill Dolan. *A diversity-promoting objective function for neural conversation models*. *arXiv preprint arXiv:1510.03055*, 2015.

[62] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville και Yoshua Bengio. *Generative Adversarial Networks*, 2014.

[63] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville και Yoshua Bengio. *A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues*. *CoRR*, abs/1605.06069, 2016.

[64] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao και Bill Dolan. *A Persona-Based Neural Conversation Model*. *CoRR*, abs/1603.06155, 2016.

[65] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela και Jason Weston. *Personalizing Dialogue Agents: I have a dog, do you have pets too?* *CoRR*, abs/1801.07243, 2018.

[66] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh και Dhruv Batra. *Visual Dialog*. *CoRR*, abs/1611.08669, 2016.

[67] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih και Michel Galley. *A Knowledge-Grounded Neural Conversation Model*. *CoRR*, abs/1702.01932, 2017.

[68] Kishore Papineni, Salim Roukos, Todd Ward και Wei Jing Zhu. *Bleu: a method for automatic evaluation of machine translation*. *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, σελίδες 311–318, 2002.

[69] Chin Yew Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. *Text Summarization Branches Out*, σελίδες 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics.

[70] Satanjeev Banerjee και Alon Lavie. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. *Proceedings of the ACL*

Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, σελίδες 65–72, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

[71] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio και Joelle Pineau. *Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses*. *CoRR*, abs/1708.07149, 2017.

[72] Kaisheng Yao, Geoffrey Zweig, Mei Yuh Hwang, Yangyang Shi και Dong Yu. *Recurrent neural networks for language understanding*. *Interspeech*, σελίδες 2524–2528, 2013.

[73] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig και Yangyang Shi. *Spoken language understanding using long short-term memory neural networks*. *2014 IEEE Spoken Language Technology Workshop (SLT)*, σελίδες 189–194. IEEE, 2014.

[74] Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun Nung Chen, Jianfeng Gao, Li Deng και Ye Yi Wang. *Multi-domain joint semantic frame parsing using bi-directional rnn-lstm*. *Interspeech*, σελίδες 715–719, 2016.

[75] Qian Chen, Zhu Zhuo και Wen Wang. *Bert for joint intent classification and slot filling*. *arXiv preprint arXiv:1902.10909*, 2019.

[76] Steve Young, Milica Gašić, Blaise Thomson και Jason D Williams. *Pomdp-based statistical spoken dialog systems: A review*. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[77] Nikola Mrkšić, Diarmuid O Séaghdha, Tsung Hsien Wen, Blaise Thomson και Steve Young. *Neural belief tracker: Data-driven dialogue state tracking*. *arXiv preprint arXiv:1606.03777*, 2016.

[78] Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed και Li Deng. *Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems*. *Proceedings of the AAAI Conference on Artificial Intelligence*, τόμος 32, 2018.

[79] Weiyan Shi, Kun Qian, Xuewei Wang και Zhou Yu. *How to build user simulators to train rl-based dialog systems*. *arXiv preprint arXiv:1909.01388*, 2019.

[80] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam Fai Wong και Shang Yu Su. *Deep dyna-q: Integrating planning for task-completion dialogue policy learning*. *arXiv preprint arXiv:1801.06176*, 2018.

[81] Tsung Hsien Wen, Milica Gasic, Nikola Mrksic, Pei Hao Su, David Vandyke και Steve Young. *Semantically conditioned lstm-based natural language generation for spoken dialogue systems*. *arXiv preprint arXiv:1508.01745*, 2015.

[82] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng και Jianfeng Gao. *Few-shot natural language generation for task-oriented dialog*. *arXiv preprint arXiv:2002.12328*, 2020.

[83] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever και others. *Language models are unsupervised multitask learners*. *OpenAI blog*, 1(8):9, 2019.

[84] Tsung Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei Hao Su, Stefan Ultes και Steve Young. *A network-based end-to-end trainable task-oriented dialogue system*. *arXiv preprint arXiv:1604.04562*, 2016.

[85] Antoine Bordes, Y Lan Boureau και Jason Weston. *Learning end-to-end goal-oriented dialog*. *arXiv preprint arXiv:1605.07683*, 2016.

[86] Wenqiang Lei, Xisen Jin, Min Yen Kan, Zhaochun Ren, Xiangnan He και Dawei Yin. *Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures*. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, σελίδες 1437–1447, 2018.

[87] Ehsan Hosseini-Asl, Bryan McCann, Chien Sheng Wu, Semih Yavuz και Richard Socher. *A simple language model for task-oriented dialogue*. *Advances in Neural Information Processing Systems*, 33:20179–20191, 2020.

[88] Victor Zhong, Caiming Xiong και Richard Socher. *Global-Locally Self-Attentive Encoder for Dialogue State Tracking*. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, σελίδες 1458–1467, Melbourne, Australia, 2018. Association for Computational Linguistics.

[89] Jason Williams, Antoine Raux, Deepak Ramachandran και Alan Black. *The Dialog State Tracking Challenge*. *Proceedings of the SIGDIAL 2013 Conference*, σελίδες 404–413, Metz, France, 2013. Association for Computational Linguistics.

[90] Matthew Henderson, Blaise Thomson και Jason D. Williams. *The Second Dialog State Tracking Challenge*. *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, σελίδες 263–272, Philadelphia, PA, U.S.A., 2014. Association for Computational Linguistics.

[91] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke και Steve J. Young. *A Network-based End-to-End Trainable Task-oriented Dialogue System*. *CoRR*, abs/1604.04562, 2016.

