Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομέας Σημάτων, Ελέγχου και Ρομποτικής
Εργαστηριο Επεξεργασίας Λόγου και Γλώσσας

BloomWise: Enhancing problem-solving capabilities of LLMs using Bloom's-Taxonomy-inspired prompts

Διπλωματικη Εργασια

της

ΜΑΡΙΑΣ ΕΛΕΝΗΣ Χ. ΖΟΥΜΠΟΥΛΙΔΗ

Επιβλέπων: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομέας Σημάτων, Ελέγχου και Ρομποτικής
Εργαστηριο Επεξεργασίας Λόγου και Γλώσσας

BloomWise: Enhancing problem-solving capabilities of LLMs using Bloom's-Taxonomy-inspired prompts

Διπλωματικη Εργασια

της

ΜΑΡΙΑΣ ΕΛΕΝΗΣ Χ. ΖΟΥΜΠΟΥΛΙΔΗ

Επιβλέπων: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Δεκεμβρίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αλέξανδρος Ποταμιάνος	Αθανάσιος Ροντογιάννης	Αθανάσιος Βουλόδημος
Αναπληρωτής Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2024

(Υπογραφή)

.....
Μαρία Ελένη Ζουμπουλίδη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η περιορισμένη ικανότητα των Μεγάλων Γλωσσικών Μοντέλων (ΜΓΜ-LLMs) στα μαθηματικά- δεξιότητα κρίσιμη για την επίλυση σύνθετων προβλημάτων- βρίσκεται στο επίκεντρο του ερευνητικού ενδιαφέροντος. Πολλές προσεγγίσεις επιστρατεύουν το in context learning. Οι κυριότερες εξ αυτών αφορούν στην ενθάρρυνση των ΜΓΜ μέσω προτροπών να προσεγγίσουν το πρόβλημα σταδιακά αναπτύσσοντας τη σκέψη τους σε κειμενική μορφή (Chain of Thought-CoT) ή να επιλύσουν το πρόβλημα με χρήση κώδικα (Program of Thought-PoT). Ωστόσο, τη μεγαλύτερη ακρίβεια επιτυγχάνουν μέθοδοι οι οποίες βασίζονται στην ενσωμάτωση πολλαπλών μεθόδων και επιλογή της κατά περίπτωση κατάλληλης, όπως, παραδείγματος χάριν, η X of Thought (XoT).

Στην παρούσα διπλωματική εργασία, προτείνουμε το BloomWise, μια νέα-εμπνευσμένη από την ταξονομία Bloom- τεχνική prompting η οποία στοχεύει στη βελτίωση των επιδόσεων των ΜΓΜ στην επίλυση μαθηματικών προβλημάτων ενθαρρύνοντάς τα να προσσεγγίσουν το πρόβλημα επιστρατεύοντας αρχικά απλές και προοδευτικά- αν είναι απαραίτητο- ανώτερες πνευματικές δεξιότητες. Μέσω εκτεταμένων πειραμάτων σε διάφορα σύνολα δεδομένων και μοντέλα, καταδεικνύουμε την αποτελεσματικότητα της μεθόδου. Επίσης, παρουσιάζουμε παραλλαγές της προσέγγισης, αναδεικνύουμε τη χρησιμότητα κάθε τμήματος της μεθόδου μέσω κατάλληλων αφαιρέσεων και πραγματοποιούμε εμβριθή ανάλυση των αποτελεσμάτων εστιάζοντας στην αποτελεσματικότητα κάθε πνευματικής δεξιότητας της ταξονομίας τόσο ανά σύνολο δεδομένων όσο και ανά μοντέλο.

Συνάγουμε συμπεράσματα τόσο για τη μέθοδό μας όσο και για τις ικανότητες των ΜΓΜ. Όσον αφορά στη μέθοδό μας, επιτυγχάνει παρεμφερή, και ορισμένες φορές μεγαλύτερη, ακρίβεια από τις προς σύγκριση μεθόδους. Συγκεκριμένα, η επίδοση του BloomWise είναι παρόμοια με αυτή της XoT και καλύτερη από αυτή των CoT και PoT, ενώ η σημαντικά μεγαλύτερη ακρίβεια του Oracle καταδεικνύει τη δυναμική της μεθόδου. Όσον αφορά στα ΜΓΜ, η μέθοδος προσφέρει πολύτιμες πληροφορίες σχετικά με τις γνωστικές δεξιότητες τις οποίες επιδεικνύει κάθε ΜΓΜ, καθώς και τις δεξιότητες οι οποίες απαιτούνται για την επίλυση διαφορετικών τύπων μαθηματικών προβλημάτων, ενισχύοντας την ερμηνευσιμότητα. Μερικές εκ των κυριότερων παρατηρήσεων είναι οι εξής: σε όλα τα μοντέλα η μεγαλύτερη ακρίβεια επεύχθη στα στάδια ¨Ανάλυση¨ και ¨Κατανόηση¨, και για τα δύσκολα προβλήματα επιτυγχάνεται καλύτερη επίδοση στα ανώτερα στάδια της ταξονομίας, ενώ το αντίστροφο δεν επιβεβαιώνεται.

Λέξεις Κλειδιά

LLMs, in-context learning, Ταξονομία Bloom, μαθηματικά προβλήματα, προτροπές

Abstract

The limited ability of Large Language Models (LLMs) in mathematics—a skill critical for solving complex problems—has garnered significant interest from the research community. Many approaches have employed in-context learning to improve LLMs' performance in such tasks. The most prominent of these focus on encouraging LLMs, through prompts, to approach problems gradually by developing their reasoning in textual form (Chain of Thought) or solving the problem using code (Program of Thought). However, the highest accuracy is achieved by methods that integrate multiple approaches and select the appropriate one for each case, such as the X of Thought (XoT).

In this thesis, we propose BloomWise, a new, Bloom's- Taxonomy-inspired prompting technique aimed at improving LLMs' performance in solving mathematical problems. BloomWise encourages models to approach problems initially with simple, and, if necessary, progressively higher cognitive skills. Through extensive experiments on various datasets and models, we demonstrate the effectiveness of the method. Additionally, we present variations of the approach, highlight the usefulness of each component through extensive ablation studies, and conduct an in-depth analysis of the results, focusing on the effectiveness of each cognitive skill in the taxonomy, both by dataset and by model.

We draw conclusions both about our method and the capabilities of LLMs. Regarding our method, it achieves accuracy comparable to, and sometimes better than, the methods it was compared against. Specifically, the performance of BloomWise is similar to XoT and better than CoT and PoT, while the significantly higher accuracy in the Oracle setting highlights the method's potential. As for the LLMs, the method offers valuable insights into the cognitive skills each LLM demonstrates, as well as the skills required for solving various types of mathematical problems, thus enhancing interpretability. Some of the key observations are as follows: across all models, the highest accuracy was achieved at the "Analyzing" and "Understanding" stages, and, while difficult problems achieve better performance at higher taxonomy stages, the reverse does not hold true.

Keywords

LLMs, in-context learning, Bloom's Taxonomy, math problems, prompts

στην οικογένειά μου

Ευχαριστίες

Η περάτωση της διπλωματικής μου εργασίας σηματοδοτεί την ολοκλήρωση των προπτυχιακών μου σπουδών στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείου. Θα ήθελα να ευχαριστήσω όσους διαδραμάτισαν καθοριστικό ρόλο σε αυτό το ταξίδι.

Αρχικά, οφείλω ένα μεγάλο ¨Ευχαριστώ¨ στον επιβλέποντα καθηγητή μου, Αλέξανδρο Ποταμιάνο. Οι πολύτιμες συμβουλές και η αμέριστη στήριξη και εμπιστοσύνη την οποία μου έδειξε μου επέτρεψαν να γνωρίσω τον κόσμο της έρευνας, όπου κανείς καλείται όχι μόνο να κατανοήσει και να εφαρμόσει την υπάρχουσα γνώση, όπως συνήθως συμβαίνει σε ένα μάθημα, αλλά και να επεκτείνει τα όριά της.

Θα ήθελα, επίσης, να ευχαριστήσω τον μεταδιδακτορικό ερευνητή Γιώργο Παρασκευόπουλο, οι συμβουλές και η καθοδήγηση του οποίου υπήρξαν καταλυτικές για την πορεία και ολοκλήρωση αυτής της εργασίας.

Φυσικά, δεν θα μπορούσα να μην ευχαριστήσω τους φίλους και συνοδιπόρους μου σε αυτή την πορεία, η συναναστροφή με τους οποίους μου δίδαξε πολλά.

Τέλος, ευχαριστώ την οικογένειά μου για την αμέριστη στήριξή της κατά τη διάρκεια των σπουδών μου, χωρίς την οποία η ολοκλήρωση αυτής της πορείας θα ήταν αδύνατη.

Αθήνα, Δεκέβμριος 2024

Μαρία Ελένη Ζουμπουλίδη

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εκτεταμένη Ελληνική Περίληψη

1.1 Εισαγωγή

Τα μεγάλα γλωσσικά μοντέλα (LLMs) έχουν επιδείξει εντυπωσιακές επιδόσεις σε ένα ευρύ φάσμα εργασιών, όπως η παραγωγή κειμένου, η ανάλυση συναισθήματος, η ανάκτηση πληροφοριών και η δημιουργία κώδικα. Αυτές οι επιτυχίες έχουν καταστήσει τα LLMs εξαιρετικά εργαλεία σε τομείς όπως η αυτοματοποίηση υποστήριξης πελατών, η δημιουργική γραφή και η ανάλυση δεδομένων. Παρ' όλα αυτά, ένα βασικό πεδίο όπου τα LLMs εξακολουθούν να παρουσιάζουν αδυναμίες είναι η επίλυση μαθηματικών προβλημάτων. Η μαθηματική σκέψη απαιτεί ακρίβεια, δομημένη συλλογιστική και πολυεπίπεδη λογική, κάτι το οποίο τα LLMs συχνά δυσκολεύονται να επιτύχουν.

Ο Stefan Banach είχε κάποτε παρατηρήσει: «Τα μαθηματικά είναι το πιο όμορφο και πιο ισχυρό δημιούργημα της ανθρώπινης νόησης». Αυτή η πεποίθηση εξαίρει τον καθοριστικό χαρακτήρα της ενίσχυσης των μαθηματικών ικανοτήτων των LLMs. Καθώς διευρύνουμε τα όρια των δυνατοτήτων των μεγάλων γλωσσικών μοντέλων, η ικανότητά τους να κατανοούν και να χειρίζονται μαθηματικές έννοιες είναι εκ των ων ουκ άνευ για την προαγωγή της γνώσης και της καινοτομίας σε διάφορους τομείς.

Πολλές πρόσφατες προσεγγίσεις έχουν αξιοποιήσει τις δυνατότητες του in-context learning (ICL) [4] για να βελτιώσουν τις δεξιότητες επίλυσης προβλημάτων των LLMs. Κάποιες εκ των πιο διαδεδομένων τεχνικών περιλαμβάνουν την ενθάρρυνση των LLMs μέσω προτροπών να αναπτύξουν τη συλλογιστική τους πορεία σε κειμενική μορφή (Chain-of-Thought (CoT)) [5] ή να αξιοποιήσουν συναρτήσεις Python μέσω των Program-Aided Language Model (PAL, [6]) και Program-of-Thought prompting (PoT, [7]). Κάθε προσέγγιση έχει πλεονεκτήματα και μειονεκτήματα: Το Chain of Thought (CoT) κατακερματίζει τον συλλογισμό σε μια διαδοχική αφήγηση, επιτρέποντας ευέλικτες προσεγγίσεις επίλυσης προβλημάτων. Ωστόσο, υστερεί σε αριθμητική ακρίβεια ([5],[8]), καθώς τα γλωσσικά μοντέλα συχνά αντιμετωπίζουν δυσκολίες με τις μαθηματικές πράξεις. Από την άλλη πλευρά, τεχνικές όπως το PoT και το PAL χρησιμοποιούν κώδικα Python, αξιοποιώντας την ακρίβεια των διερμηνέων για αξιόπιστους υπολογισμούς, αλλά αδυνατούν να διαχειριστούν άγνωστες μεταβλητές. Πρόσφατες ερευνητικές προσπάθειες έχουν στραφεί προς την ενσωμάτωση πολλαπλών μεθοδολογιών, με στόχο την αναγνώριση της καταλληλότερης προσέγγισης για το εκάστοτε πρόβλημα, συγκεράζοντας τα πλεονεκτήματα διαφόρων τεχνικών. Η μέθοδος X of Thoughts [3] είναι μια τεχνική προτροπής (prompting) κατά την οποία επιλέγεται, εφαρμόζεται και επαληθεύεται η καταλληλότερη εκ των τεχνικών CoT (Chain of Thought), PoT (Program of Thought) και EoT (Equations of Thought) (επαναληπτικά, έως ότου βρεθεί η σωστή λύση), με στόχο τη βελτίωση της απόδοσης των LLMs στην επίλυση μαθηματικών προβλημάτων.

Βασιζόμενοι στην ιδέα της ενσωμάτωσης πολλαπλών μεθόδων και επιδιώκοντας την ενίσχυση της αυτογνωσίας των LLMs, προτείνουμε το BloomWise, μια νέα, cognitively-

motivated μέθοδο prompting. Το BloomWise ενθαρρύνει τα LLMs να επιστρατεύουν σταδιακά ανώτερες γνωστικές λειτουργίες σε ιεραρχική σειρά για να επιλύσουν ένα πρόβλημα, έως ότου βρεθεί η σωστή λύση. Ενθαρρύνοντας την ενεργοποίηση γνωστικών διαδικασιών, το BloomWise εμβαθύνει στη σφαίρα των γνωστικών ικανοτήτων, ευθυγραμμίζοντας τις στρατηγικές επίλυσης προβλημάτων με την Ταξονομία του Bloom, προκειμένου να βελτιώσει την κατανόηση και την ακρίβεια στην αντιμετώπιση σύνθετων μαθηματικών προκλήσεων. Με τη χρήση πολυεπίπεδων γνωστικών προτροπών (multi-level cognitive prompting) στοχεύουμε στην ενίσχυση τόσο της ερμηνευσιμότητας όσο και της ακρίβειας των αποτελεσμάτων.

Η κύρια ιδέα πίσω από το BloomWise είναι να ενθαρρύνει το LLM να φτάσει σε μια λύση αξιοποιώντας γνωστικές δεξιότητες αυξανόμενης πολυπλοκότητας. Η απάντηση η οποία αντιστοιχεί σε κάθε ένα από τα πρώτα πέντε επίπεδα της Ταξονομίας του Bloom αυτοαξιολογείται από το LLM, το οποίο ανακοινώνει επιτυχία εάν η λύση θεωρείται σωστή, αποτρέποντας τη μετάβαση στο επόμενο επίπεδο της ταξονομίας. Προκειμένου να καταδείξουμε την αποτελεσματικότητα της προσέγγισής μας, διεξάγάγαμε εκτενή πειράματα σε τέσσερα δημοφιλή σύνολα δεδομένων μαθηματικών προβλημάτων και επιτυγχάνουμε σταθερές βελτιώσεις. Επιπλέον, εξερευνήσαμε διάφορες παραλλαγές της μεθόδου μας, όπως το majority voting μεταξύ των επιπέδων και το Program of Bloom, το οποίο συνδυάζει το BloomWise με το PoT. Επίσης, αξιολογήσαμε τη δυναμική της προτεινόμενης μεθόδου χρησιμοποιώντας ένα Oracle ως κριτήριο για την απόφαση για μετάβαση ή μη στο επόμενο επίπεδο της ταξονομίας. Τα κύρια σημεία συνεισφοράς της μελέτης είναι τα εξής:

1.Προτείνουμε μια νέα μέθοδο prompting για την επίλυση μαθηματικών (και πιθανώς άλλων) προβλημάτων, βασισμένη στην Ταξονομία του Bloom.

2.Ενσωματώνουμε την ιδέα του Πρόωρου Τερματισμού (early stopping) στο prompting, καθώς η εκτέλεση της μεθόδου σταματά όταν η λύση η οποία προκύπτει σε ένα επίπεδο της ταξονομίας αξιολογηθεί ως σωστή.

3.Διερευνούμε την απόδοση διαφόρων LLMs σε κάθε γνωστικό επίπεδο της Ταξονομίας του Bloom για τέσσερα δημοφιλή σύνολα δεδομένων μαθηματικών προβλημάτων. Τα αποτελέσματά μας προσφέρουν πολύτιμες πληροφορίες σχετικά με τις γνωστικές δεξιότητες που επιδεικνύει κάθε LLM, καθώς και τις δεξιότητες που απαιτούνται για την επίλυση διαφορετικών τύπων μαθηματικών προβλημάτων.

4.Επιπλέον, καταδεικνύουμε τη δυναμική του cognitively-motivated prompting για τη βελτίωση της ακρίβειας και την ενίσχυση της ερμηνευσιμότητας.

1.2 Θεωρητικό Υπόβαθρο

1.2.1 Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

Παρά το γεγονός πως δεν υπάρχει απόλυτα σαφής και κοινά αποδεκτός ορισμός, θα μπορούσαμε να πούμε πως η Τεχνητή Νοημοσύνη αποτελεί τον κλάδο της Επιστήμης Υπολογιστών ο οποίος ασχολείται με την αυτοματοποίηση της νοήμονος συμπεριφοράς. Η Μηχανική Μάθηση, η οποία συνιστά κλάδο της Τεχνητής Νοημοσύνης, συνίσταται στις αρχές, μεθόδους και αλγορίθμους για μάθηση και πρόβλεψη με βάση προηγούμενα δεδομένα με στόχο οι μηχανές να εργάζονται αξιοποιώντας την εμπειρία και όχι σαφώς

καθορισμένες οδηγίες. Οι εργασίες μηχανικής μάθησης διακρίνονται σε τρεις κύριες κατηγορίες, με βάση τον τρόπο εκπαίδευσης ή τη διαθέσιμη ανατροφοδότηση. Στην επιβλεπόμενη μάθηση (supervised learning), το σύστημα εκπαιδεύεται δεχόμενο εισόδους και τα αντίστοιχα επιθυμητά αποτελέσματα, με στόχο την εκμάθηση ενός γενικού κανόνα για τη σύνδεση εισόδων και εξόδων. Στην κατηγορία αυτή εμπίπτουν η ταξινόμηση (classification) και η παλινδρόμηση (reggression). Η ταξινόμηση είναι η διαδικασία κατάταξης των δεδομένων σε διακριτές κατηγορίες (π.χ., ταξινόμηση email ως «spam» ή «not spam»), ενώ η παλινδρόμηση είναι η πρόβλεψη συνεχών τιμών (π.χ., πρόβλεψη της τιμής ενός αυτοκινήτου βάσει του έτους κατασκεύης, της κατάστασης του οχήματος, της απόστασης που έχει διανύσει κλπ). Στη μη επιβλεπόμενη μάθηση (unsupervised learning), το σύστημα καλείται να ανακαλύψει δομή και πρότυπα στα δεδομένα χωρίς τη χρήση των σωστών ετικετών. Χαρακτηριστικά παραδείγματα αποτελούν η Συσταδοποίηση (clustering) και η Μείωση διαστάσεων (dimensionality reduction). Η συσταδοποίηση είναι η ομαδοποίηση δεδομένων που παρουσιάζουν ομοιότητες (π.χ., κατηγοριοποίηση ταινιών), ενώ η μείωση διαστάσεων είναι η συμπύκνωση της πληροφορίας μειώνοντας τις διαστάσεις των δεδομένων (π.χ., με τη χρήση τεχνικών όπως η PCA) για πιο εύκολη επεξεργασία ή οπτικοποίηση. Στην ημι-επιβλεπόμενη μάθηση (semi-supervised learning), μόνο ένα υποσύνολο των δεδομένων διαθέτει επισημασμένα αποτελέσματα. Τέλος, η ενισχυτική μάθηση (reinforcement learning) βασίζεται στην αλληλεπίδραση με ένα δυναμικό περιβάλλον όπου το σύστημα λαμβάνει ανατροφοδότηση μέσω επιβραβεύσεων, επιδιώκοντας την επίτευξη ενός συγκεκριμένου στόχου. Σήμερα, το ερευνητικό ενδιαφέρον επικεντρώνεται στη Βαθιά Μάθηση (Deep Learning), υποκατηγορία της Μηχανικής Μάθησης, βάση της οποίας συνιστούν τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANNs). Όπως καθίσταται σαφές από το όνομά τους, η δομή τους είναι εμπνευσμένη από τους βιολογικούς νευρώνες του ανθρώπινου εγκεφάλου. Παραδοσιακά, τα ANNs αποτελούνται από κόμβους (νευρώνες) που οργανώνονται σε στρώσεις. Συγκεκριμένα, τα ANNs περιλαμβάνουν μια είσοδο και μια έξοδο, καθώς και μία ή περισσότερες κρυφές στρώσεις ενδιάμεσα. Κάθε κόμβος μπορεί να συνδέεται με έναν ή περισσότερους άλλους κόμβους, και κάθε σύνδεση συνοδεύεται από ένα βάρος. Για να προκύψει η έξοδος κάθε κόμβου, υπολογίζεται το άθροισμα των εξόδων των κόμβων των προηγούμενων στρώσεων με τους οποίους είναι συνδεδεμένος. Πρόσφατες εξελίξεις στη Βαθιά Μάθηση περιλαμβάνουν τεχνικές αιχμής όπως ο μηχανισμός προσοχής (attention mechanism) και οι μετασχηματιστές (transformers). Ο μηχανισμός προσοχής επιτρέπει στα μοντέλα να εστιάζουν επιλεκτικά σε καίριες περιοχές των δεδομένων εισόδου, βελτιστοποιώντας τη διαδικασία κατανόησης και γενίκευσης. Οι μετασχηματιστές, βασιζόμενοι στον μηχανισμό προσοχής, έχουν επιφέρει σημαντική πρόοδο, ιδιαιτέρως στον τομέα της Επεξεργασίας Φυσικής Γλώσσας (NLP), προσφέροντας εξαιρετική απόδοση στην ανάλυση και παραγωγή κειμένου. Αυτές οι τεχνικές έχουν υπερκεράσει τις παραδοσιακές προσεγγίσεις νευρωνικών δικτύων, παρέχοντας αυξημένη ακρίβεια και αποδοτικότητα σε πλήθος εφαρμογών.

1.2.2 Μεγάλα Γλωσσικά Μοντέλα

Τα γλωσσικά μοντέλα είναι μοντέλα τα οποία αναθέτουν πιθανότητες σε ακολουθίες λέξεων, δηλαδή εκτιμούν την πιθανότητα εμφάνισης μιας λέξης βάσει των συμφραζομένων [9]. Αποτελούν ακρογωνιαίο λίθο της Επεξεργασίας Φυσικής Γλώσσας (NLP), καθώς παρέχουν έναν τρόπο μετατροπής ποιοτικών πληροφοριών κειμένου σε ποσοτικά δεδομένα που είναι κατανοητά από τη μηχανή.

Ιστορικά, τα πρώτα γλωσσικά μοντέλα βασίζονταν σε στατιστικές μεθόδους. Παραδείγματος χάριν, τα n-grams και τα μοντέλα κρυφών Μαρκοβιανών αλυσίδων χρησιμοποιούσαν τις συχνότητες των λέξεων και φράσεων για να εκτιμήσουν τις πιθανότητες. Στις πιο σύγχρονες προσεγγίσεις, χρησιμοποιούνται βαθιά νευρωνικά δίκτυα, με τα Μεγάλα Γλωσσικά Μοντέλα (LLMs) να έχουν φέρει επανάσταση στην επεξεργασία και κατανόηση της φυσικής γλώσσας.

Τα LLMs, όπως τα GPT, Llama και Mixtral, είναι εκπαιδευμένα σε τεράστια σύνολα δεδομένων και αξιοποιούν εξελιγμένες τεχνικές, όπως οι μετασχηματιστές (transformers), για να επιτύχουν υψηλές επιδόσεις σε μια πληθώρα γλωσσικών εργασιών.

1.2.3 Μάθηση βάσει prompts

Η παραδοσιακή μέθοδος για την προσαρμογή των προ-εκπαιδευμένων γλωσσικών μοντέλων σε επιμέρους προβλήματα είναι το fine-tuning. Ωστόσο, αυτό δεν αποτελεί πάντα τη βέλτιστη προσέγγιση: είναι πιθανό να οδηγήσει σε υπερπροσαρμογή του προ-εκπαιδευμένου μοντέλου, ενώ η τροποποίηση των παραμέτρων του μπορεί να προκαλέσει απώλεια σημαντικών γλωσσικών σχέσεων και αναπαραστάσεων τις οποίες το μοντέλο απέκτησε κατά την προ-εκπαίδευση. Επιπλέον, το συνεχώς αυξανόμενο μέγεθος των σύγχρονων προ-εκπαιδευμένων μοντέλων καθιστά το fine-tuning πιο δαπανηρό και μη αποδοτική από άποψη πόρων επιλογή. Για αυτούς τους λόγους, τα τελευταία χρόνια η ερευνητική κοινότητα έχει στραφεί σε εναλλακτικές λύσεις για την προσαρμογή των προ-εκπαιδευμένων γλωσσικών μοντέλων.

Θα εστιάσουμε στη μάθηση βάσει prompt. Η μάθηση βάσει prompt διατηρεί το προ-εκπαιδευμένο μοντέλο αμετάβλητο και εισάγει έναν μικρό αριθμό παραμέτρων, που αποτελούν το λεγόμενο prompt, είτε στο επίπεδο της κειμενικής εισόδου είτε απευθείας στο επίπεδο της ενσωμάτωσης (embedding) του μοντέλου. Καθώς το μέγεθος των σύγχρονων προ-εκπαιδευμένων γλωσσικών μοντέλων αυξάνεται διαρκώς, η μάθηση βάσει prompt ερευνάται κυρίως λόγω των χαμηλότερων απαιτήσεών της σε υπολογιστικούς πόρους σε σύγκριση με την παραδοσιακή μέθοδο του fine-tuning.

Υπάρχουν διάφοροι τύποι prompts που χρησιμοποιούνται για την καθοδήγηση των μοντέλων. Στο zero-shot learning, το μοντέλο καλείται να εκτελέσει μια εργασία χωρίς να του δοθεί κάποιο παράδειγμα της εργασίας, απλώς βασιζόμενο στη διατύπωση της εντολής. Αντίθετα, στο few-shot learning, το μοντέλο λαμβάνει λίγα παραδείγματα της εργασίας μέσα στο prompt, ώστε να προσαρμοστεί καλύτερα.

1.2.4 Ταξονομία Bloom

Η Ταξονομία του Bloom είναι ένα πολυεπίπεδο μοντέλο ταξινόμησης της σκέψης σύμφωνα με έξι γνωστικά επίπεδα πολυπλοκότητας. Στην αρχική εκδοχή της Ταξονομίας, τα τρία χαμηλότερα επίπεδα είναι: η απομνημόνευση, η κατανόηση και η εφαρμογή. Τα τρία υψηλότερα επίπεδα είναι: η ανάλυση, η σύνθεση και η αξιολόγηση. Η ταξονομία είναι ιεραρχική και κάθε επίπεδο συμπεριλαμβάνει τα χαμηλότερα επίπεδα. Κατά τη διάρκεια της δεκαετίας του 1990, η ταξονομία αναθεωρήθηκε. Η νέα δομή της ταξονομίας είναι: η απομνημόνευση, η κατανόηση, η εφαρμογή, η ανάλυση, η αξιολόγηση και η δημιουργία. Η εργασία μας βασίζεται στην αναθεωρημένη ταξονομία των [10]. Τα βήματα που χρησιμοποιούνται στην Ταξονομία ορίζονται ως εξής:

1. **Απομνημόνευση:** Η ανάκτηση, η αναγνώριση και η ανάκληση σχετικών γνώσεων από τη μακροπρόθεσμη μνήμη.

2. **Κατανόηση:** Η κατασκευή νοήματος από προφορικά, γραπτά και γραφικά μηνύματα μέσω της ερμηνείας, της παροχής παραδειγμάτων, της ταξινόμησης, της περίληψης, της εξαγωγής συμπερασμάτων, της σύγκρισης και της εξήγησης.

3. **Εφαρμογή:** Η εκτέλεση ή η χρήση μιας διαδικασίας.

4. **Ανάλυση:** Η διάσπαση του υλικού σε επιμέρους μέρη και ο καθορισμός του τρόπου με τον οποίο τα μέρη σχετίζονται μεταξύ τους και με τη συνολική δομή ή σκοπό μέσω της διάκρισης, της οργάνωσης και της απόδοσης αιτιών.

5. **Αξιολόγηση:** Η διατύπωση κρίσεων βασισμένων σε κριτήρια και πρότυπα μέσω του ελέγχου και της κριτικής.

6. **Δημιουργία:** Η σύνθεση στοιχείων για τη δημιουργία ενός συνεκτικού ή λειτουργικού συνόλου· η αναδιοργάνωση στοιχείων σε ένα νέο μοτίβο ή δομή μέσω της δημιουργίας, του σχεδιασμού ή της παραγωγής.

Η αναθεωρημένη ταξονομία φαίνεται στο σχήμα 1.1.

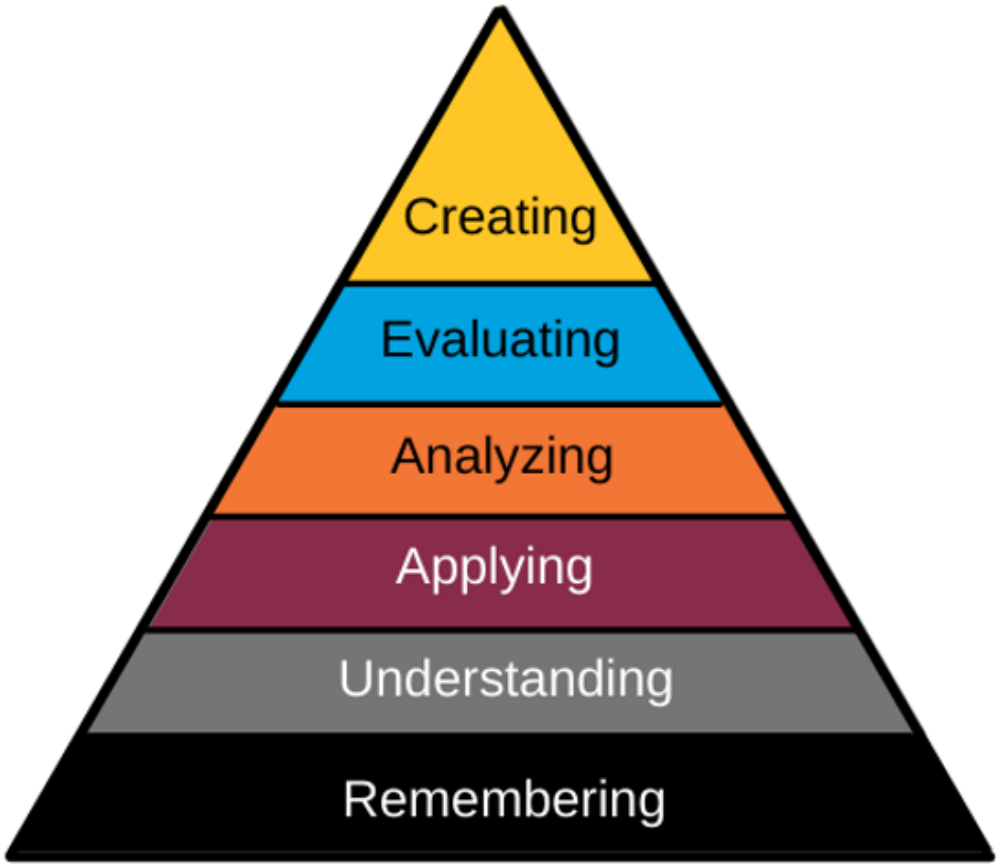
1.3 Ορισμός του προβλήματος

Δεδομένου ενός συνόλου μαθηματικών προβλημάτων $P = \{p_1, p_2, \ldots, p_N\}$ με αντίστοιχες λύσεις $S = \{s_1, s_2, \ldots, s_N\}$, το πρόβλημα της Επίλυσης Μαθηματικών Προβλημάτων αφορά την ανάπτυξη ή χρήση μιας υπολογιστικής μεθοδολογίας που μπορεί να ερμηνεύσει κάθε πρόβλημα p_i, να εφαρμόσει την κατάλληλη συλλογιστική και μαθηματικές πράξεις, και να παραγάγει τη σωστή λύση s_i.

1.4 Σχετική βιβλιογραφία

Συλλογισμός με Μεγάλα Γλωσσικά Μοντέλα

Καθώς ο τομέας των Μεγάλων Γλωσσικών Μοντέλων αναπτύσσεται ραγδαία, έχουν αναπτυχθεί διάφορες στρατηγικές προτροπών (prompting strategies) για την ενίσχυση των



Σχήμα 1.1: *Αναθεωρημένη ταξονομία Bloom*

συλλογιστικών τους ικανοτήτων. Κάποιες από τις πιο αποτελεσματικές τεχνικές αποτελούν η Chain of thought, η οποία ενθαρρύνει το LLM να χρησιμοποιήσει ενδιάμεσα βήματα συλλογισμού επιτρέποντας έτσι πολύπλοκες δυνατότητες συλλογισμού [5] αλλά και τεχνικές οι οποίες αξιοποιούν τον προγραμματισμό για την αντιμετώπιση διαδικαστικών προκλήσεων [7, 6] (Program of Thought-PoT, PAL). Επίσης, χρησιμοποιείται ευρέως zero-shot, το οποίο απαιτεί μόνο μία πρόταση για να καθοδηγηθούν τα LLMs [11]. Επιπλέον, η προσέγγιση Tree-of-Thoughts [12] εξετάζει διάφορες συλλογιστικές πορείες και διατρέχει δενδροειδείς δομές καταστάσεων σκέψης. Ακόμα, η X of Thoughts [3] επιλέγει, εφαρμόζει και επαληθεύει την πιο κατάλληλη εκ των τεχνικών CoT , PoT και EoT (Equations of Thought) επαναληπτικά, έως ότου βρεθεί η σωστή λύση.

Μεταγνωστική Προτροπή

Ο όρος «Μεταγνώση» χρησιμοποιείται για να δηλώσει τη γνώση για τη γνώση ή, πιο απλά, τη σκέψη για τη σκέψη. Ο Αμερικανός αναπτυξιακός ψυχολόγος [13] όρισε τη μεταγνώση ως «γνώση για τη σκέψη και έλεγχο της σκέψης».

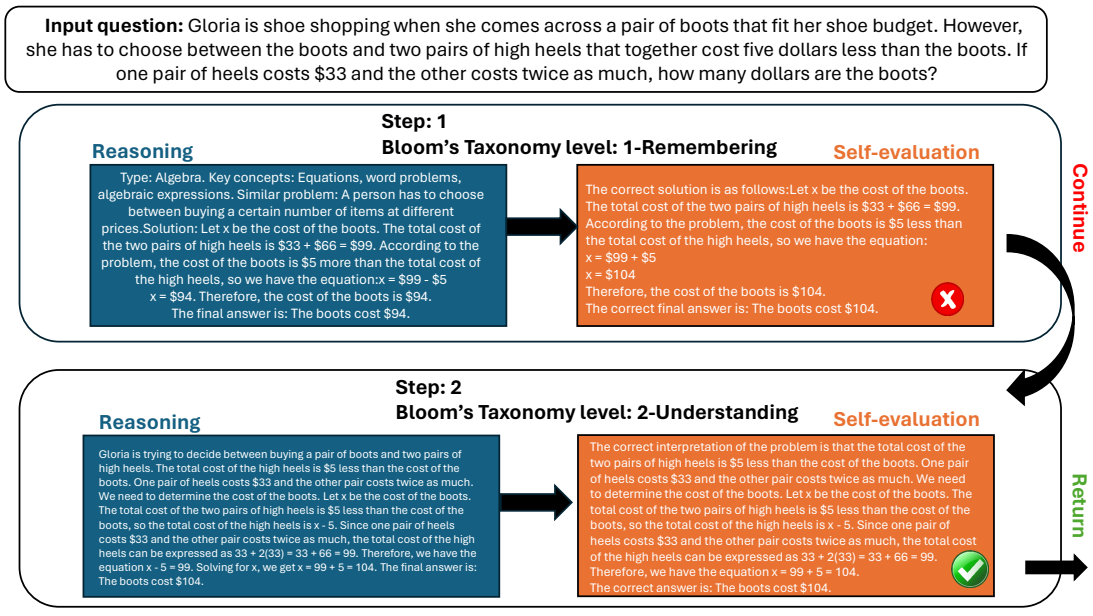
Αυτή η έννοια επεκτάθηκε στον τομέα του instruction tuning στα μεγάλα γλωσσικά μοντέλα. Οι [14] εισήγαγαν τα Μεταγνωστικά prompts, μια καινοτόμο στρατηγική βασισμένη

στον ανθρώπινο συλλογισμό. Η μέθοδος εφαρμόστηκε σε εργασίες Κατανόησης Φυσικής Γλώσσας.

1.5 Η μέθοδός μας

1.5.1 Το framework

Στόχος μας είναι η ανάπτυξη ενός γενικευμένου πλαισίου επίλυσης προβλημάτων το οποίο αξιοποιεί prompts καθένα από τα οποία αντιστοιχεί σε ένα επίπεδο της ταξονομίας του Bloom, έως ότου επιτευχθεί η σωστή λύση. Σε αυτή την ενότητα, περιγράφουμε τη συνολική δομή του πλαισίου και παρουσιάζουμε αναλυτικά κάθε επιμέρους τμήμα.



Σχήμα 1.2: *Σύνοψη του αλγορίθμου BloomWise Early Stop. Το μαθηματικό πρόβλημα εμφανίζεται στην κορυφή, ακολουθούμενο από την έξοδο σε κάθε επίπεδο της ταξονομίας του Bloom (μπλε πλαίσιο στα αριστερά) και την έξοδο για το βήμα της αυτοαξιολόγησης (πορτοκαλί πλαίσιο στα δεξιά). Οι προτροπές δεν εμφανίζονται, παρακαλούμε ανατρέξτε στο Παράρτημα B . Η αυτοαξιολόγηση αποτυγχάνει στο επίπεδο 1 «Ανάκληση» και ο αλγόριθμος προχωρά στο επίπεδο 2 «Κατανόηση», όπου η αυτοαξιολόγηση πετυχαίνει. Κατά συνέπεια, το LLM επιστρέφει την απάντηση από το επίπεδο 2.*

BloomWise Framework

Η συνολική διαδικασία περιγράφεται στον Αλγόριθμο 1.1.

ΚΩΔΙΚΑΣ 1.1: *BloomWise Early Stop*

```
Input: i
level ← 1
for l in levels do
    res ← Reasoning(level_prompt, i)
    eval ← SelfEvaluate(res)
    if eval = false and level ≤ 5 then
        level ← level + 1
    else
        break
    end if
end for
return Output
```

Prompts

Σχεδιάσαμε προτροπές (prompts) οι οποίες αντιστοιχούν στα πρώτα πέντε επίπεδα της Ταξονομίας του Bloom [1]. Οι αναλυτικές προτροπές παρουσιάζονται στους Πίνακες Β.1-Β.5 στο Παράρτημα Β [2].

Ανάκληση: διάβασε προσεκτικά το πρόβλημα, εντόπισε τον τύπο και τις κύριες έννοιές του, θυμήσου ένα παρόμοιο ή το ίδιο πρόβλημα και απάντησε.

Κατανόηση: συνόψισε το πρόβλημα και τις σχετικές έννοιές του και προσπάθησε να λύσεις το πρόβλημα με βάση αυτά.

Εφαρμογή: σχεδίασε και εφάρμοσε μια μεθοδολογία βήμα προς βήμα για να φτάσεις στη λύση.

Ανάλυση: βρες μια λύση αφού αναλύσεις τις σκέψεις σου για τον τρόπο με τον οποίο πρέπει να επιλυθεί το πρόβλημα.

Αξιολόγηση: κάνε κριτική αξιολόγηση της λύσης κατά την έκφρασή της.

Αυτοαξιολόγηση

Στη φάση της επαλήθευσης, αξιολογείται η αποτελεσματικότητα της λύσης μέσω αυτοαξιολόγησης: Το LLM λαμβάνει το πρόβλημα και μια πιθανή λύση την οποία έχει δημιουργήσει, και καλείται να αναπαραγάγει τη λύση ενώ ελέγχει για σφάλματα. Στην προτροπή, ενημερώσαμε το LLM ότι η λύση πιθανώς είναι λανθασμένη. Υιοθετείται μια αντιπαραθετική προσέγγιση για να αυξηθούν οι πιθανότητες εντοπισμού σφαλμάτων. Έχει παρατηρηθεί ότι η απαίτηση αναδιατύπωσης βοηθά τα μικρότερα LLM να πραγματοποιήσουν μια πιο ακριβή αξιολόγηση. Η προτροπή αυτοαξιολόγησης και ένα παράδειγμα απάντησης παρουσιάζονται στον Πίνακα Β.6. Περισσότερα παραδείγματα αυτοαξιολόγησης παρατίθενται στο Σχήμα 1.2.

Επανάληψη και Πρόωρος Τερματισμός

Για ένα δεδομένο πρόβλημα, το LLM δέχεται προτροπές οι οποίες είναι δομημένες σύμφωνα με το αντίστοιχο επίπεδο της ταξονομίας του Bloom. Η διαδικασία αυτή συνεχίζεται

[1] Το έκτο επίπεδο της ταξονομίας-'Δημιουργία¨-ίσως θα ήταν χρήσιμο για την επίλυση άλλου είδους προβλημάτων. Θεωρήθηκε λιγότερο κατάλληλο για μαθηματικά προβλήματα.

[2] Σημειώστε ότι οι προτροπές έχουν σχεδιαστεί σκόπιμα για να είναι πολύ βασικές και απλές, προκειμένου να εκτιμηθεί η δυναμική της μεθόδου· δεν χρησιμοποιήθηκαν τεχνικές βελτίωσης προτροπών (prompt engineering).