[92] Paweł Budzianowski, Tsung Hsien Wen, Bo Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan και Milica Gašić. *MultiWOZ-A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling*. *arXiv preprint arXiv:1810.00278*, 2018.

[93] Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang και Jindong Chen. *MultiWOZ 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines*. *arXiv preprint arXiv:2007.12720*, 2020.

[94] Ting Han, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng και Minlie Huang. *MultiWOZ 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation*. *CCF International Conference on Natural Language Processing and Chinese Computing*, σελίδες 206–218. Springer, 2021.

[95] Fanghua Ye, Jarana Manotumruksa και Emine Yilmaz. *Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation*. *arXiv preprint arXiv:2104.00773*, 2021.

[96] Matthew Henderson, Blaise Thomson και Steve Young. *Deep neural network approach for the dialog state tracking challenge*. *Proceedings of the SIGDIAL 2013 Conference*, σελίδες 467–471, 2013.

[97] Elnaz Nouri και Ehsan Hosseini-Asl. *Toward Scalable Neural Dialogue State Tracking Model*. *CoRR*, abs/1812.00899, 2018.

[98] Liliang Ren, Kaige Xie, Lu Chen και Kai Yu. *Towards Universal Dialogue State Tracking*. *CoRR*, abs/1810.09587, 2018.

[99] Jiatao Gu, Zhengdong Lu, Hang Li και Victor O. K. Li. *Incorporating Copying Mechanism in Sequence-to-Sequence Learning*. *CoRR*, abs/1603.06393, 2016.

[100] Puyang Xu και Qi Hu. *An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking*. *CoRR*, abs/1805.01555, 2018.

[101] Oriol Vinyals, Meire Fortunato και Navdeep Jaitly. *Pointer Networks*, 2015.

[102] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher και Pascale Fung. *Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems*. *CoRR*, abs/1905.08743, 2019.

[103] Liliang Ren, Jianmo Ni και Julian J. McAuley. *Scalable and Accurate Dialogue State Tracking via Hierarchical Sequence Generation*. *CoRR*, abs/1909.00754, 2019.

[104] Michael Heck, Carelvan Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi και Milica Gasic. *TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking*. *CoRR*, abs/2005.02877, 2020.

[105] Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S. Yu, Richard Socher και Caiming Xiong. *Find or Classify? Dual Strategy for Slot-Value Predictions on Multi-Domain Dialog State Tracking*. *CoRR*, abs/1910.03544, 2019.

[106] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov και Luke Zettlemoyer. *BART: Denoising*

Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR*, abs/1910.13461, 2019.

[107] Zhaojiang Lin, Andrea Madotto, Genta Indra Winata και Pascale Fung. *MinTL: Minimalist Transfer Learning for Task-Oriented Dialogue Systems*. *CoRR*, abs/2009.12005, 2020.

[108] Yunyi Yang, Yunhao Li και Xiaojun Quan. *UBAR: Towards Fully End-to-End Task-Oriented Dialog Systems with GPT-2*. *CoRR*, abs/2012.03539, 2020.

[109] Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai και Yi Zhang. *Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System*. *CoRR*, abs/2109.14739, 2021.

[110] Jeffrey Zhao, Mahdis Mahdieh, Ye Zhang, Yuan Cao και Yonghui Wu. *Effective Sequence-to-Sequence Dialogue State Tracking*. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, σελίδες 7486–7493, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.

[111] Chengchang Zeng, Shaobo Li, Qin Li, Jie Hu και Jianjun Hu. *A Survey on Machine Reading Comprehension: Tasks, Evaluation Metrics, and Benchmark Datasets*. *CoRR*, abs/2006.11880, 2020.

[112] Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung και Dilek Hakkani-Tür. *Dialog State Tracking: A Neural Reading Comprehension Approach*. *CoRR*, abs/1908.01946, 2019.

[113] Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin και Dilek Hakkani-Tür. *From Machine Reading Comprehension to Dialogue State Tracking: Bridging the Gap*. *CoRR*, abs/2004.05827, 2020.

[114] Johannes E. M. Mosig, Shikib Mehri και Thomas Kober. *STAR: A Schema-Guided Dialog Dataset for Transfer Learning*. *CoRR*, abs/2010.11853, 2020.

[115] Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A. Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho και Rajen Subba. *Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue State Tracking*. *CoRR*, abs/2105.04222, 2021.

[116] Chia-Hsuan Lee, Hao Cheng και Mari Ostendorf. *Dialogue State Tracking with a Language Model using Schema-Driven Prompting*. *CoRR*, abs/2109.07506, 2021.

[117] Vahid Noroozi, Yang Zhang, Evelina Bakhturina και Tomasz Kornuta. *A Fast and Robust BERT-based Dialogue State Tracker for Schema-Guided Dialogue Dataset*. *CoRR*, abs/2008.12335, 2020.

[118] Hwaran Lee, Jinsik Lee και Tae-Yoon Kim. *SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking*. *CoRR*, abs/1907.07421, 2019.

[119] Jie Cao και Yi Zhang. *A Comparative Study on Schema-Guided Dialogue State Tracking*. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, σελίδες 782–796, Online, 2021. Association for Computational Linguistics.

[120] Chetan Naik, Arpit Gupta, Hancheng Ge, Lambert Mathias και Ruhi Sarikaya. *Contextual Slot Carryover for Disparate Schemas*. *CoRR*, abs/1806.01773, 2018.