μέχρι να εντοπιστεί μια σωστή λύση ή να έχουν δοκιμαστεί όλα τα επίπεδα. Αυτή η πρακτική βασίζεται στην υπόθεση ότι ένα υψηλότερο επίπεδο της ταξονομίας ενθαρρύνει το LLM να προσεγγίσει το πρόβλημα με πιο εμβριθή τρόπο σε σύγκριση με ένα χαμηλότερο επίπεδο. Επιπλέον, εάν η προτροπή που αντιστοιχεί σε ένα συγκεκριμένο επίπεδο οδηγήσει σε σωστή λύση, δεν είναι απαραίτητη η περαιτέρω επεξεργασία για το συγκεκριμένο πρόβλημα και, συνεπώς, ο πρώορος τερματισμός είναι εύλογος. Αυτή η διαδικασία περιγράφεται στο Σχήμα 1.2.

Πρόωρος Τερματισμός με τη Χρήση ενός oracle

Υπάρχει πληθώρα τρόπων επιλογής του επιπέδου της ταξονομίας στο οποίο θα τερματιστεί η διαδικασία. Προκειμένου να εξερευνήσουμε πλήρως τις δυνατότητες της μεθόδου μας, εξετάσαμε επίσης τη χρήση ενός oracle για να αποφασιστεί το βέλτιστο σημείο διακοπής. Δεδομένης μιας ερώτησης q, ορίζουμε την ορθότητα των απαντήσεων οι οποίες προκύπτουν με κάθε προτροπή ως $\hat{R}_s(q) \in \{0, 1\}$ σε σύγκριση με την σωστή απάντηση (gold label). Ο δείκτης $s \in \mathcal{S}$ αντιστοιχεί στο στάδιο της ταξονομίας του Bloom το οποίο χρησιμοποιείται για την απάντηση της ερώτησης q, δηλαδή ανάκληση, κατανόηση, εφαρμογή, ανάλυση και αξιολόγηση. Ορίζουμε την ακρίβεια στην περίπτωση του oracle για N ερωτήσεις ως:

$$Acc_{oracle} = \frac{1}{N} \sum_{q} \hat{R}(q), \tag{1.1}$$

όπου

$$\hat{R}(q) = \bigvee_{s \in \mathcal{S}} \hat{R}_s(q).$$

Στο BloomWise oracle ένα πρόβλημα θεωρείται λυμένο ορθά αν κάποια από τις μεθόδους οι οποίες αντιστοιχούν στα επίπεδα της ταξονομίας παράγει τη σωστή απάντηση.

BloomWise Majority Voting

Ως εναλλακτική στον πρόωρο τερματισμό, εξετάσαμε επίσης μια προσέγγιση κατά την οποία την τελική έξοδο αποτελεί το πλειοψηφούν εκ των αποτελεσμάτων τα οποία αντιστοιχούν στα πέντε πρώτα στάδια της ταξονομίας του Bloom. Για μια δεδομένη ερώτηση q, κάθε prompt $s \in \mathcal{S}$ το οποίο αντιστοιχεί σε ένα επίπεδο της ταξονομίας του Bloom οδηγεί σε ένα αριθμητικό αποτέλεσμα $R_s(q)$. Η τελική απάντηση καθορίζεται με βάση το πιο συχνό αποτέλεσμα μεταξύ αυτών των σταδίων. $\hat{R}_{\text{majority}}(q) \in \{0, 1\}$ σε σύγκριση με την σωστή ετικέτα.

Για το majority voting, ορίζουμε την ακρίβεια για N αξιοποιώντας την εξίσωση (1.1) όπου:

$$\hat{R}(q) = \hat{R}_{\text{majority}}(q).$$

Σε περιπτώσεις ισοψηφίας (δηλαδή, όταν η μεγαλύτερη συχνότητα αντιστοιχεί σε δύο ή περισσότερες απαντήσεις), επιλέγεται η πρώτη απάντηση που εμφανίζεται μεταξύ αυτών με τη μεγαλύτερη συχνότητα.

1.5.2 Πειραματική διάταξη

Σύνολα Δεδομένων Τα πειράματά μας διεξήχθησαν σε ένα πλήρες σύνολο τεσσάρων συνόλων δεδομένων μαθηματικών προβλημάτων τα οποία καλύπτουν διάφορα απαιτητικά σενάρια μαθηματικής συλλογιστικής: GSM8K, SVAMP, Algebra και GSM-hard (αυτό το

Σύνολο δεδομένων	Πλήθος Δειγμάτων
GSM8K [15]	1,319
SVAMP [16]	1,000
Algebra [17]	222
GSM-hard [6]	1,319

Πίνακας 1.1: *Στατιστικά των συνόλων δεδομένων που χρησιμοποιήσαμε*

σύνολο δεδομένων περιλαμβάνει προβλήματα από το σύνολο GSM8K με αντικατάσταση μικρών αριθμητικών τιμών από μεγαλύτερες με στόχο την αύξηση της πολυπλοκότητας των υπολογισμών). Οι λεπτομέρειες σχετικά με τα σύνολα δεδομένων δίνονται στον Πίνακα. 1.1.

Μοντέλα Αξιοποιήσαμε το OpenAI API για τα πειράματά μας [3] . Συγκεκριμένα, χρησιμοποιούμε το gpt-3.5-turbo τόσο για την παραγωγή της λύσης όσο και για την αυτοαξιολόγηση. Πραγματοποιούμε επίσης πειράματα με ένα ευρύ φάσμα μοντέλων διαφορετικών μεγεθών, συμπεριλαμβανομένου και mixture of experts. Τα μοντέλα είναι τα εξής: Llama2 (13B και 70B) [18] και Mixtral 8x7B instruct [19].

1.5.3 Πειράματα και Αποτελέσματα

Σύγκριση με σύγχρονες μεθόδους

Στον Πίνακα 1.2, παρουσιάζουμε την ακρίβεια η οποία επετεύχθη σε καθένα εκ των τεσσάρων συνόλων δεδομένων για τα τέσσερα LLMs με τα οποία πειραματιστήκαμε. Θεωρούμε τρεις μεθόδους προτροπών ως μεθόδους αναφοράς, συγκεκριμένα τις CoT [5], PoT [7] και XoT [3], και τις συγκρίνουμε με τις BloomWise Early Stop (BLES), Majority Voting (BLM) και Oracle (BLO). Συνολικά, κατά μέσο όρο σε όλα τα σύνολα δεδομένων και τα μοντέλα, οι BLM και XoT έχουν παρόμοια απόδοση με ακρίβεια περίπου 56.5%, ακολουθούμενες από τις CoT και BLES με 50.6% και 49.1%, αντίστοιχα, ενώ η PoT καταγράφει τη χειρότερη απόδοση με 40.2%. Για το Oracle, η ακρίβεια της BLO είναι σημαντικά υψηλότερη, φτάνοντας το 71%, γεγονός το οποίο καταδεικνύει τη δυναμική της BloomWise.

Απόδοση ανά Σύνολο Δεδομένων: Το BLM επιτυγχάνει τη μεγαλύτερη ακρίβεια για τα GSM8K και SVAMP, ενώ για το Algebra τα XoT και BLM έχουν παρεμφερή απόδοση. Συγκεκριμένα, όσον αφορά στο SVAMP, η μέση ακρίβεια του BLM για τα 4 μοντέλα είναι 76.4% ενώ του XoT 69.5%. Όσον αφορά στο GSM8K το BLM επιτυγχάνει 63% έναντι 59.4% του XoT. Και οι τρεις παραλλαγές του BloomWise επιτυγχάνουν κακή απόδοση για το GSM-hard. Η συμπεριφορά αυτή είναι εύλογη λόγω των πολύπλοκων υπολογισμών οι οποίοι χαρακτηρίζουν το συγκεκριμένο σύνολο δεδομένων, η καλύτερη επίδοση στο οποίο καταγράφεται για το XoT ακολουθούμενο από το PoT- λόγω της χρήσης Python. Το BLO μειώνει το σχετικό ποσοστό σφάλματος σε σύγκριση με την καλύτερη μέθοδο (BLM ή XoT) έως και κατά 53% για το SVAMP, 47% για το GSM8K, 30% για το Algebra, χωρίς καμία βελτίωση για το GSM-hard.

[3] https://openai.com

	Σύνολο Δεδομένων	CoT	PoT	XoT	BloomWise EarlyStop (BLES)	BloomWise Majority Voting (BLM)	BloomWise Oracle (BLO)
GPT 3.5 turbo	GSM8K	80.2	77.2	**83.3**	78.2	82.6	92.5
	SVAMP	79.5	79.5	83.6	84.7	**88.0**	94.8
	Algebra	81.5	62.5	**89.9**	85.1	86.0	92.3
	GSM-hard	42.4	61.8	**63.4**	40.6	44.4	58.0
LLaMA2-70B	GSM8K	**59.1**	52.3	57.9	38.7	56.4	76.8
	SVAMP	75.2	73.6	**77.0**	55.9	76.2	90.0
	Algebra	43.2	35.6	**64.0**	28.4	45.5	67.5
	GSM-hard	27.1	43.1	**43.8**	13.1	20.2	32.7
LLaMA2-13B	GSM8K	27.5	26.4	30.2	29.7	**37.6**	59.9
	SVAMP	40.3	50.4	46.2	43.4	**55.2**	78.6
	Algebra	20.3	26.1	**30.2**	25.2	29.3	53.2
	GSM-hard	8.5	**22.7**	17.4	7.2	9.8	19.7
Mixtral 8x7B Instruct	GSM8K	66.8	13.9	66.0	69.4	**75.4**	88.7
	SVAMP	72.2	6.8	71.2	82.4	**86.2**	92.7
	Algebra	54.9	2.3	54.0	71.1	**71.6**	86.9
	GSM-hard	31.5	8.5	31.4	33.2	**37.7**	50.9

Πίνακας 1.2: *Η ακρίβεια της λύσης σε διάφορα μοντέλα και σύνολα δεδομένων. Τα αποτελέσματα των μεθόδων αναφοράς για τα Llama2 και GPT 3.5 turbo προέρχονται από [3]. Η πιο ακριβής μέθοδος εμφανίζεται με έντονη γραφή και η δεύτερη πιο ακριβής είναι υπογραμμισμένη (εξαιρούμε το BloomWise Oracle).*

Απόδοση ανά μοντέλο: Η καλύτερη μέθοδος εξαρτάται σε μεγάλο βαθμό από το μοντέλο. Το BLM συνιστά την καλύτερη επιλογή για τα LLaMA2-13B και mixtral, ενώ το XoT είναι η κορυφαία επιλογή για τα LLaMA2-70B και gpt3.5-turbo. Το BLES αποδίδει καλά για το gpt3.5-turbo και το mixtral, αλλά σχετικά άσχημα για τα μοντέλα LLaMA2 (βλ. επίσης την επόμενη ενότητα για μια εμβριθή ανάλυση της απόδοσης με πρόωρο τερματισμό). Το BLO μειώνει το ποσοστό σφάλματος σε σύγκριση με την καλύτερη μέθοδο προτροπής για όλα τα μοντέλα, το οποίο κυμαίνεται κατά μέσο όρο μεταξύ 15% για το LLaMA2-70B και 37% για το mixtral.

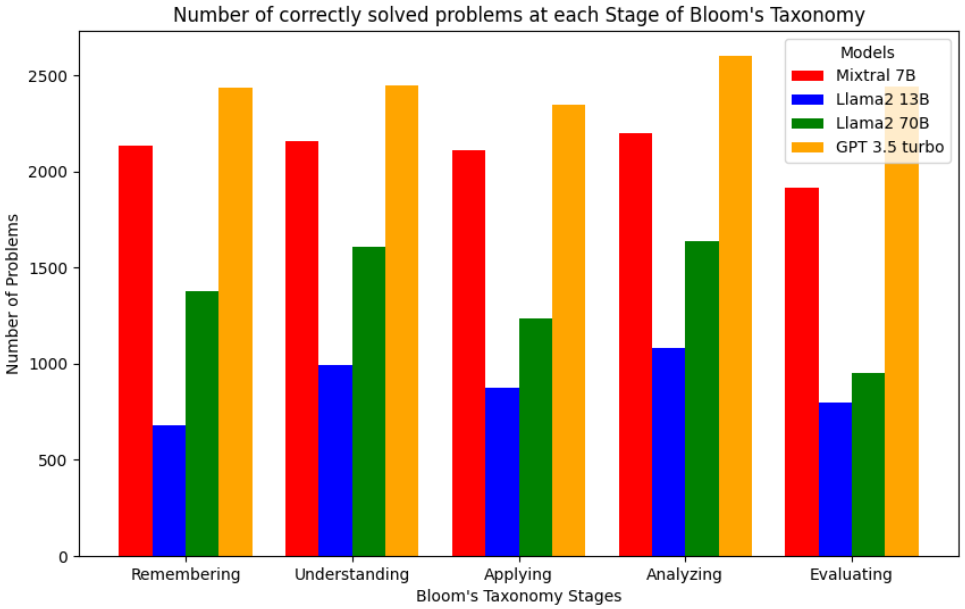
Μελέτες απαλοιφής και βελτιωμένη αυτοαξιολόγηση

Η ανάλυση των αποτελεσμάτων θα δομηθεί σε δύο άξονες. Ο πρώτος αφορά στη μελέτη και τον σχολιασμό των αποτελεσμάτων με βάση τα επίπεδα της ταξονομίας Bloom. Ο δεύτερος εστιάζει στην κατά το δυνατό γεφύρωση του χάσματος απόδοσης μεταξύ της αυτοαξιολόγησης (BLES) και του ιδανικού σεναρίου (BLO) του BloomWise μέσω της διερεύνησης καλύτερων προσεγγίσεων αυτοαξιολόγησης. Προς τον σκοπό αυτό, παρουσιάζουμε μια παραλλαγή της μεθόδου μας: το Program of Bloom.

Απόδοση σε κάθε γνωστικό επίπεδο: LLMs και γνωστικές ικανότητες Στο Σχήμα 1.3 αποτυπώνεται το πλήθος των προβλημάτων τα οποία επιλύθηκαν ορθά ανά γνωστικό επίπεδο για όλα τα προς εξέταση LLMs. Για αυτήν την ανάλυση, χρησιμοποιήσαμε τα prompts τα οποία αντιστοιχούν σε κάθε ένα εκ των πρώτων πέντε σταδίων της ταξονομίας (χωρίς πρόωρο τερματισμό). Ένα πρόβλημα είναι πιθανό να επιλύεται σωστά σε περισσότερα του ενός στάδια.

Dataset	Remembering	Understanding	Applying	Analyzing	Evaluating
GSM8K	72.6	73.2	70.2	78.0	73.7
SVAMP	81.6	83.3	76.5	81.8	81.2
Algebra	75.2	74.7	75.2	85.6	80.2
GSM-hard	37.8	36.5	37.0	42.8	36.6

Πίνακας 1.3: *Ακρίβεια ανά στάδιο της ταξινομίας Bloom για το GPT 3.5 turbo.*



Σχήμα 1.3: *Προβλήματα που λύθηκαν σωστά ανά επίπεδο της ταξονομίας Bloom*

Από το ιστόγραμμα, μπορούμε να εξαγάγουμε αρκετά συμπεράσματα για τη σχέση μεταξύ των γνωστικών ικανοτήτων ενός LLM, του μεγέθους του και της πολυπλοκότητάς του.

Πρώτον, όλα τα μοντέλα επιτυγχάνουν την καλύτερη απόδοσή τους στο στάδιο Ανάλυση', ακολουθούμενη από το Κατανόηση', παρόλο που η συνολική τους απόδοση διαφέρει. Παρόλο που διαισθητικά θα αναμέναμε ότι μικρότερα, λιγότερο εξελιγμένα μοντέλα θα υπερείχαν σε απλούστερες γνωστικές ικανότητες, αυτό δεν υποστηρίζεται από τα δεδομένα στο ιστόγραμμα. Αποτυπώνεται σαφώς η ανάγκη για εις βάθος κατανόηση και επεξεργασία του ίδιου του προβλήματος για όλα τα μοντέλα, καθώς τα στάδια Κατανόηση' και Ανάλυση' δίνουν έμφαση σε αυτά τα χαρακτηριστικά. Η υψηλή ακρίβεια σε αυτά τα στάδια μπορεί επίσης να οφείλεται στο γεγονός ότι οι προτροπές που συνδέονται με αυτά είναι παρόμοιες με τις προτροπές μεθόδων όπως το CoT. Συνεπώς, παρότι τα μικρότερα μοντέλα είναι λιγότερο εξελιγμένα, έχουν εκτεθεί σε τέτοια μοτίβα «σκέψης».

Δεύτερον, μεγαλύτερα LLMs, όπως το GPT-3.5 Turbo, ή πιο εξελιγμένα μοντέλα όπως το Mixtral, επιδεικνύουν πιο ομοιογενή απόδοση σε όλα τα γνωστικά επίπεδα, η οποία καταδεικνύει την ικανότητά τους να αντιμετωπίζουν προβλήματα σε οποιοδήποτε επίπεδο γνωστικής πολυπλοκότητας. Αντιθέτως, μικρότερα ή λιγότερο εξελιγμένα μοντέλα, όπως

το Llama2 που κυμαίνεται από 13b έως 70b, παρουσιάζουν μεγαλύτερη διακύμανση στην απόδοσή τους στα διαφορετικά επίπεδα. Αυτή η συμπεριφορά ενισχύει την πεποίθηση ότι η βαθιά κατανόηση του προβλήματος είναι εκ των ων ουκ άνευ, ειδικά για τα μικρότερα LLMs, πιθανότατα επειδή εφαρμόζουν πιστά αυτό που αναφέρεται ρητά στην προτροπή. Αντιθέτως, τα μεγαλύτερα μοντέλα είναι ικανά να εντοπίσουν και να καλύψουν τα κενά στην κατανόηση του προβλήματος πριν προχωρήσουν στη λύση, ακόμη και όταν δεν ζητείται ρητά. Αυτό υποδηλώνει ότι το μέγεθος και η πολυπλοκότητα πιθανόν συμβάλλουν στην ικανότητα ενός LLM να διατηρεί σχετικά σταθερή απόδοση ανεξάρτητα από τη γνωστική λειτουργία την οποία επιστρατεύει. Αυτή είναι μια ακόμη ένδειξη των βελτιωμένων ικανοτήτων γενίκευσης των μεγαλύτερων μοντέλων σε λιγότερο γνωστές μεθόδους προτροπής, όπως παραδείγματος χάριν, στο στάδιο 'Εφαρμογή'.

Απόδοση σε κάθε γνωστικό επίπεδο: Τύπος προβλήματος και γνωστικό βάθος Χάριν απλοποίησης αυτής της ανάλυσης, θα εστιάσουμε στα αποτελέσματα τα οποία αφορούν στο πιο εξελιγμένο μοντέλο, το GPT-3.5 Turbo. Τα αποτελέσματα ανά επίπεδο της ταξονομίας για κάθε σύνολο δεδομένων παρατίθενται στον Πίνακα 1.3 (για τον ενδιαφερόμενο αναγνώστη, τα αποτελέσματα για τα υπόλοιπα μοντέλα εμφανίζονται στο Παράρτημα Α). Το GSM8K, το πιο απαιτητικό σύνολο δεδομένων από πλευράς μεθοδολογίας επίλυσης, επιτυγχάνει μεγαλύτερη ακρίβεια στα στάδια ¨Ανάλυση¨ και ¨Αξιολόγηση¨, γεγονός το οποίο καταδεικνύει πως τα δυσκολότερα προβλήματα απαιτούν επιστράτευση υψηλότερων σταδίων της ταξονομίας για να επιλυθούν με ακρίβεια. Το αντίστροφο φαίνεται πως δεν ισχύει. Για το σύνολο δεδομένων Algebra, η καλύτερη απόδοση επιτυγχάνευαι στα δύο τελευταία στάδια. Το σύνολο δεδομένων SVAMP (το καλύτερο και το δεύτερο καλύτερο σε ακρίβεια για τα BLM και BLES, αντίστοιχα) αποδίδει καλύτερα στα στάδια ¨Κατανόηση¨ και ¨Ανάλυση¨, αλλά καταγράφει τη χαμηλότερη απόδοσή του στην εφαρμογή. Το πιο απαιτητικό σύνολο δεδομένων από πλευράς υπολογισμών, -το GSM-hard-, επιτυγχάνει την υψηλότερη ακρίβειά του στο ¨Ανάλυση¨, ακολουθούμενη από το ¨Ανάκληση¨, κάτι που θα μπορούσε να υποδηλώνει υψηλό επίπεδο εξοικείωσης αυτού του μοντέλου με τα προβλήματα.

Αυτοαξιολόγηση Τα σχετικά με την ακρίβεια της αυτοαξιολόγησης αποτελέσματα παρατίθενται στον Πίνακα 1.4. Έχει διαπιστωθεί ότι η αυτοαξιολόγηση λειτουργεί καλύτερα για μεγαλύτερα ή πιο εξελιγμένα μοντέλα. Αυτά τα μοντέλα, με τις εκτεταμένες βάσεις γνώσεων και τις προηγμένες ικανότητες συλλογισμού, είναι καλύτερα εξοπλισμένα προκειμένου να αξιολογούν την ποιότητα των απαντήσεών τους και να πραγματοποιούν τις απαραίτητες προσαρμογές. Μπορούν να εντοπίζουν κενά στην κατανόησή τους, να αναγνωρίζουν πότε μια λύση δεν είναι βιώσιμη και να προσπαθούν έως ότου να καταλήξουν σε μια πιο ακριβή και ολοκληρωμένη απάντηση.

Program of Bloom Προκειμένου να μειώσουμε τα σφάλματα στους υπολογισμούς, χρησιμοποιήσαμε Python κώδικα. Πιο συγκεκριμένα, χρησιμοποιήσαμε τις ίδιες προτροπές αλλά ζητήσαμε την απάντηση σε μορφή Python κώδικα, ο οποίος εκτελέστηκε με ασφάλεια. Η διαχείριση του Python κώδικα, ειδικά για την αυτοαξιολόγηση, είναι δύσκολη για τα μικρότερα LLMs, επομένως περιορίσαμε τα πειράματά μας στο GPT-3.5 Turbo. Για το

Model	Ακρίβεια αυτοαξιολόγησης
GPT 3.5 turbo	68.1
Llama2 70B	56.5
Llama2 13B	45.8
Mixtral 8x7B	62.8

Πίνακας 1.4: *Ακρίβεια αυτοαξιολόγησης ανά μουτέλο. Τα μεγαλύτερα/πιο εξελιγμένα LLMs επιτυγχάνουν πιο ακριβή αυτοαξιολόγηση.*

σύνολο δεδομένων GSM-hard—το πιο απαιτητικό από άποψη υπολογισμών—η ακρίβεια βελτιώθηκε κατά 13%. Για τα υπόλοιπα σύνολα δεδομένων, η ακρίβεια ήταν χαμηλότερη σε σύγκριση με αυτή που επιτεύχθηκε με το BloomWise. Τα αποτελέσματα παρατίθενται στον Πίνακα 1.5. Μια πιθανή εξήγηση για τη μειωμένη απόδοση αυτής της παραλλαγής είναι το γεγονός ότι οι προτροπές δεν είναι κατάλληλες για την έκφραση της απάντησης σε μορφή κώδικα. Η κειμενική αιτιολόγηση αποτελεί πιο κατάλληλο τρόπο για τη διαμόρφωση της απάντησης, καθώς η αυστηρή δομή που απαιτείται για τη χρήση κώδικα δεν επιτρέπει πάντα την πιστή εφαρμογή των προτροπών.

Dataset	BloomWise	Program of Bloom
GSM8K	78.2	73.2
SVAMP	84.7	81.0
Algebra	85.1	67.1
GSM-hard	40.6	53.6

Πίνακας 1.5: *Σύγκριση μεταξύ Program of Bloom και BloomWise - Αποτελέσματα για GPT3.5 turbo*

Κεφάλαιο 2

introduction

2.1 Motivation

Large Language Models (LLMs) have demonstrated impressive performance across a wide range of tasks, such as text generation, sentiment analysis, information retrieval, and code generation. These successes have made them highly valuable tools in fields like customer support automation, creative writing, and data analysis. However, one key area where LLMs continue to exhibit weaknesses is in solving mathematical problems. Mathematical reasoning requires precision, structured thinking, and often multi-layered logic, which LLMs frequently struggle to achieve.

Stefan Banach once remarked, “Mathematics is the most beautiful and most powerful creation of the human spirit". This statement highlights the universal significance of mathematics as both a foundational tool and an art form that embodies human ingenuity. Within this context, equipping large language models (LLMs) with enhanced mathematical reasoning abilities becomes not only a technical goal but a vital step in advancing artificial intelligence. Mathematics is the language through which we describe and understand the complexities of nature, engineer groundbreaking technologies, and explore theoretical landscapes that push the limits of human knowledge. As we continue to push the boundaries of what LLMs can achieve, their ability to comprehend, reason through, and manipulate mathematical concepts emerges as a cornerstone for unlocking their full potential. From solving real-world problems in physics, engineering, and computer science to enabling breakthroughs in machine learning itself, robust mathematical reasoning in LLMs has far-reaching implications. This capability enables the development of more accurate predictive models, enhances decision-making in complex systems, and facilitates collaboration between humans and machines in solving intricate scientific challenges.

Many recent approaches have leveraged the capabilities of in-context learning (ICL) [4] to enhance LLMs’ problem-solving skills. Some of the most widely used techniques involve encouraging the LLM through prompts to develop a textual rationale with Chain-of-Thought prompting [5] (CoT) or Python functions with Program-Aided Language Model [6] and Program-of-Thought prompting [7] (PAL or PoT). Each approach has both advantages and drawbacks: Chain of Thought (CoT) breaks down reasoning into a sequential narrative, allowing for dynamic problem-solving approaches. However, it can sometimes compromise numerical accuracy [5],[8], as language models frequently encounter challenges with mathematical operations. On the other hand, techniques like PoT and PAL utilize Python code to tackle questions, leveraging the precision of Python interpreters for reliable computations but cannot deal with unknown variables.

Recently, research efforts have pivoted towards integrating multiple methodologies,

aiming to identify the most suitable approach for each specific problem, while harnessing the collective strengths of various techniques. The X of Thoughts [3] is a prompting method according to which the most suitable among the techniques of CoT (Chain of Thought), PoT (Program of Thought), and EoT (Equations of Thought) is selected, applied, and verified (iteratively, until a correct solution is found) with the goal of improving LLMs' performance in solving mathematical problems.

Building on the foundation of methodological integration and striving for enhanced self-awareness in LLMs, we propose BloomWise, a novel cognitively-motivated prompting method. BloomWise encourages LLMs to methodically engage higher-order cognitive functions in a hierarchical fashion to reach a solution, persisting through a process until the correct resolution is found. By fostering a deliberate activation of cognitive processes, BloomWise delves deeper into the realm of cognitive capabilities, aligning problem-solving strategies with Bloom's taxonomy of educational objectives (see Figure 5.5) to improve understanding and precision in tackling complex mathematical challenges. By utilizing multi-level cognitive prompting we aim to enhance both the interpretability and accuracy of outcomes.

The main motivation behind BloomWise is to encourage the LLM to reach a solution by utilizing cognitive skills of gradually increasing complexity via multi-level prompting. The answer at each of the first five levels of Bloom's taxonomy is self-evaluated by the LLM, which announces success if the solution is deemed correct, preventing progression to the next level. To demonstrate the effectiveness of our approach, we conduct extensive experiments on four popular mathematical reasoning datasets and four LLMs and achieve consistent improvements. Additionally, we explore several variants of our method, including majority voting among levels and Program of Bloom, which combines Bloom prompting with PoT. Moreover, we assess the full potential of the proposed method using an Oracle to decide if we should move to the next level of the taxonomy.

2.2 Contributions

The main contributions of this thesis are:

1. We introduce a novel cognitively-motivated multi-level prompting method for solving mathematical (and potentially other types of) problems based on Bloom's taxonomy.

2. We incorporate the idea of early stopping in prompting, as the execution of the method terminates before iterating to all levels of the taxonomy when a solution that is self-evaluated as correct is reached.

3. We investigate the performance of various LLMs at each cognitive level of the Bloom taxonomy for four popular math datasets.

Our results offer valuable insights into the cognitive skills exhibited by each LLM, as well as the skills required to solve different types of mathematical problems. Furthermore, we demonstrate the potential of muti-level cognitively-motivated prompting for improving accuracy and enhancing interpretability.

2.3 Thesis outline

In Chapter 3, Background: Artificial Intelligence, Machine Learning, and Deep Learning, we establish the theoretical foundation necessary to equip the reader for the subsequent chapters. This section aims to familiarize the reader with key concepts in Artificial Intelligence, Machine Learning, and Deep Learning, while introducing the fundamental categories of machine learning. Additionally, it outlines core principles related to the training of machine learning models and presents a range of commonly employed methods and models in both traditional machine learning and deep learning domains.

In Chapter 4, Background: Word Representation Methods and Large Language Models, we present an overview of the field of Natural Language Processing, discussing various word representation techniques and language models that have been developed over time to model and interpret human language.

In chapter 5, Mathematical Reasoning using in-context learning, we define the problem and present the related work.

In Chapter 6, Proposed Approach, we analyze the approach our work follows, explaining its purpose, as well as our method and presenting the results obtained from our experiments. We also provide a discussion over our results, explaining our conclusions.

In Chapter 7, Conclusions, limitations and future work, we provide a summary of our methods and findings, discuss the limitations of this work and suggest some potential approaches for future work.

Κεφάλαιο 3

Background: Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence (AI), often viewed as the cornerstone of modern technological progress, has roots that stretch deep into human history, shaped by an enduring fascination with creating autonomous beings. The idea of intelligent machines or entities functioning independently of human control has been explored for millennia through myth, literature, and philosophy. In Homer's The Odyssey, for example, the mythical Feakes' boats are portrayed as having minds of their own. These self-guided vessels, needing no captains or steering wheels, are capable of understanding the thoughts and intentions of their passengers, navigating effortlessly to any destination. This ancient narrative reveals an early human vision of systems that can perceive and respond to human desires, prefiguring the goals of modern AI. Similarly, in Greek mythology, Talos—a bronze automaton forged by the god Hephaestus—was created to protect the island of Crete, patrolling its shores and defending it autonomously from threats. These early imaginings of sentient boats and mechanical guardians exemplify humanity's timeless dream of building intelligent systems that can act independently, laying the conceptual groundwork for the development of AI long before the digital age. Despite these ancient imaginings, the development of AI as we know it today depended on advancements in computer technology. Early AI systems focused on formal tasks, where formal languages were used to represent knowledge, allowing computers to reason through logical inference. However, these systems faced significant limitations, particularly when dealing with tasks that are intuitive for humans, such as voice recognition. The challenge lay in describing all relevant knowledge explicitly, which proved to be impractical for many real-world applications.

This challenge led to the evolution of a new paradigm in AI—machine learning (ML). Rather than relying solely on pre-defined rules, ML systems are designed to recognize patterns within data, allowing them to learn from experience and extract useful knowledge for specific tasks. A key feature of machine learning is the algorithm's ability to map data representations to outputs, identifying which features correlate with particular outcomes [20]. While this approach improved AI performance, it introduced new challenges: determining the most relevant features for each task proved difficult.

To address this, representation learning emerged, where AI systems are capable of learning not only the mapping from data to output but also the features themselves. This allows systems to extract the most meaningful representations directly from raw data, leading to significant advances in performance across various applications. However, extracting abstract, high-level features from complex data can be difficult due to the many factors influencing data variation [20].

This complexity gave rise to deep learning (DL), a subset of machine learning that uses layered architectures to build increasingly abstract representations. Each higher layer of the model derives more complex features from simpler ones below, enabling AI systems to solve tasks ranging from image segmentation and machine translation to emotion recognition and conversational agents. Deep learning, now extensively researched and applied across both commercial and academic domains, has become instrumental in pushing the boundaries of what AI can achieve.

In the following sections, we delve deeper into the key concepts underlying artificial intelligence, machine learning and deep learning.

3.1 Artificial intelligence

Artificial Intelligence is the branch of Computer Science that deals with the automation of intelligent behavior.

Early AI efforts concentrated on rule-based systems, where human knowledge was encoded into machines through formal languages and logic-based frameworks. While these systems showed promise for structured tasks, they struggled with more complex, intuitive tasks, such as recognizing patterns in data or adapting to new information. These limitations spurred the development of more advanced approaches, such as Machine Learning (ML), where systems learn directly from data, and Deep Learning (DL), which uses layered neural networks to model complex representations.

As ML and DL have rapidly evolved, they have become the driving forces behind many of the most impressive AI applications today, from natural language processing and computer vision to autonomous vehicles and personalized recommendations.

3.2 Machine Learning

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that focuses on principles, methods, and algorithms for learning and making predictions based on past evidence. Unlike traditional AI systems that rely on explicitly coded instructions, ML enables machines to improve their performance on tasks by learning from data and experience. The primary goal of ML is to create systems that can analyze data, recognize patterns, and make decisions or predictions without being explicitly programmed for every scenario, allowing them to adapt and improve over time.

Machine learning can be categorized into four primary types based on the approach used for training the algorithm: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. In the following sections, we will delve into each of these categories, examining their characteristics and the methods commonly employed within them.

3.2.1 Supervised learning

In supervised learning, the machine learning algorithm is trained using data with predetermined correct outputs, often referred to as labeled data. The algorithm is provided with a dataset consisting of input-output pairs, which it uses to learn how to predict correct outputs based on given inputs.

Mathematically, given a collection of input variables X and corresponding output variables Y, an algorithm is employed to learn a mapping function f that links the input X to the output Y such that:

$$Y = f(X)$$

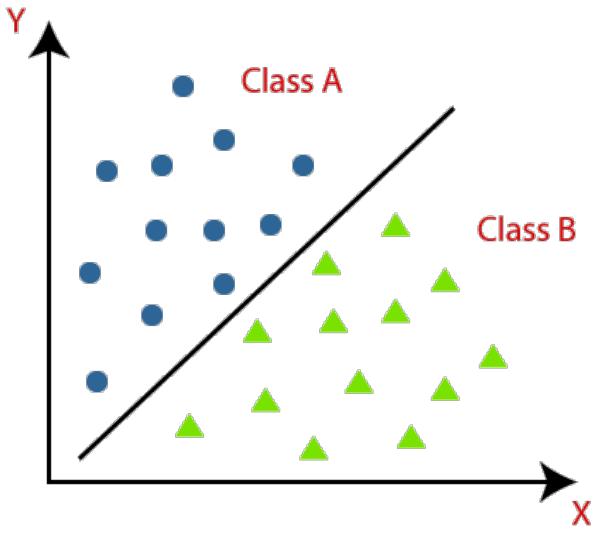
In this setup, both the labeled inputs X and their associated outputs Y are available and utilized in the training process.

Since the data must be labeled, typically by a human, supervised learning datasets are often limited to small or medium sizes, which can restrict the performance of supervised learning models.

Supervised learning algorithms can be divided into two categories, depending on the type of output they are designed to predict: Classification and Regression.

Classification

Classification refers to the task of determining a mapping function from independent (input) variables to discrete output variables, commonly referred to as labels. In other words, classification involves assigning each input to a specific class from a predefined set of classes. An example of a typical classification problem is spam email detection.



Σχήμα 3.1: *Classification example*

In the following paragraphs, we introduce some of the most widely used traditional machine learning models for classification:

Logistic Regression In binary classification, where the task is to differentiate between two classes, logistic regression models the probability of a data point, represented by a feature vector ϕ, belonging to class C_1 as:

$$p(C_1|\phi) = y(\phi) = \sigma(w^\top \phi)$$

where w is a parameter vector, and the logistic sigmoid function $\sigma(a)$ is defined as:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

The probability of the data point belonging to the second class C_2 can then be computed as:

$$p(C_2|\phi) = 1 - p(C_1|\phi)$$

To estimate the parameters of the logistic regression model, maximum likelihood estimation is used. The likelihood function can be expressed as:

$$p(t|w) = \prod_{n=1}^{N} p(C_1|\phi_n)^{t_n}(1 - p(C_1|\phi_n))^{1-t_n}$$

where N is the number of training samples, $t_n \in \{0, 1\}$ represents the class label for sample n, and ϕ_n is the corresponding feature vector. By taking the negative logarithm of the likelihood, we define the error function to be minimized as:

$$E(w) = -\ln p(t|w) = -\sum_{n=1}^{N} [t_n \ln p(C_1|\phi_n) + (1 - t_n) \ln(1 - p(C_1|\phi_n))]$$

This error function is known as the cross-entropy loss function. Although logistic regression is typically used for binary classification, it can be extended to handle multiple classes. One approach, known as the "one-vs-rest" strategy, trains a logistic regression model for each class, comparing the probability of belonging to that class against all others. Another approach is multinomial logistic regression, which extends binary logistic regression by using the softmax function:

$$p(C_k|\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = w_k^\top \phi$, w_k is the parameter vector for class k, and $p(C_k|\phi)$ represents the probability of the sample belonging to class k [21].

Naive Bayes Classifier The Naive Bayes classifier is a probabilistic model used for classification based on Bayes' Theorem. For a class y and a feature vector $x = (x_1, x_2, ..., x_n)$, Bayes' Theorem is expressed as:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

By assuming the conditional independence of features, this can be simplified to:

$$P(y|x) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)}$$

Since the denominator remains unchanged across all classes, classification is achieved by selecting the class $\hat{y}$ that maximizes the posterior probability:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

When dealing with continuous features, we can assume they are sampled from a Gaussian distribution and model the likelihood as:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

This forms the basis of the Gaussian Naive Bayes classifier. While the Naive Bayes classifier is known for its speed and simplicity, its assumption of feature independence often limits its effectiveness in complex real-world applications.

K-Nearest Neighbors (KNN) The K-Nearest Neighbors (KNN) algorithm is a non-parametric classification method that predicts the class of a data point based on its proximity to other data points in the feature space. For a given test sample, the algorithm computes its distance from all training samples and selects the K nearest neighbors. The test sample is then classified based on the majority class among these K neighbors.

The most common distance measure used is the Euclidean distance, which for two points $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ is defined as:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}$$

Although KNN is typically used for classification, it can also be applied to regression tasks, where the predicted value is the average of the K nearest neighbors' values.

Support Vector Machines (SVM) For a binary classification problem with two linearly separable classes ω_1 and ω_2, the SVM algorithm seeks to find the optimal hyperplane:

$$g(x) = w^\top \phi(x) + w_0 = 0$$

where $\phi(x)$ is a transformation into a higher-dimensional feature space [21], [22]. The optimal hyperplane is defined as the one that maximizes the margin—the minimum distance between the hyperplane and any training sample. The training samples that lie closest to the hyperplane are called support vectors.

For cases where the classes are not linearly separable in the input space, an appropriate transformation $\phi()$ can map the data into a higher-dimensional space where linear separation becomes possible. This is known as the "kernel trick."

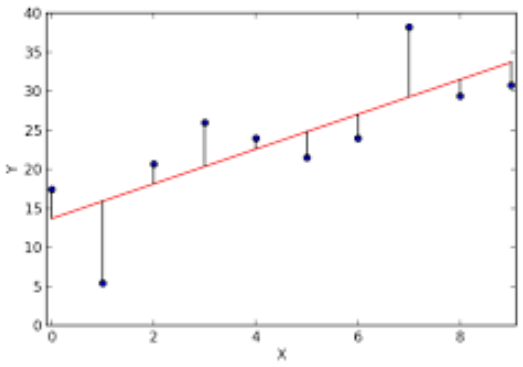
In multi-class classification problems, SVM can be extended using either the "one-vs-one" approach, which trains an SVM for every pair of classes, or the "one-vs-rest"

approach, which trains an SVM for each class, distinguishing it from all other classes.

SVMs can be used for both classification and regression tasks.

Regression

Regression is the task of determining a mapping function from independent (input) variables to dependent (output) variables. The output values that the regression algorithm attempts to predict are continuous. Typical regression tasks involve predicting variables such as stock market or real estate prices.



Σχήμα 3.2: *Regression example*

Linear Regression Let $x \in \mathbb{R}^n$ be an input vector and $y \in \mathbb{R}$ be a scalar value to be predicted as its output. In linear regression, y is considered to be a linear function of the input x. We define $\hat{y}$ as the value approximating y that the linear regression model predicts. We can then express $\hat{y}$ as:

$$\hat{y} = w^\top x$$

where $w \in \mathbb{R}^n$ is a vector of parameters [20].

The vector w consists of a series of weights, each of which determines the degree to which its associated feature x_i contributes to the output. A larger absolute value of the i-th weight w_i signifies a stronger influence of the feature x_i on the predicted value $\hat{y}$, while a weight of zero indicates that the feature does not contribute to the model's prediction.

To determine the optimal parameters w for the linear regression model, we first need to define a performance measure. The mean squared error (MSE) can be used as this metric and is defined as:

$$\text{MSE} = \frac{1}{m} \sum_i (\hat{y}_i - y_i)^2$$

where $\hat{y}$ represents the predictions of the model, y are the correct output values, and m is the number of samples. The goal is to minimize the MSE, with an MSE of zero indicating perfect accuracy ($\hat{y} = y$).

Minimizing the MSE can be achieved by finding the point at which its gradient becomes zero:

$$\nabla_w \text{MSE} = 0$$

3.2.2 Unsupervised learning

In unsupervised learning, all of the data is unlabeled, meaning that only the input X is available and not the output Y. The goal of unsupervised learning algorithms is to discover the underlying structure or distribution within the data.

Since the data used in unsupervised learning is unlabeled, no human labor is required to create the datasets, making them typically much larger than those used in supervised learning. Moreover, unsupervised learning algorithms are more capable of adapting to new data, as they can dynamically adjust the underlying structures they have learned.

Clustering

Clustering refers to the process of organizing unlabeled data into groups based on their similarities. Clustering algorithms can be categorized into various types based on how they group data. Some of the most notable types are:

- **Hierarchical Clustering:** Hierarchical clustering algorithms are divided into two types: Agglomerative and Divisive. Agglomerative clustering takes a bottom-up approach, starting by treating each data point as its own cluster and successively merging similar clusters until the desired number of clusters is achieved. Divisive clustering, on the other hand, follows a top-down approach, beginning with all data points in a single cluster and iteratively splitting clusters based on dissimilarity.
- **Centroid-based or Partition Clustering:** In these algorithms, data points are divided into a predefined number of clusters, with each cluster represented by a vector. The distance between each data point and the cluster vectors is calculated (using one of the widely used metrics, eg Euclidean, Manhattan), and the data point is assigned to the nearest cluster.
- **Density-based Clustering:** Density-based algorithms group data based on regions of high density, with clusters forming in areas where data points are concentrated, surrounded by regions of lower density.
- **Probabilistic Clustering:** Probabilistic clustering methods group data based on the likelihood that they belong to the same distribution. Each data point is assigned to the cluster to which it most likely belongs. The probabilities are determined through an optimization process, with common distributions used including the Normal and Gaussian distributions.

Clustering algorithms can also be divided based on how they assign data points to clusters:

- **Hard Clustering:** Hard clustering algorithms assign each data point to a single cluster and a data point either belongs to a cluster or it does not.

- **Soft Clustering:** In soft clustering, data points are assigned to clusters based on probabilities, calculating the likelihood of each data point belonging to each cluster.

K-Means

K-Means is a widely-used clustering algorithm that divides the data into K pre-defined, non-overlapping clusters, where each data point belongs to exactly one cluster (hard clustering). Given N data points $x_1, x_2, \ldots, x_N$, and K clusters, a vector μ_k is assigned to each cluster. The algorithm aims to minimize the following objective function:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

where $r_{nk} = 1$ if data point x_n is assigned to cluster k, and $r_{nk} = 0$ otherwise. The objective function J represents the sum of squared distances from each data point to the vector μ_k of its corresponding cluster.

The algorithm iteratively optimizes r_{nk} and μ_k to minimize J. It begins with initial values for μ_k and alternates between two steps: first, it minimizes J with respect to r_{nk} while keeping μ_k fixed, and second, it minimizes J with respect to μ_k, keeping r_{nk} fixed. In the first step, each data point x_n is assigned to the cluster with the smallest distance $\|x_n - \mu_k\|^2$. In the second step, the new value of μ_k is calculated as:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

The process repeats until the assignments of data points to clusters remain unchanged.

Dimensionality Reduction

Dimensionality reduction involves reducing the number of features in a dataset. In machine learning, datasets often have a lot of features, making it difficult to visualize or train a model effectively. This leads to issues associated with high-dimensional data.

One such issue is data sparsity: when the dimensionality is high, the combinations of attribute values in the training data become a small subset of the possible attribute combinations. This can result in overfitting, where the model performs well on the training data but struggles to generalize to unseen data.

Another challenge is that, as dimensionality increases, the distances between data points in the feature space may converge, rendering distance-based measures (used by algorithms such as KNN or K-Means) less meaningful for encoding similarity.

The goal of dimensionality reduction is to reduce the number of features while retaining as much of the data's integrity as possible. One of the most commonly used techniques for this purpose is Principal Component Analysis (PCA).

PCA

Principal Component Analysis (PCA) projects data onto a lower-dimensional linear space in a way that maximizes the variance of the data in this reduced space. PCA identifies the first principal component as the direction in which the variance of the data

is greatest. Subsequent principal components are defined incrementally, each orthogonal to the previous components and chosen to maximize variance. To reduce dimensionality, we retain only the projections of the data onto the first M principal components, where M is the desired number of dimensions.

3.2.3 Semi- supervised learning

In semi-supervised learning, only a subset of the dataset is labeled. This technique merges aspects of both supervised and unsupervised learning, making it suitable when there are only a limited number of labeled examples but a vast amount of unlabeled data [23] . The approach begins by training a model on the labeled subset, then incorporates the unlabeled data to refine its understanding of the data's structure, leading to improved generalization to new, unseen data.

3.2.4 Reinforcement learning

In reinforcement learning, an agent interacts with an environment to achieve a goal by learning from the consequences of its actions. Unlike supervised learning, where the model learns from labeled examples, reinforcement learning relies on trial and error, receiving feedback in the form of rewards or penalties. The agent aims to maximize cumulative rewards by learning an optimal policy, which dictates the best action to take in any given state. Through exploration of the environment and exploitation of the learned policy, the agent progressively improves its decision-making, enabling it to generalize and perform well in new, unseen situations.

3.3 Deep Learning

Deep learning is a subset of machine learning that uses multi-layered neural networks to automatically learn and extract complex features from large datasets, enabling tasks such as image recognition, speech processing, and natural language understanding.

Artificial Neural Networks (ANNs) form the foundation of deep learning and are named due to their structural resemblance to biological neurons in the human brain. ANNs are composed of interconnected nodes, or neurons, that are arranged in layers. Typically, ANNs have an input layer, an output layer, and one or more hidden layers in between. Each node can connect to one or more other nodes, with each connection assigned a weight. The output of each node is computed by summing the outputs from the nodes in the previous layer it is connected to. However, this calculation is initially a linear combination of the previous layer's outputs. If this linear function alone is used, the entire network would be reduced to a simple linear model, even with multiple layers. To enable the network to learn more complex relationships and use multiple layers effectively, a non-linear function, known as the activation function, is applied to the node's output. Furthermore, during the training process, a metric called the loss function is defined to measure the error between the model's predictions and the actual labels in the dataset. The goal of the neural network is to minimize this loss function during training. To

achieve this, an algorithm called back-propagation is employed, which calculates the required gradients for optimization. The following sections will delve into these concepts, including the activation function, loss function, and back-propagation.

Activation function The activation function is a non-linear function that takes the weighted sum of the outputs from previous nodes (or input data in the case of the first layer) and computes the output of a node in the neural network. Activation functions are typically differentiable, which allows them to be used with the back-propagation algorithm (discussed in subsequent sections). They generally vary between the input and hidden layers and the output layer of the neural network.

Some of the most commonly used activation functions include:

Activation Function	**Definition**	**Description**
Linear/Identity	$f(x) = x$	A linear activation function, rarely used because it lacks non-linearity, reducing the network to a single layer. Not suitable for back-propagation due to its constant derivative.
Sigmoid/Logistic	$f(x) = \frac{1}{1+e^{-x}}$	Used for probability predictions with output between (0, 1). Suffers from vanishing gradient and is not symmetric around zero, making training less stable.
Tanh (Hyperbolic Tangent)	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	Similar to sigmoid but zero-centered with an output range of (-1, 1). Often used in hidden layers but still suffers from vanishing gradient.
ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$	Efficient and prevents the vanishing gradient problem. However, it has a zero gradient for negative inputs, potentially leading to dead neurons (dying ReLU problem).
Leaky ReLU	$f(x) = \begin{cases} x & \text{for } x > 0 \\ ax & \text{for } x \leq 0 \end{cases}$	A variant of ReLU that allows small negative values by introducing a slope for negative inputs, helping avoid dead neurons during back-propagation.

Πίνακας 3.1: *Summary of Common Activation Functions in Neural Networks*

Loss function The loss function is a metric that measures the error associated with particular decisions or actions. In supervised learning, we need to quantify the difference between our model's predictions and the actual labels, which is done using a loss function. The function maps the predicted labels $\hat{y}$ and the true labels y to a real number, indicating the total error. The goal of training is to minimize this loss. Below are some common loss functions used in machine learning, categorized by task:

Loss Functions for Regression Problems

Loss Function	Definition	Description
Mean Squared Error (MSE)	$L(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$	Measures the squared difference between predicted and true labels, emphasizing larger errors and ignoring the sign of the error.
Mean Absolute Error (MAE)	$L(w) = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$	Computes the absolute difference between predicted and true labels. Less sensitive to large errors and more robust to outliers compared to MSE.

Πίνακας 3.2: *Loss Functions for Regression Problems*

Loss Functions for Classification Problems

Loss Function	Definition	Description
Kullback-Leibler Divergence (KL Divergence)	$\mathrm{KL}(P, Q) = -\sum_x P(x) \log \frac{Q(x)}{P(x)}$	Measures the divergence between true and predicted distributions, indicating how much the predicted distribution deviates from the true distribution.
Cross-Entropy Loss	$H(P, Q) = -\sum_x P(x) \log Q(x)$	Commonly used in classification, it quantifies the difference between the true and predicted probability distributions.

Πίνακας 3.3: *Loss Functions for Classification Problems*

Backpropagation In neural networks, a loss function is used to evaluate the network's error, which we aim to minimize during training. To achieve this, we need to compute the gradient of the loss function with respect to the network's weights. In the hidden layers, the input to each node is influenced by the outputs of the previous layers, and the output of each node, in turn, affects the final predictions and the overall loss. This interdependence between layers complicates the calculation of gradients for each parameter. To efficiently compute these gradients, we use the back-propagation algorithm [24]. After each forward pass through the network, back-propagation performs a backward pass, starting from the last layer and moving toward the input layer. It uses the chain rule to compute the gradient of the loss function for each weight by leveraging the gradients computed in the previous layers. One key advantage of back-propagation is that it provides insight into how changes in the network's weights impact the loss function and the model's overall behavior.

3.4 DL models

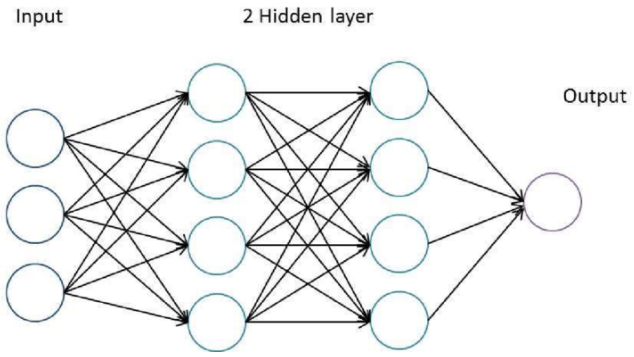
In this section, we introduce several of the most widely used deep learning model types.

3.4.1 Feed-forward neural networks

Feed-forward neural networks (FFNN) are a type of artificial neural network (ANN) where the connections between nodes move in one direction, from the input layer to the output layer, without forming cycles. FFNNs are primarily used for supervised learning tasks with independent, non-sequential data. The simplest form of FFNN is the perceptron, a linear model with no hidden layers. It calculates the weighted sum of input features and passes it through an activation function:

$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right)$$

where w_i are the weights, x_i are the input features, and b is a bias term. For more complex tasks involving non-linearly separable data, the multi-layer perceptron (MLP) is used. An MLP consists of multiple layers, including at least one hidden layer, and can learn non-linear functions for both regression and classification tasks.

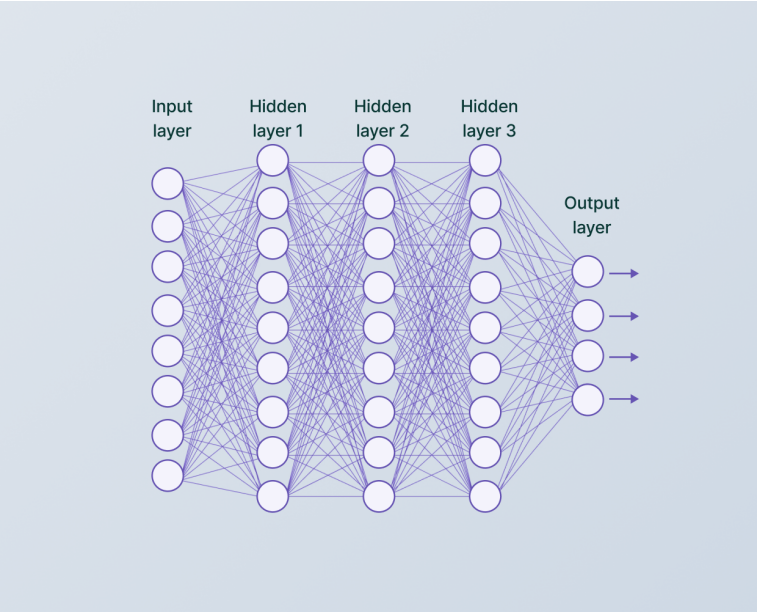


Σχήμα 3.3: *FFNN example*

3.4.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [25] are designed to process image data by using convolution, inspired by the human visual cortex. CNNs consist of three main types of layers:

- **Convolutional layer:** Applies filters to the input image, generating activation maps. Typically followed by a non-linear activation function like ReLU.
- **Pooling layer:** Reduces dimensionality by aggregating pixel values, improving efficiency and reducing overfitting. Max and average pooling are commonly used.
- **Fully-connected layer:** The final layer that performs classification using features extracted by previous layers.



Σχήμα 3.4: *CNN example*

3.4.3 Recurrent Neural Networks

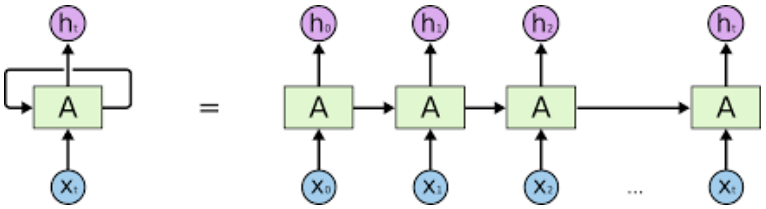
Humans do not reset their thoughts every moment. Each word's meaning builds upon the understanding of prior words—thinking is continuous. Recurrent neural networks solve this problem by incorporating loops, allowing the network to retain and use information over time [26].

First introduced as early as the 1980s [27, 28], a recurrent neural network (RNN) is a type of artificial neural network characterized by a feedback loop, enabling data from previous inputs to be fed back into the network's nodes. This structure allows RNNs to maintain an "internal memory," referred to as a hidden state, which is continually updated with new input and retains past information to influence future outputs. When the feedback loop is conceptually "unrolled" over time, an RNN can be viewed as multiple iterations of the same network, each corresponding to a different point in the input sequence and passing information forward in a sequential manner. The vanilla RNN architecture can be mathematically expressed by the following equations:

$$h_t = f_h(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = f_y(W_{yh}h_t + b_y)$$

where x_t represents the input vector at time step t, h_t is the hidden state, and y_t is the output. The terms b_h and b_y are bias terms for the hidden state and output, respectively, and f_h and f_y denote the activation functions. The weight matrices W_{hx}, W_{hh}, and W_{yh} correspond to the input-to-hidden, hidden-to-hidden, and hidden-to-output connections, respectively. Time is considered in terms of input points, with each new input corresponding to a new time step.



Σχήμα 3.5: *RNN example*

3.4.4 Long Short-Term Memory networks

One of the key strengths of RNNs is their potential to relate past information to the current task.

In certain situations, only recent information is necessary to complete a task. For example, in a language model predicting the next word in a sentence, the immediate prior words often provide sufficient context. When the gap between the relevant information and its use is small, RNNs can effectively learn from this past data.

Nevertheless, there are instances where a more extended context is required, and the gap between relevant information and the point where it is needed can become significant. As this gap increases, RNNs struggle to maintain the connection between distant past inputs and the current task. While, in theory, RNNs are capable of managing such long-term dependencies, practical challenges arise due to inherent limitations [29].

One of the main reasons for this limitation is the vanishing gradient problem. During back-propagation through time (BPTT), RNNs rely on repeated multiplication of gradients across many layers in time. This repeated multiplication can cause gradients to shrink exponentially, approaching zero. As a result, the network struggles to retain information from earlier layers, limiting its ability to learn long-term dependencies. Conversely, RNNs can also encounter the exploding gradient problem, where gradients grow exponentially and become excessively large. This instability can lead to an unstable network, making it difficult to converge to an optimal solution.

These issues make it challenging for RNNs to capture and retain information from distant past events, affecting their performance on tasks requiring long-term memory and stable learning over extended sequences.

Fortunately, Long Short Term Memory networks (LSTM) were developed to address these issues [30]. LSTMs add a cell state, in addition to the hidden state, which runs through all layers of the network and is modified only by minor linear interactions, making it more capable of retaining past information. To control what information is added or removed from the cell state, LSTMs employ structures called gates.

The LSTM architecture can be mathematically described by the following equations, where x_t is the input vector, c_t is the cell state, and h_t is the hidden state at time step t. W_f, U_f, W_i, U_i, W_o, U_o, W_c, and U_c are weight matrices, and b_f, b_i, b_o, and b_c are bias terms:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.4.4.1}$$

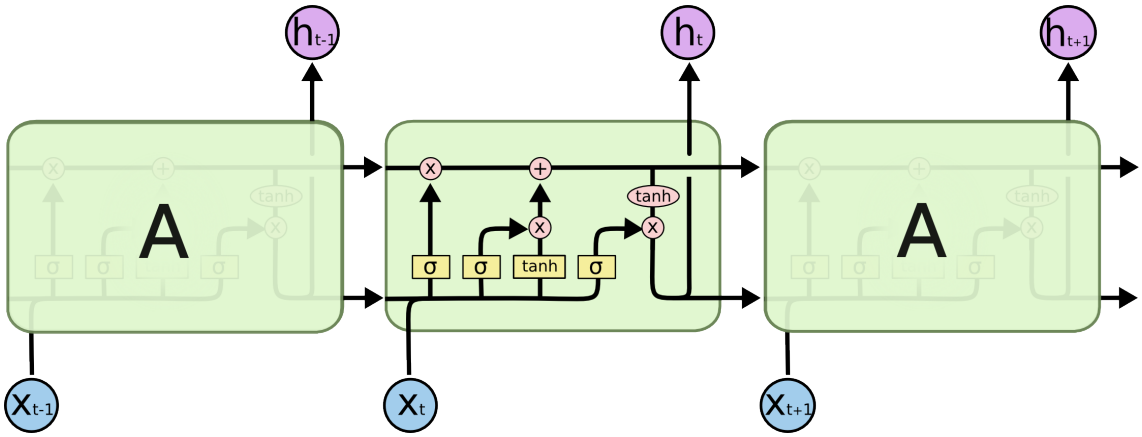
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.4.4.2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.4.4.3}$$

$$\hat{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.4.4.4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{3.4.4.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.4.4.6}$$



Σχήμα 3.6: *LSTM example*

Below, we describe the function of each equation:

Updating the cell state c_t: The forget gate f_t determines which information from the cell state should be retained or discarded. The current input x_t and the hidden state from the previous time step h_{t-1} are multiplied by the respective weight matrices, summed with a bias term, and passed through a sigmoid function. This function outputs values in the range (0, 1), where values close to zero suggest forgetting the information, and values close to one suggest retaining it (Equation 3.4.4.1). The result is then multiplied with the cell state c_{t-1} (Equation 3.4.4.5).

While the forget gate determines what to discard, the input gate i_t decides what new information should be stored in the cell state. The current input x_t and hidden state h_{t-1} are processed through a sigmoid function, limiting the values to (0, 1) (Equation 3.4.4.2). The input gate decides which values from the candidate vector $\hat{c}_t$, generated by passing the input and hidden state through a `tanh` function (Equation 3.4.4.4), will be stored.

The final cell state c_t is updated by combining the retained information from the previous cell state and the new information to be stored, as determined by the input gate (Equation 3.4.4.5).

Updating the hidden state h_t: The output gate o_t controls what information from the cell state c_t should be passed to the hidden state h_t. The gate's value is computed using a sigmoid function, which takes as input the current input x_t and the hidden state h_{t-1} (Equation 3.4.4.3). The updated cell state c_t is then passed through a `tanh` function, and the result is multiplied by the output gate's value to produce the hidden state h_t (Equation 3.4.4.6), which also serves as the LSTM's output.

3.4.5 Attention Mechanisms

In case of long sequences, LSTMs fail to remember information because of encoding of the entire sequence into one vector (the last hidden state). [31] provides a solution to this problem through Attention Mechanisms. Attention mechanisms allow the model to focus on the most important parts of the input. In an encoder-decoder architecture with RNNs, the context vector c_t for the decoder at time step t is computed as:

$$c_t = \sum_j a_{tj} h_j$$

where h_j is the encoder's hidden state at time step j, and a_{tj} represents the attention weights that determine the significance of each encoder output for calculating the context vector. The attention weights a_{tj} are calculated using the softmax function:

$$a_{tj} = \text{softmax}(e_{tj}) = \frac{\exp(e_{tj})}{\sum_k \exp(e_{tk})}$$

The alignment score e_{tj} is a function of the previous decoder state s_{t-1} and the encoder hidden state h_j, defined as:

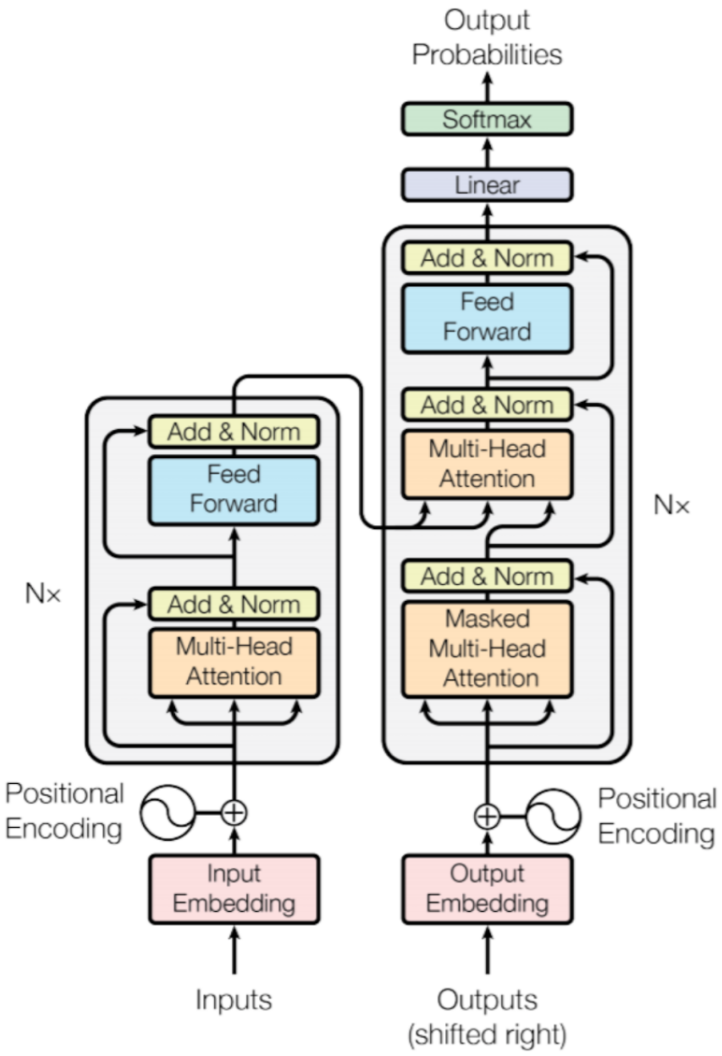
$$e_{tj} = f(s_{t-1}, h_j)$$

The alignment function f, often implemented as a small feedforward neural network, computes the relevance between a query s_t and a key h_i in attention mechanisms. Common types include: Additive Attention, $f(s_t, h_i) = v_a^\top \tanh(W_a[s_t; h_i])$, combining query and key with a non-linearity; Dot-Product Attention, $f(s_t, h_i) = s_t^\top h_i$, a simple dot product; Scaled Dot-Product Attention, $f(s_t, h_i) = \frac{s_t^\top h_i}{\sqrt{n}}$, normalizing for dimensionality; and General Attention, $f(s_t, h_i) = s_t^\top W_a h_i$, introducing a learned weight matrix W_a for added flexibility.

Attention mechanisms can be categorized into several key types based on their roles and functionality. Self-attention allows a model to relate different parts within a sequence, in order to compute its representation. Multi-head attention is an extension of self-attention, where multiple attention mechanisms (or "heads") operate in parallel, each focusing on different parts of the input. This allows the model to capture various aspects of relationships in the data. Cross-attention refers to attention between two sequences, such as between the encoder and decoder in machine translation, where the decoder attends to the encoder's hidden states. Each of these mechanisms enables more flexible and accurate representation learning in models like Transformers.

3.4.6 Transformers

[32] introduced a novel model architecture called the Transformer, which is built entirely on attention mechanisms, without relying on recurrence or convolution. This design allows for more parallelization and significantly faster training. The Transformer model achieved state-of-the-art results in translation tasks, revolutionizing the field of natural language processing by demonstrating the power of attention mechanisms for efficiently capturing long-range dependencies in sequences.



Σχήμα 3.7: *Transformer*

Transformers utilize an autoregressive encoder-decoder architecture. The encoder maps an input sequence x to a sequential representation z and the decoder leverages z to produce the output sequence y.

The **Encoder** consists of multiple layers, each consisting of two sub-components. The first sub-layer applies a multi-head self-attention mechanism, while the second sub-layer is a fully connected feed-forward neural network. Both are followed by layer normalization, with residual connections added between the input and output of each layer to improve information flow.

The **Decoder** mirrors this structure with repeating layers but has three key components per module. First, a masked multi-head self-attention mechanism is used to ensure that each position in the sequence attends only to earlier positions. The second component is a multi-head attention layer that focuses on the encoder's output, followed by a fully connected feed-forward neural network. As with the encoder, layer normalization is applied after each of the three layers.

Positional encodings aim to encode the sequential order of the input sequence via

utilizing the sine and cosine functions in order to calculate a representation of dimension d_m, equal to the model's representations, summed together with the latter.

3.5 Machine Learning Concepts

In the following sections, we explore some fundamental concepts of Machine Learning.

3.5.1 Generalization, overfitting and underfitting

Generalization refers to a machine learning model's ability to perform well on unseen data that was not part of the training process. A well-generalized model effectively captures the underlying patterns in the data, allowing it to make accurate predictions on new inputs. However, if a model is too simple or not properly trained, it can lead to underfitting, where it fails to learn the relationships between the input features and the output, resulting in poor performance on both the training and test data. On the other hand, if the model learns the training data too closely, modeling properties of the training set such as noise that do not occur in the test set, instead of modeling the general data distribution it may overfit. Overfitting causes the model to perform well on the training set but poorly on unseen test data, as it fails to generalize to new instances. The key to successful machine learning is finding the right balance between underfitting and overfitting to ensure good generalization.

To prevent overfitting, various techniques have been developed. Among the most commonly used methods are regularization, dropout, and early stopping. These methods are described below:

Regularization: Regularization helps reduce overfitting by encouraging the model to learn smaller weights, which results in a less flexible model that does not conform too closely to the training data. This is achieved by adding a penalty term (regularization term) to the loss function, which imposes a higher penalty for larger parameter values. For a model with parameters w and a loss function $L(w)$, the regularized loss function is expressed as:

$$\tilde{L}(w) = L(w) + a\Omega(w)$$

where $\Omega(w)$ is the norm penalty term, and a is a hyperparameter that determines how much influence the penalty term has on the overall loss.

Different regularization methods are defined based on the type of norm used in the penalty term. Two of the most widely used methods are:

- **L1 Regularization:** Also known as Lasso regression, L1 regularization uses the L1 norm (or Manhattan distance), defined as:

$$\Omega(w) = \sum_{i=1}^{N} |w_i|$$

  where w_i are the model's parameters.

- **L2 Regularization:** Also referred to as weight decay or ridge regression, L2 regularization uses the L2 norm (or Euclidean distance), defined as:

$$\Omega(w) = \sum_{i=1}^{N} w_i^2$$

where w_i are the model's parameters.

L2 regularization penalizes large parameter values more heavily due to the squared terms, while it affects smaller values less. This reduces model complexity but does not force parameters to zero, unlike L1 regularization, which affects all parameters more uniformly and can push some to zero, creating sparser models. As a result, L1 regularization can also be used for feature selection, as parameters driven to zero effectively remove their corresponding features [20].

Dropout: Dropout is another commonly used regularization technique, especially in neural networks, to prevent overfitting. During training, dropout randomly sets a fraction of the neuron outputs to zero for each training example, with a probability p. This process can be viewed as training multiple different networks, each with some neurons "dropped." By doing this, dropout prevents the model from depending too much on specific neurons, promoting better generalization.

Early Stopping: During training, both the training error and validation error (the error on a separate validation set) usually decrease. However, after a certain point, the validation error may start to increase as the model begins to overfit the training data. Early stopping mitigates this by monitoring the validation error and saving the model parameters each time the validation error improves. When the validation error stops improving for a predefined number of epochs, training is halted, and the model with the lowest validation error is selected as the final model. This method prevents overfitting by stopping training at the optimal point.

3.5.2 Evaluation metrics

Evaluating a machine learning model's performance is vital, as standard loss functions often don't meet the needs of specific applications. In medical applications, for example, it's crucial to detect all true cases (true positives) to avoid serious health risks, even if it increases false positives.

In binary classification, the positive class is the class that the model is designed to detect, while the negative class represents the other.

- **True Positives (TP)**: The model correctly predicts the positive class.
- **True Negatives (TN)**: The model correctly predicts the negative class.
- **False Positives (FP)**: The model incorrectly predicts the positive class.
- **False Negatives (FN)**: The model incorrectly predicts the negative class.

From these, we derive the following performance metrics for binary classification:

Accuracy: Accuracy is the ratio of correct predictions to the total number of predictions, and is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

While it measures overall model performance, it may not be suitable for imbalanced datasets or cases where one class is more important than the other.

Precision: Precision measures how many of the instances predicted as positive are actually positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision is important when the cost of false positives is high.

Recall: Recall, or sensitivity, measures how many of the actual positive instances were correctly identified by the model:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall is crucial when missing true positive cases (false negatives) is costly.

F1-Score: F1-score is the harmonic mean of precision and recall, and provides a balanced measure:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Alternatively, in terms of TP, FP, and FN:

$$\text{F1-score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

In multi-class classification, accuracy is extended as the proportion of correct predictions across all classes. Precision, recall, and F1-score, however, are inherently binary metrics, so for multi-class problems, we treat each class as "positive" and others as "negative," calculating these metrics for each class individually.

3.5.3 Transfer Learning

Transfer learning is a machine learning technique that seeks to apply knowledge gained from one task to enhance performance on another, typically related, task. In transfer learning, a model or a portion of a model that has been trained on a particular task is used as an initial point for training a model on a different task. This approach is especially advantageous when the target task has limited data, but there is a related task with substantially more data. In such cases, the features and representations learned from the first task can be beneficial for the target task, enabling the model to generalize better or train more efficiently, even with a small number of examples for the target task.

Κεφάλαιο 4

Background: Word Representation and Large Language Models

The ability for computers to understand human language is pivotal, as it transforms the way we interact with technology, allowing communication to flow naturally and effortlessly. This capability not only enhances everyday tasks, such as virtual assistance and real-time translation, but also unlocks vast potential in understanding the wealth of unstructured data—turning scattered information into meaningful insights. By bridging the divide between human intention and computational precision, language understanding empowers technology to respond to our needs more intelligently, fostering a seamless synergy between human creativity and machine capability.

Using raw words as input for computations can be quite inefficient. One reason is the variation in word lengths, making it difficult for machine learning models to handle them. Another issue is the sparsity of this representation, as many possible character combinations do not form valid words. To address this, words are typically converted into numerical representations before being used as input for machine learning algorithms.

In natural language, words rarely stand alone—they form part of sentences, which then combine to create paragraphs and larger texts. The meaning of a word is strongly shaped by its position in a sentence and the words that surround it. Thus, understanding the grammatical and contextual relationships between words is essential for grasping the intended meaning of any text. As a result, simply representing individual words falls short for effective Natural Language Processing (NLP); instead, Language Models (LMs) are needed to capture and interpret word sequences in their full context.

In the following sections, we will delve deeper into these concepts, exploring the challenges of processing natural language, how numerical representations of words are used in machine learning, and the role of Language Models (LMs) in capturing the contextual relationships between words.

4.1 Word representation methods

In the following subsections, we will explore some of the most widely used word representation techniques, ranging from traditional methods like one-hot encoding to more advanced approaches based on distributional semantics, such as GloVe and ELMo.

4.1.1 TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) is a measure of the importance of a term to a document, in a collection of documents. It is the product of two statistics, term frequency and inverse document frequency.

Term frequency (TF) [33] is a measure of how often a term t appears within a document d. The simplest way to define term frequency is by counting the number of occurrences of a term in the document.

IDF [34] is derived from the following formula:

$$\text{IDF}(i) = \log\left(\frac{D}{d_i}\right),$$

where D is the total number of given documents, and d_i is the number of documents in which term i appears. By combining TF and IDF, we obtain the TF-IDF measure. For a word w, a collection of documents D, and a document d within that collection, TF-IDF is defined as:

$$\text{TF-IDF}(w, d, D) = \text{TF}(w, d) \times \text{IDF}(w, D)$$

The higher the TF-IDF value for a word w, the more significant that word is in document d within the collection D. TF-IDF is straightforward and highly efficient to compute. However, it does not capture the semantic meaning or syntactic properties of words, and it ignores the word order within a document, often leading to less meaningful representations [34]. Moreover, because a separate TF-IDF value is calculated for each word-document pair, it creates a high-dimensional representation, reducing storage efficiency and increasing the computational cost of working with TF-IDF vectors.

4.1.2 Categorical word representation

Categorical word representation techniques map words to basic representations such as numerical values or vectors. Two widely used methods in this category are label encoding and one-hot encoding.

Label Encoding:
In label encoding [35], each word in a vocabulary V is assigned a unique numerical value.

Label encoding, while straightforward, often performs poorly for text data because it imposes an arbitrary order on words, which isn't meaningful for non-ordinal data.

One-Hot Encoding:
One-hot encoding maps each word w in a vocabulary V of size $|V|$ to a binary vector of dimension $|V|$. In this vector, only the element corresponding to the word's index in the vocabulary is set to 1, while all other elements are 0 [34]. While one-hot encoding is computationally efficient, it has several limitations. It does not encode word similarities, as all words are equidistant from each other, thus lacking meaningful semantic or syntactic information [36, 35]. Moreover, with a large vocabulary, one-hot vectors become high-dimensional, leading to increased computational cost and complexity. This can result in a high number of parameters in neural network models, potentially causing overfitting.

4.1.3 Non-contextual word embeddings

Distributional representation methods are derived from the research area of distributional semantics, which is based on the idea that the words that occur in similar contexts have a similar meaning [37]. Based on distributional semantics, word embeddings have led to significant improvements in NLP tasks. Word embeddings are dense, low-dimensional vectors learned from the context in which words appear, capturing word similarity while being more efficient than sparse representations like one-hot encoding or TF-IDF.

Several methods have been proposed for computing word embeddings, often utilizing deep learning models. Among the most influential are Word2Vec and GloVe.

Word2Vec: Word2Vec [38] is a shallow neural network with two hidden layers. It calculates dense word vectors based on the context in which words appear, and these vectors capture linguistic patterns and word similarity. Word2Vec uses two main models: Continuous Bag of Words (CBOW) and Skip-gram.

- *CBOW:* CBOW predicts a center word from its surrounding context. It has an input layer, a hidden layer, and a softmax output layer. The hidden layer averages the context words' representations to predict the target word.

- *Skip-gram:* Skip-gram predicts context words from a given target word. It adjusts its representations using back-propagation based on the correlation between the predicted and actual context words [37, 39, 40].

GloVe: GloVe (Global Vectors for Word Representation) [41] uses aggregated global word co-occurrence statistics to learn word vectors. Unlike Word2Vec, which relies on local context through a sliding window, GloVe captures global statistics from the entire corpus. It is based on the idea that word co-occurrence ratios encode meaningful relationships, and it trains word vectors such that their dot product approximates the logarithm of the words' co-occurrence probability.

4.1.4 Contextual word representations

While embeddings like Word2Vec and GloVe consider context to generate word embeddings, the word vectors they produce remain fixed. This means that a word is represented the same way in all situations, regardless of the particular context or the word's semantic and syntactic features in that context. [42] provides a solution to this problem by introducing ELMO.

ELMO is composed of three key components: a convolutional neural network (CNN), a bidirectional LSTM, and an embedding layer. It takes character-based input, where character embeddings are passed through CNN layers that capture statistical, temporal, and spatial features. The CNN outputs are then fed into a two-layer bidirectional LSTM, which encodes sequential information by processing the input sequence in both forward and backward directions through its hidden states. The final word embeddings are generated by taking a linear combination of all the hidden states from the bidirectional

LSTM, allowing the model to capture and integrate the various types of information encoded at different layers of the LSTM. The contribution of each hidden state to the final embeddings is task-specific, allowing for improved performance by prioritizing relevant information [42].

4.2 Language Models

Language models are models that assign probabilities to sequences of words [9]. Historically, early language models relied on statistical methods, such as n-grams and Hidden Markov Models, which estimated probabilities based on word and phrase frequencies. In more recent approaches, deep neural networks are employed, with Large Language Models (LLMs) revolutionizing natural language processing and understanding. In this section, we provide an analysis of the most popular traditional and modern language models.

4.2.1 N-grams

One of the earliest and most fundamental approaches to language modeling is the **n-gram model**. An n-gram represents a contiguous sequence of n words, where a 2-gram (or bigram) refers to a pair of consecutive words, a 3-gram refers to a sequence of three consecutive words, and so on. These models estimate the probability of a word occurring based on the previous $n-1$ words in a sequence, relying on statistical data from large corpora. This is commonly achieved through the use of frequency counts from the corpus.

To model the probability of a word sequence more effectively, the n-gram model adopts the **Markov assumption**, which simplifies the computation by assuming that the probability of a word depends only on the preceding $n-1$ words, rather than on the entire sequence of prior words. This assumption allows for more manageable calculations, as it reduces the complexity of the dependencies between words.

The n-gram model computes the probability of a sequence of k words as:

$$P(w_1, w_2, \ldots, w_k) = P(w_1) \prod_{i=2}^{k} P(w_i \mid w_{i-n+1}, \ldots, w_{i-1})$$

This formula represents the probability of the sequence $w_1, w_2, \ldots, w_k$ occurring in context, under the assumption that each word w_i depends only on the previous $n-1$ words.

For example, in the case of a bigram model ($n = 2$), the probability of a sequence of k words simplifies to:

$$P(w_1, w_2, \ldots, w_k) = P(w_1) \prod_{i=2}^{k} P(w_i \mid w_{i-1})$$

Here, the probability of each word w_i is conditioned only on the word immediately preceding it, w_{i-1}. This considerably simplifies the calculation by assuming that a word depends only on its immediate predecessor.

To further illustrate, take a large corpus, count the number of occurrences of the sequence “The water of Walden Pond is so beautifully,” and then count the number of times this sequence is followed by the word "blue." This would answer the question, "Out of the occurrences of the sequence h, how often was it followed by the word w?" Formally, this is represented as:

$$P(\text{blue} \mid \text{The water of Walden Pond is so beautifully}) = \frac{C(\text{The water of Walden Pond is so beautifully blue})}{C(\text{The water of Walden Pond is so beautifully})}$$

In this case, C(The water of Walden Pond is so beautifully blue) refers to the count of the full sequence including the word "blue," while C(The water of Walden Pond is so beautifully) represents the count of the sequence without the final word.

This is an example of how n-gram models estimate the likelihood of a word given its preceding context. More generally, the n-gram model computes the conditional probability of a word w_n given the preceding words as:

$$P(w_n \mid w_{n-1}, w_{n-2}, \ldots, w_1) = \frac{C(w_1 w_2 \ldots w_n)}{C(w_1 w_2 \ldots w_{n-1})}$$

where $C(w_1 w_2 \ldots w_n)$ represents the count of the entire sequence of n words, and $C(w_1 w_2 \ldots w_{n-1})$ is the count of the first $n-1$ words [9].

4.2.2 Neural language models

With the advent of neural networks, a new generation of language models emerged, which leverage deep learning techniques to capture more complex patterns and dependencies in language, going beyond the limitations of traditional n-gram models by utilizing architectures such as feed-forward neural networks (FFNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and more recently, transformers, enabling these models to understand and generate text with greater fluency and contextual awareness. In this section, we will delve into these advanced neural language models, exploring how they function, their underlying architectures, and the ways in which they have revolutionized natural language processing by enabling more accurate and contextually-aware predictions and text generation.

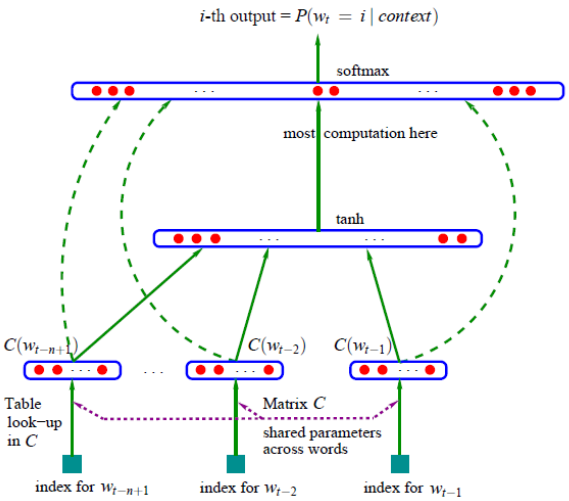
FFNN-based Language Models

One of the earliest works to advocate for the use of neural networks in language modeling was [43]. A single-layer neural network, with no hidden layers, was introduced, where both the input and output dimensions were set to match the size of the vocabulary. In this model, each input word w_i was represented by setting the corresponding i-th input unit to 1. Their approach demonstrated improved results over traditional N-gram models. However, despite these gains, the model suffered from poor generalization capabilities and struggled to capture context-dependent features [44].

A more prominent contribution to the use of neural networks for language modeling came a few years later. [1] introduced the concept of representing words as learnable

embeddings, stored in a look-up matrix of dimensions $|V| \times m$, where $|V|$ is the size of the vocabulary and m is the dimensionality of the word embeddings. The model processes a window of previous words leading up to the current word, retrieving the corresponding embeddings from the look-up table. These embeddings are then concatenated into a single vector and passed through a hidden layer before reaching the final softmax output layer.

While the feed-forward neural network language model (FFNNLM) proposed by [1] showed a significant boost in performance over traditional methods, it also introduced some key limitations. The most notable of these is the fixed-size context window, which is predefined during training and restricts the model to a limited scope of preceding words. Furthermore, the feed-forward architecture requires learning a large number of parameters, making the model computationally expensive and difficult to scale for larger contexts or vocabularies.



Σχήμα 4.1: *The FFNN-LM proposed by [1].*

RNN-LSTM-based Language Models

In order to overcome the limitations of FFNN-Language Models, RNN-Language Models were later proposed. Unlike the former, these models are capable of modeling sequential information, such as word sequences of arbitrary length. They do not require a fixed window and reduce the number of parameters that need to be learned during training, due to parameter sharing.

[45] and [46] are among the most representative contributions in the development of Recurrent Neural Network Language Models (RNNLMs). The core architecture of the model comprises three layers: an input layer, a hidden layer, and an output layer. At each time step t, the model takes as input a concatenation of the vector representation of the current word and the hidden state from the previous time step. This allows the model to maintain a memory of past information and capture dependencies across time steps, making it particularly suitable for handling sequences of arbitrary length. The hidden layer processes this input and generates an updated hidden state, which is passed on

to the next time step, while the output layer produces a probability distribution over the vocabulary for the next word in the sequence. The recursive nature of RNNs allows for the modeling of long-range dependencies in sequential data, addressing one of the primary limitations of feed-forward neural networks, which rely on a fixed context window.

Despite these improvements, the initial RNNLM still faced challenges in handling very long-term dependencies due to issues like vanishing gradients, which were later addressed with LSTM-Language Models.

4.2.3 Large-scale pre-trained language models

With the rise of deep learning, deeper neural networks have proven effective across various tasks. While these models are often trained with supervision, unsupervised training is commonly used in NLP due to the high cost of data annotation and the abundance of unlabeled data. Given their many parameters and extensive training needs, large datasets are essential. Using large-scale unlabeled datasets allows these models to develop a broad understanding of language, which can then be fine-tuned for specific tasks.

Since these corpora are not labeled, traditional supervised learning techniques cannot be applied directly. Therefore, it is necessary to design alternative tasks that enable models to learn in unsupervised or self-supervised settings. Some of the key tasks include Language Modeling, where the goal is to predict the probability of tokens within a given context, Masked Language Modeling, which involves masking certain tokens in the input sequence and having the model predict them using the surrounding words, and Next Sentence Prediction, where the model is tasked with determining if a given sentence logically follows the preceding one in the corpus. Through these methods, language models are able to develop a rich understanding of language and extract meaningful representations from raw text.

Once pre-trained, these models can be further adapted to specific tasks using a second phase of training in a supervised learning setting with smaller labeled datasets, tailored to the particular task at hand.

The combination of this approach, along with the development of the attention mechanism and the introduction of the transformer architecture, has led to the rise of large-scale pre-trained language models. These models are initially trained using unsupervised or self-supervised objectives on massive text corpora, often containing billions of words, and are then fine-tuned for individual tasks using smaller, task-specific datasets (see Section 3.4 for more details on fine-tuning methods). The most well-known model that laid the foundation for many subsequent advancements in the field is GPT.

GPT

GPT-1 : The first Generative Pre-trained Transformer (GPT-1) was developed by OpenAI in 2018 [2]. In this work, the concept of using unlabeled data to train a generative language model was introduced, which could then be fine-tuned for specific downstream NLP tasks. GPT-1 features a 12-layer transformer decoder with masked self-attention,

preventing the model from accessing future context words to the right of the current word during training. It was trained on the extensive BooksCorpus dataset, enabling it to capture long-range dependencies and gain comprehensive knowledge from large, contiguous text sequences.

In the pre-training phase, GPT-1 was optimized using gradient descent, with the objective of maximizing the likelihood for a standard language modeling task:

$$L_1(U) = \sum_i \log(P(u_i \mid u_{i-k}, \ldots, u_{i-1}; \Theta))$$

where k represents the size of the context window, P is the conditional probability modeled by the transformer decoder with parameters Θ, and U is the unlabeled corpus used for training.

During the fine-tuning phase, the model was adapted to a supervised setting using a labeled dataset C, where each sample consists of input tokens $x_1, \ldots, x_m$ and a corresponding label y. The input tokens are processed by the model's decoder layers, and the output of the final layer h_m^l is passed through a linear classification layer with parameters W_y, which computes the probability as:

$$P(y \mid x_1, \ldots, x_m) = \text{softmax}(h_m^l W_y)$$

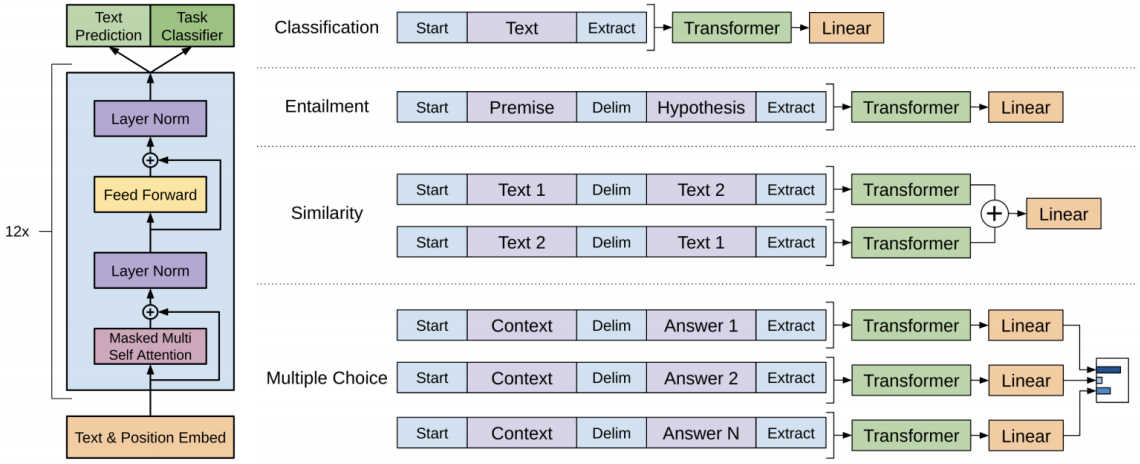
The goal during fine-tuning is to maximize the following objective:

$$L_2(C) = \sum_{x,y} \log P(y \mid x_1, \ldots, x_m)$$

This objective is combined with the original language modeling objective L_1, as the authors found that this improves the model's ability to generalize and speeds up convergence. The final training objective during fine-tuning is thus formulated as:

$$L_3(C) = L_2(C) + \lambda \cdot L_1(C)$$

where λ is set to 0.5. GPT-1 consists of 117M parameters.



Σχήμα 4.2: *GPT-1 [2].*

GPT-2 : In 2019, OpenAI introduced GPT-2 [47], a model that greatly expanded on GPT-1 by increasing the number of parameters to 1.5 billion, compared to GPT-1's 117 million. The layers increased from 12 to 48 layers. GPT-2 was also trained on a larger and more diverse dataset (WebText corpus), enabling it to outperform GPT-1, particularly in language comprehension and zero-shot learning tasks.

GPT-3 : In 2020, OpenAI introduced the third iteration of the Generative Pre-trained Transformer, GPT-3 [48]. GPT-3 was developed with the goal of creating a highly capable and efficient language model that could perform various tasks without the need for extensive fine-tuning, simply by observing a few examples. This version significantly outscaled its predecessors, containing 175 billion parameters and 96 transformer decoder layers, compared to the 12 layers in GPT-1 and 48 in GPT-2. Trained on an enormous dataset sourced from the internet, GPT-3 is recognized for its remarkable ability to generate human-like text and tackle tasks outside of its direct training, such as solving basic arithmetic problems and even writing code.

GPT-3.5 : Released in 2022, GPT-3.5 is an incremental improvement over GPT-3 [49]. While maintaining the same architecture and 175 billion parameters as GPT-3, GPT-3.5 was trained on a larger and more diverse dataset, further improving its performance, particularly in few-shot and zero-shot learning tasks. This update made the model more robust in generating coherent and contextually accurate text, with improved handling of complex tasks.

GPT-3.5 Turbo : GPT-3.5 Turbo, introduced later in 2022, is an optimized and more cost-efficient version of GPT-3.5. While it shares the same capabilities and model architecture, it is designed to run faster and more efficiently, making it ideal for production environments such as ChatGPT. GPT-3.5 Turbo offers similar performance to GPT-3.5 but at a lower computational cost, making it more suitable for real-time applications.

LLaMA

LLaMA [50] is a collection of foundation language models, developed with a range of model sizes from 7 billion (7B) to 65 billion (65B) parameters. These models were trained on trillions of tokens, demonstrating that state-of-the-art performance can be achieved using exclusively *publicly available datasets*, without the need for proprietary or inaccessible resources. The focus of this work is to train a series of language models that deliver the best possible performance across various inference budgets, achieved by training on a larger volume of tokens than is typically used in similar models. Notably, LLaMA-13B surpasses the performance of GPT-3 (which has 175 billion parameters) on most standard benchmarks, while LLaMA-65B is competitive with leading models such as Chinchilla-70B and PaLM-540B, despite having significantly fewer parameters. This efficiency highlights LLaMA's potential for producing high-quality language representations with optimized computational resources.

Pre-training Data : The training dataset used for LLaMA is composed of a mixture of various publicly available sources, covering a broad range of domains. This approach mirrors data selection strategies used in the training of other large language models but is restricted to datasets that are compatible with open-sourcing. The overall dataset consists of approximately 1.4 trillion tokens after tokenization. In most cases, each token in the dataset is used only once during training, with the exception of data from specific domains such as Wikipedia and Books, where approximately two epochs of training are performed.

Some of the datasets used include English Common Crawl, C4, GitHub repositories, academic articles, Wikipedia, and datasets from the Books domain. These sources provide a diverse and extensive corpus, enabling the model to generalize effectively across a wide variety of tasks and domains.

Architecture: LLaMA's architecture is built on the transformer framework [32], incorporating several enhancements introduced in subsequent models, such as GPT-3 and PaLM. These modifications aim to improve the model's training stability and overall performance.

Key changes include the use of *pre-normalization* [inspired by GPT-3], where the input to each transformer sub-layer is normalized using RMSNorm [51] rather than normalizing the output, which enhances training stability. Additionally, the *SwiGLU activation function* [inspired by PaLM] replaces the traditional ReLU activation. This change, introduced by [52], boosts performance, and the dimensionality is set to $\frac{2}{3}$ of 4d, as opposed to the 4d used in PaLM. Lastly, *rotary positional embeddings* (RoPE) [inspired by GPT-Neo] are employed instead of absolute positional embeddings. This method, proposed by [53], is applied at each layer of the network to enhance the model's handling of positional information.

Llama2 : LLaMA 2 [18] (7B,13B,70B parameters) introduces several key improvements over LLaMA 1, including significant refinements in the pretraining process and data

quality. While continuing to use an optimized auto-regressive transformer architecture, LLaMA 2 benefits from more robust data cleaning and an updated data mix. The training corpus, consisting solely of publicly available sources, was expanded by 40%, with a total of 2 trillion tokens. Additionally, efforts were made to remove data from sources containing a high volume of personal information, aiming to improve the model's factual accuracy and reduce hallucinations. To further enhance performance, LLaMA 2 doubled the model's context length and adopted grouped-query attention (GQA) to improve inference scalability, particularly for larger models. These improvements allow LLaMA 2 to offer better performance and scalability than LLaMA 1.

Mistral

Mistral [54] 7B language model that demonstrates superior performance and efficiency compared to larger models like LLaMA 2 (13B parameters) and LLaMA 1 (34B parameters) on a wide range of benchmarks, including reasoning, mathematics, and code generation tasks. Mistral 7B achieves these improvements through the integration of advanced attention mechanisms, such as grouped-query attention (GQA) and sliding window attention (SWA), both designed to optimize inference speed and memory usage.

The grouped-query attention (GQA) mechanism accelerates inference while reducing memory requirements during decoding, enabling larger batch sizes and improving throughput. This is particularly beneficial for real-time applications. Additionally, sliding window attention (SWA) allows the model to efficiently handle longer sequences by attending to tokens beyond a fixed window size, significantly extending the attention span without a proportional increase in computational cost. For example, with a window size of 4096, Mistral 7B can theoretically attend to up to 131K tokens, achieving up to 2x speed improvements over traditional attention mechanisms.

Mistral 7B is built on a transformer architecture, similar to LLaMA, but introduces several key modifications for improved scalability and efficiency. Notably, the model employs a rolling buffer cache to manage memory usage efficiently by limiting cache growth and overwriting past values as new tokens are processed. This allows for an 8x reduction in cache memory usage when handling sequences of up to 32k tokens, with no significant impact on model quality. Furthermore, the model supports pre-fill and chunking techniques, which allow for efficient generation of long sequences by pre-loading the cache with prompt chunks and computing attention over both the cache and new chunks during inference.

Mixtral

Mixtral 8x7B [55] is a Sparse Mixture of Experts (SMoE) language model that builds upon the architecture of Mistral 7B, incorporating specialized mechanisms to enhance both performance and computational efficiency. Unlike traditional dense models, Mixtral leverages a sparse design in which each layer contains eight distinct feedforward blocks, or "experts." For each token, a router network dynamically selects two experts at each layer to process the token's current state, combining their outputs. As a result, each

token effectively accesses a model with 47 billion parameters, but only uses 13 billion active parameters during inference, offering substantial efficiency improvements while maintaining high performance.

One of Mixtral's key innovations is its use of grouped-query attention (GQA) and sliding window attention (SWA), both of which allow it to handle long sequences of up to 32,000 tokens more efficiently. These mechanisms reduce the computational cost of processing long sequences while maintaining the ability to attend to information spread across large context windows. This allows Mixtral to outperform models like LLaMA 2 (70B) and GPT-3.5 on benchmarks for tasks such as mathematics, code generation, and multilingual understanding.

Mixtral's SMoE architecture introduces significant scalability benefits. The model's sparse design allows it to increase the total parameter count without increasing the computational cost per token. By selecting only two experts per token, Mixtral reduces memory and processing demands, which accelerates inference speed for smaller batch sizes and increases throughput for larger ones. This makes the model well-suited for real-time applications, where efficient inference is critical.

The sparse architecture relies on expert parallelism, which distributes the computation of different experts across multiple GPUs. During inference, tokens are routed to their assigned experts, processed, and then returned to their original locations, allowing for efficient parallel processing. Additionally, Mixtral uses a rolling buffer cache and pre-fill techniques, which further enhance performance by preloading context data into memory, allowing the model to handle long prompts effectively without performance degradation.

Overall, Mixtral 8x7B represents a major leap in scalable language modeling, offering a blend of high performance and efficiency through its sparse mixture of experts architecture. It surpasses models like LLaMA 2 (70B) and GPT-3.5 in key domains, particularly where handling long sequences or complex tasks is required.

4.2.4 Adapting pre-trained language models

Pre-trained language models are built on extensive amounts of unlabeled data, utilizing unsupervised or self-supervised learning approaches to capture a broad understanding of language. Training these models demands substantial computational resources and can take several days due to the sheer size of the datasets and the complexity of the models. However, since the datasets for many downstream NLP tasks are often too small to support training such models from scratch, and The hardware resources needed for large-scale training are often unavailable, a different approach is taken. Instead of training a new model for each task, the pre-trained language models are fine-tuned. This process involves loading the pre-trained weights, which encode general language knowledge, and adapting them to specific tasks using smaller, task-specific datasets. This approach allows for efficient reuse of the models' language understanding while minimizing the computational burden.

Fine-tuning : Fine-tuning has long been the primary approach for adapting pre-trained language models to specific tasks. This method involves loading the pre-trained weights and training the model further with a task-specific dataset for a few more epochs. Typically, the entire set of model parameters is adjusted to minimize the task-related loss. While fine-tuning is relatively simple and has been widely adopted in various studies with positive results, it does come with certain drawbacks. One major issue is that fine-tuned models tend to overfit, particularly when the dataset is small. Additionally, the process is highly resource-intensive, both in terms of computation and memory usage. As pre-trained models grow larger, the memory required during fine-tuning becomes substantial, and storing these fine-tuned models also incurs significant costs. Another challenge is that altering the pre-trained parameters can cause the model to forget valuable information learned during the pre-training phase. Due to these limitations, and with the increasing size of pre-trained models, alternative techniques have been developed in recent years to address the inefficiencies and risks associated with fine-tuning.

Adapter-tuning : Another approach to adapting pre-trained language models involves adding a small number of new parameters. In this method, small residual modules, known as residual adapters, are inserted into the transformer layers of pre-trained models. Originally introduced by [56], these adapters are trained on downstream task datasets while keeping the original model parameters frozen. This method, often referred to as adapter-tuning, has become a popular alternative to fine-tuning [57].

The architecture and placement of adapter modules within transformer layers vary in the literature and can impact performance. For instance, Houlsby et al. [58] proposed inserting adapter modules twice in each transformer layer—once after the multihead attention sub-layer and again after the feed-forward sub-layer. Their design includes two feed-forward layers, a non-linearity, and a bottleneck structure that projects features to a smaller dimension before returning them to their original size. [59], on the other hand, suggest a simpler configuration by placing the adapter only after the feed-forward sub-layer of the transformer.

Adapter-tuning offers comparable performance to fine-tuning while significantly reducing the number of trainable parameters, leading to more efficient storage when training models for multiple downstream tasks. Another key advantage is that adapters enable efficient information sharing across tasks without suffering from issues common in multitask learning, such as catastrophic forgetting or inference. Task-specific adapters can be trained independently and later combined through attention mechanisms, which mitigates these problems and enhances performance.

Prompt-based learning : While previous methods focus on modifying the pre-trained language model itself—either by tuning all parameters or adding new ones—recent years have seen the rise of an alternative approach that leaves the pre-trained model unchanged. Instead, this method modifies the model's input by inserting a small set of additional parameters, referred to as a prompt. These prompts can be added at the text input level or directly within the model's embeddings. As the size of pre-trained models continues to

grow, prompt-based learning has gained popularity due to its cost-effectiveness in terms of both training and storage resources when compared to traditional fine-tuning methods.

Κεφάλαιο 5

Mathematical reasoning using in-context learning

5.1 Definition

Given a set of mathematical problems $P = \{p_1, p_2, \ldots, p_N\}$ with corresponding solutions $S = \{s_1, s_2, \ldots, s_N\}$, the problem of Mathematical Problem Solving (MPS) involves the development or use of a computational methodology capable of interpreting each problem p_i, applying appropriate reasoning and mathematical operations, and producing the correct solution s_i.

5.2 Related Work

5.2.1 Reasoning with LLMs

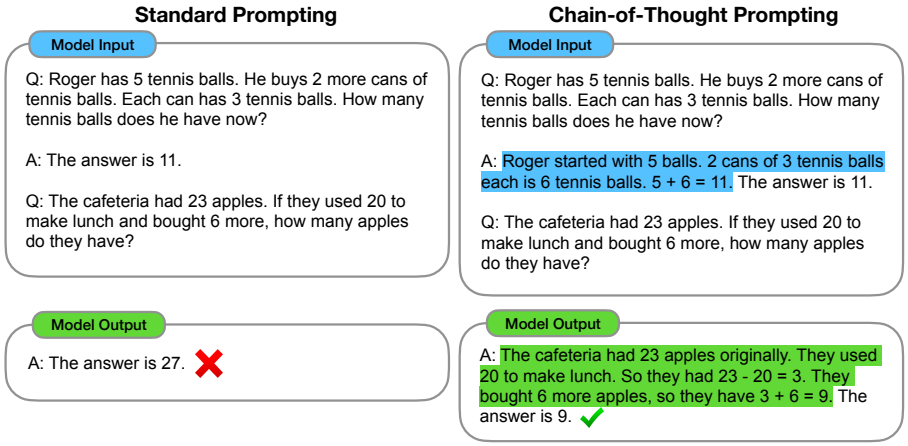
As the domain of Large Language Models (LLMs) flourishes, a variety of prompting strategies have been developed to enhance their reasoning capabilities. Initial achievements include facilitating reasoning through step-by-step chain-of-thought processes [5], utilizing programming to address procedural challenges [6, 7] and employing zero-shot prompts that require only a single sentence to guide LLMs [11]. Furthermore, the Tree-of-Thoughts approach [12] intentionally navigates through various reasoning pathways and traverses tree-shaped structures of reasoning states. Moreover, X of Thoughts [3] selects, applies and verifies the most suitable among the techniques of CoT (Chain of Thought), PoT (Program of Thought), and EoT (Equation of Thought) iteratively, until a correct solution is found.

We will delve deeper into the methods we use as baselines to gain a better understanding of their application and performance.

Chain of Thought (CoT)

Chain of Thought (CoT) prompting [5] is a technique used to enhance the reasoning capabilities of large language models by encouraging them to generate a series of intermediate reasoning steps when solving complex problems. This approach mimics the human thought process of breaking down a multi-step task, such as a math word problem, into smaller, manageable steps before arriving at a final solution. By providing examples of chain of thought reasoning during few-shot prompting, large language models are able to develop the ability to reason step-by-step, which significantly improves their performance on tasks requiring complex reasoning. CoT prompting demonstrates that when large models are given demonstrations of step-by-step reasoning, they can naturally

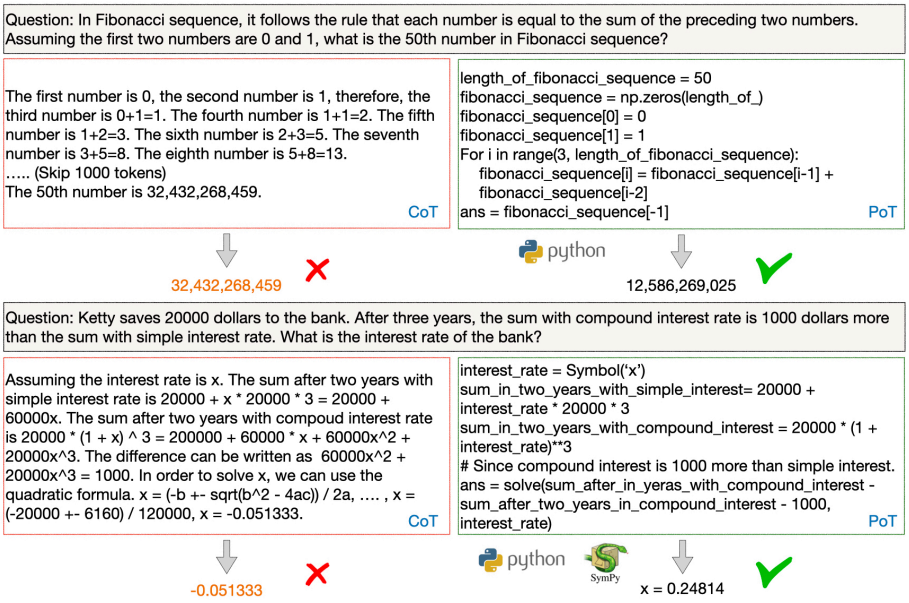
generate coherent reasoning chains that lead to accurate answers. This method has proven particularly effective in solving problems that would otherwise result in incorrect answers without such intermediate reasoning steps.



Σχήμα 5.1: *CoT example*

Program of Thought (PoT)

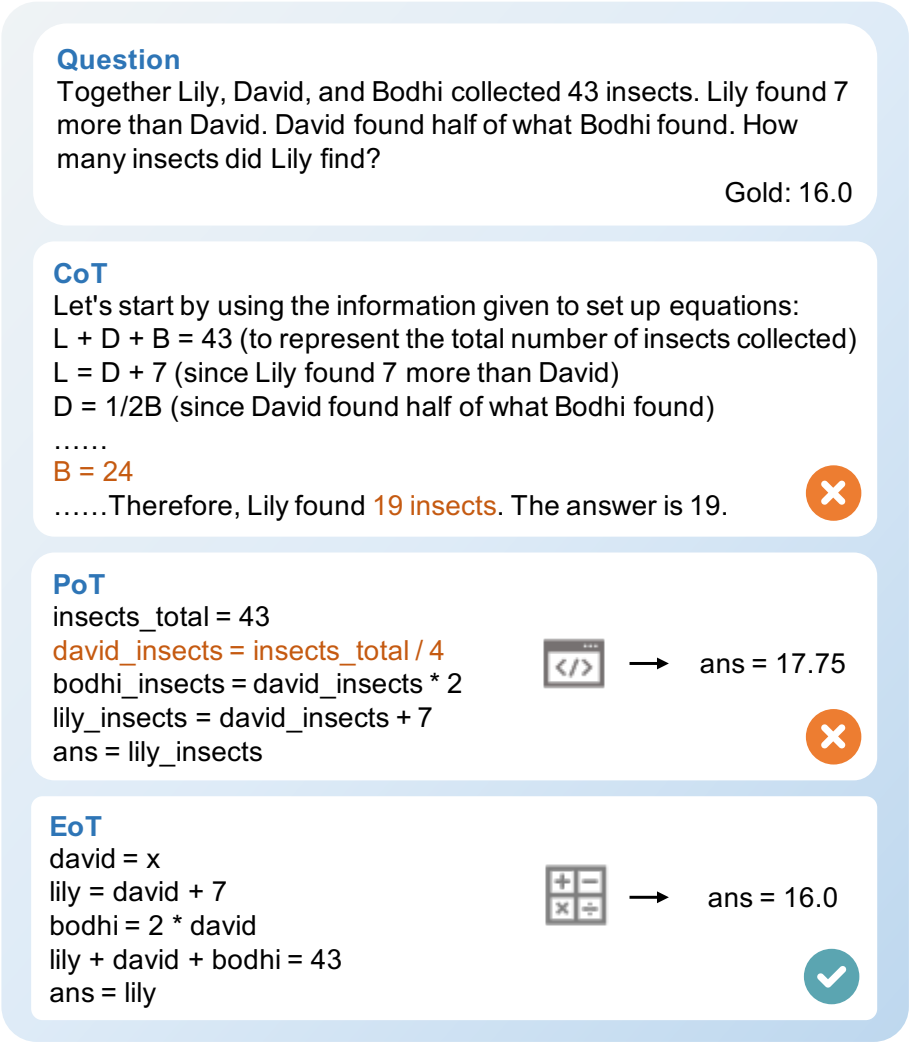
Program of Thoughts (PoT) [7] is an approach where large language models (LLMs) express their reasoning steps as executable Python code. Unlike Chain of Thought (CoT) prompting, which requires LLMs to both reason and compute within the model itself, PoT decouples these tasks by allowing the model to generate Python programs that represent the solution process. The generated Python code is then executed by an external interpreter, such as a Python runtime, to perform the necessary computations. This method addresses the limitations of language models in handling complex arithmetic, iteration, and advanced mathematical problems, which are better suited for programmatic execution. By generating code that can leverage libraries like SymPy for solving equations, PoT achieves greater accuracy and efficiency. In cases where CoT might produce errors or struggle with multiple iterations, PoT expresses these computations concisely as Python code, leading to improved performance. Studies show that PoT results in a performance improvement over CoT across various benchmarks, highlighting the benefits of using executable programs to solve complex tasks.



Σχήμα 5.2: *PoT example*

Equations of Thought (EoT)

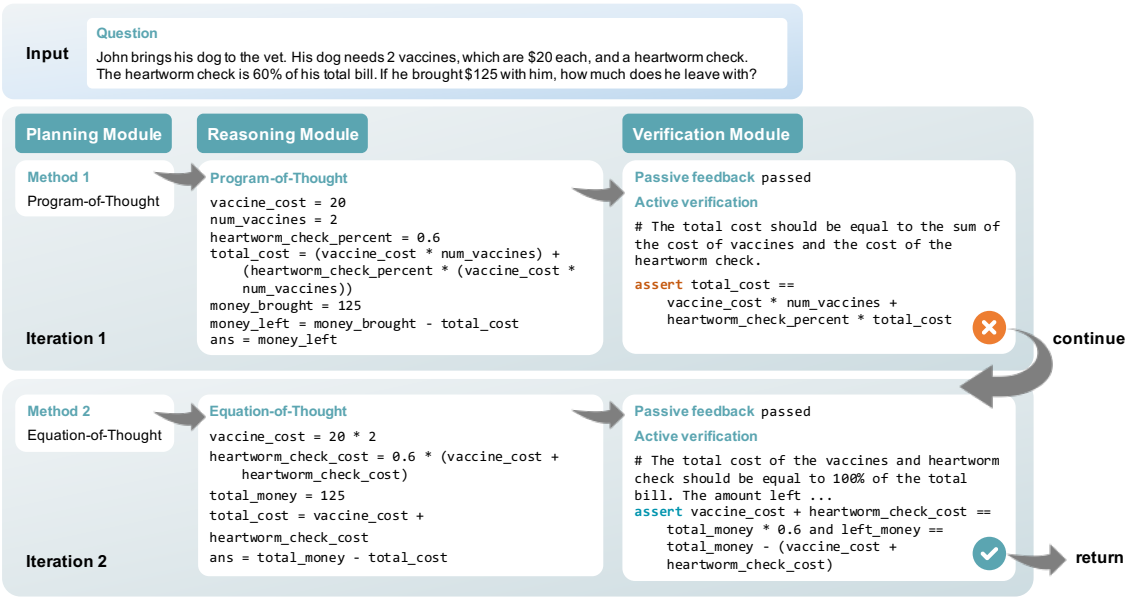
Equation of Thought (EoT) [3] is a prompting technique designed to handle reasoning tasks that involve unknown variables. Unlike Chain of Thought (CoT) and Program of Thoughts (PoT), EoT formulates problems as linear equations, which allows it to incorporate unknown values directly into the problem description. While CoT can correctly set up such equations, it often fails to perform accurate calculations, and PoT struggles to handle unknown variables, as Python requires all variables to be assigned specific values. This limitation can lead to misleading steps in PoT, where incorrect assumptions about unknown variables are made. EoT, in contrast, expresses the problem context clearly as equations and solves them using a deterministic equation solver, avoiding the pitfalls of both CoT and PoT. This makes EoT particularly effective for tasks involving complex reasoning with unknown values.



Σχήμα 5.3: *EoT example*

X of Thoughts (XoT)

X-of-Thought (XoT) [3] is a prompting method designed to improve the performance of large language models (LLMs) in solving mathematical problems by iteratively selecting, applying, and verifying the most suitable technique from Chain of Thought (CoT), Program of Thought (PoT), and Equations of Thought (EoT). The XoT framework follows three key steps: Planning, where the most appropriate technique is chosen for the problem; Reasoning, where the selected method is applied; and Verification, which includes two types of checks. The first, Passive verification, compares the results with those from external tools, while the second, Active verification, involves self-assessment based on the problem's statement. These steps are repeated iteratively until a correct solution is found.



Σχήμα 5.4: *XoT example*

5.2.2 Bloom's Taxonomy

Bloom's Taxonomy is a multi-tiered model of classifying thinking according to six cognitive levels of complexity. In the original version of the Taxonomy, the lowest three levels are: remembering, understanding, and applying. The highest three levels are: analyzing, synthesizing, and evaluating. The taxonomy is hierarchical, as shown in Fig. 5.5, where each level is subsumed by the higher levels. During the 1990s, the taxonomy was revised. The new structure of the taxonomy is: remembering, understanding, applying, analyzing, evaluating, and creating. Our work is based on the revised taxonomy of [10]. The steps used in the Taxonomy are defined as follows:

1. **Remembering:** Retrieving, recognizing, and recalling relevant knowledge from long-term memory.

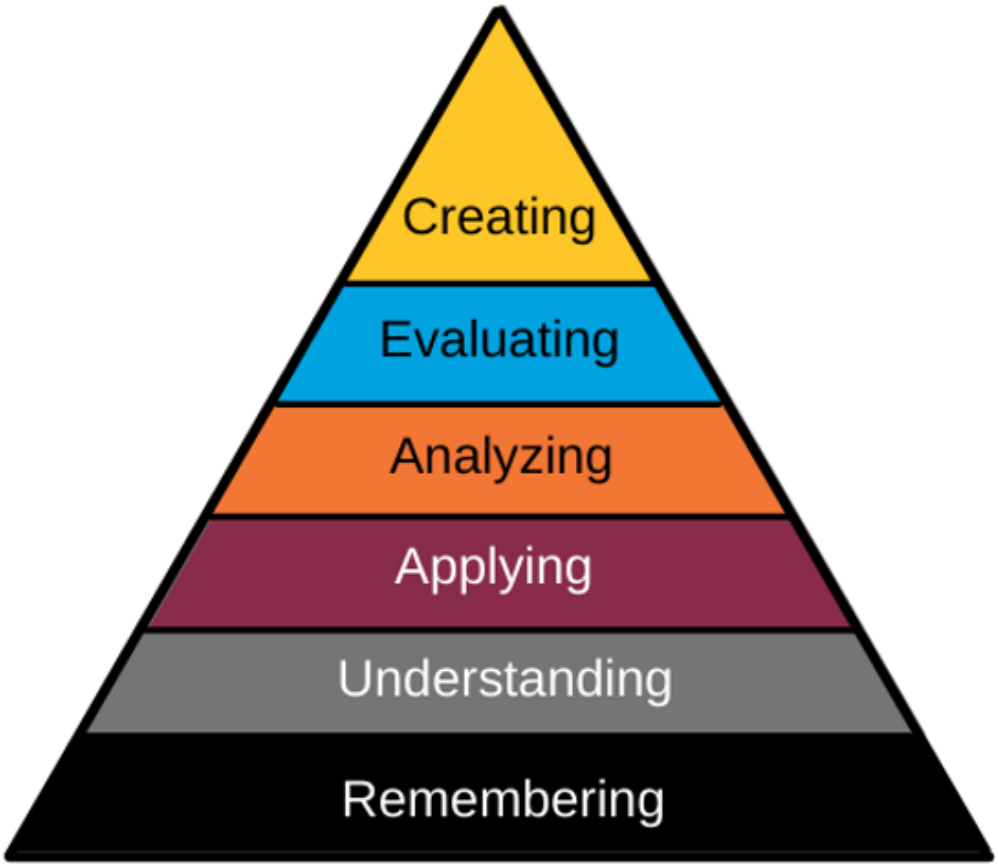
2. **Understanding:** Constructing meaning from oral, written, and graphic messages through interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining.

3. **Applying:** Carrying out or using a procedure through executing, or implementing.

4. **Analyzing:** Breaking material into constituent parts, determining how the parts relate to one another and to an overall structure or purpose through differentiating, organizing, and attributing.

5. **Evaluating:** Making judgments based on criteria and standards through checking and critiquing.

6. **Creating:** Putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing.



Σχήμα 5.5: *Bloom's Taxonomy (Revised)*

5.2.3 Metacognitive Prompting

The term "Metacognition" is used to indicate cognition about cognition, or more informally, thinking about thinking. The American developmental psychologist John H. Flavell [13] defined metacognition as "knowledge about cognition and control of cognition".

This concept was expanded in the field of instruction tuning in LLMs. [14] have introduced Metacognitive prompting, a novel prompting strategy for LLMs rooted in human introspective reasoning. The method was applied to Natural Language Understanding tasks.

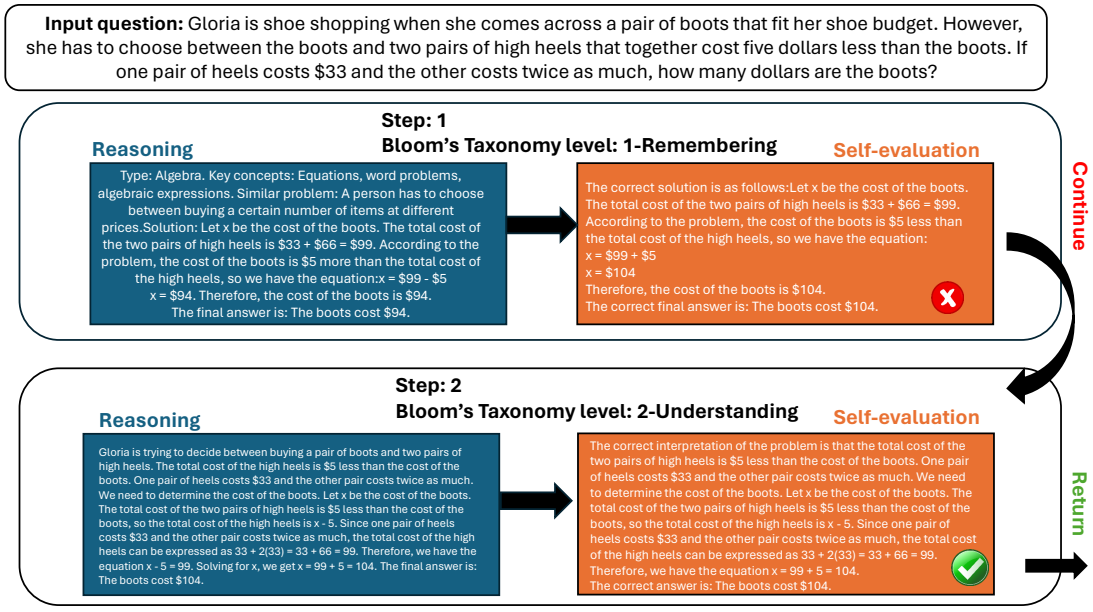
Κεφάλαιο 6

Proposed approach

6.1 Framework

BloomWise

Our goal is to develop a generalized problem solving framework that iterates through prompts, each corresponding to a level of Bloom's taxonomy, until the correct solution is reached. Next, we describe the overall framework and introduce each module in detail.



Σχήμα 6.1: *Overview of BloomWise Early Stop algorithm. The math problem is shown on top, followed by the output at each Bloom taxonomy level (blue box on the left) and the output for the self-evaluation step (orange box on the right). Input prompts are not shown, please refer to Appendix B. Self-evaluation fails at "Remembering" level 1 and the algorithm moves to "Understanding" level 2, where the self-evaluation succeeds. Consequently, the LLM returns the reply from level 2.*

The Framework The overall pipeline is described in Algorithm 6.1.In this framework, the process iterates through the first five[1] levels of Bloom's taxonomy for a given problem. At each level, the language model receives as input the problem and a prompt tailored to that specific level. The response generated by the LLM is then self-evaluated. If the solution is evaluated as correct, success is declared, and the process halts, preventing progression to the next level. Conversely, if the solution is evaluated as incorrect, the

[1]The sixth level of the taxonomy "Creating" could be useful for some tasks that require synthesis but we have found it of little relevance for math problems.

process advances to the next level. This continues until all levels have been attempted. An overview of the framework can be found in Figure 6.1.

ΚΩΔΙΚΑΣ 6.1: *BloomWise Early Stop*

```
Input: i
level ← 1
for l in levels do
    res ← Reasoning(level_prompt, i)
    eval ← SelfEvaluate(res)
    if eval = false and level ≤ 5 then
        level ← level + 1
    else
        break
    end if
end for
return Output
```

Prompts We designed prompts corresponding to each of the first five levels of Bloom's Taxonomy. The detailed prompts are shown in Tables B.1-B.5 in Appendix B[2].

Remembering: read the problem carefully, identify its type and main concepts, recall similar or the exact same problem and reply.

Understanding: summarize the problem and its relevant concepts.

Applying: design and apply a step-by-step methodology to reach a solution.

Analyzing: reach a solution after analyzing your thoughts on how the given problem should be solved.

Evaluating: critically evaluate the solution when expressing it.

Self-evaluation The verification module assesses the effectiveness of the reasoning solution through self-evaluation: The LLM is given the problem and a possible solution that it has generated, and is prompted to reproduce the solution while checking for errors. We have added a disclaimer in the prompt informing the LLM that the solution is probably wrong. An adversarial approach is adopted to raise the chances of identifying errors. It has been observed that requesting rephrasing helps smaller LLMs perform a more accurate evaluation. The self-evaluation prompt and an example reply in shown in Table B.6. Further self-evaluation examples can be found in Figure 6.1.

Iteration and Early Stopping For a given problem, the LLM processes prompts that are structured according to the corresponding level of Bloom's taxonomy. This process continues until either a correct solution is identified or all levels have been attempted. This practice is based on the assumption that a higher level of the taxonomy encourages

[2]Note that the prompts have been designed on purpose to be very basic and simple to gauze the potential of the method; we did not try any prompt engineering.

the LLM to approach the problem in a more in-depth manner compared to a lower level. Additionally, if the prompt corresponding to a specific level leads to a correct solution there is no need for higher cognitive processing for the specific problem and early stopping is reasonable. This process is outlined in Figure 6.1.

Early Stopping using an Oracle There are many ways to decide on which level of the taxonomy to stop. To explore the full potential of our method, we also investigated using an oracle to decide the optimal stopping point. Given a question q, we denote the correctness of the reasoning answers using each prompt as $\hat{R}_s(q) \in 0, 1$ compared to the gold label. The subscript $s \in \mathcal{S}$ corresponds to the reasoning approach of the Bloom taxonomy that is used to answer the question q, i.e., remembering, understanding, applying, analyzing, and evaluating. We define the accuracy under the oracle setting for N questions as:

$$Acc_{oracle} = \frac{1}{N} \sum_{q} \hat{R}(q), \tag{6.1}$$

where

$$\hat{R}(q) = \bigvee_{s \in \mathcal{S}} \hat{R}_s(q).$$

This BloomWise Oracle algorithm will solve correctly a given problem if any of the methods generates the correct answer.

BloomWise Majority Voting As an alternative to early stopping we also investigated an approach where the final output is determined by a majority vote of the outputs corresponding to all five first stages of Bloom's taxonomy. This method leverages the collective decision-making power of multiple cognitive reasoning stages. For a given question q, each reasoning stage $s \in \mathcal{S}$ of Bloom's taxonomy generates a numerical result $R_s(q)$. The majority vote setting determines the final answer, based on the most frequent result among these stages. $\hat{R}_{\text{majority}}(q) \in 0, 1$ compared to the gold label. We define the accuracy under the majority vote setting for N questions using (6.1), where

$$\hat{R}(q) = \hat{R}_{\text{majority}}(q).$$

The majority vote setting represents a consensus-based approach where the model aims to solve a given problem based on the most frequent results from the methods employed. In cases where there is a tie (i.e., two or more answers have the same highest frequency), the first encountered answer among those with the highest frequency is chosen.

6.2 Experimental setting

Datasets Our experiments are conducted on a comprehensive set of four math reasoning datasets, encompassing various challenging math reasoning scenarios: GSM8K, SVAMP, Algebra and GSM-hard (this dataset includes problems from the GSM8K dataset with

Dataset	# Samples
GSM8K [15]	1,319
SVAMP [16]	1,000
Algebra [17]	222
GSM-hard [6]	1,319

Πίνακας 6.1: *Statistics of the datasets we used*

small numerical values replaced by larger ones to increase calculation complexity). The details of the statistics of the datasets can be found in Table 6.1.

Models We query OpenAI API for experiments [3]. Specifically, we use gpt-3.5-turbo for both the reasoning and the self-evaluation modules of our framework. We also experiment with a diverse set of foundation models of different sizes, including a mixture of expert models. The base models are the following: Llama2 (13B and 70B) [18] and Mixtral 8x7B instruct [19].

6.3 Results and discussion

6.3.1 Comparison to state-of-the-art

	Dataset	CoT	PoT	XoT	BloomWise EarlyStop (BLES)	BloomWise Majority Voting (BLM)	BloomWise Oracle (BLO)
GPT 3.5 turbo	GSM8K	80.2	77.2	**83.3**	78.2	82.6	92.5
	SVAMP	79.5	79.5	83.6	84.7	**88.0**	94.8
	Algebra	81.5	62.5	**89.9**	85.1	86.0	92.3
	GSM-hard	42.4	61.8	**63.4**	40.6	44.4	58.0
LLaMA2-70B	GSM8K	**59.1**	52.3	57.9	38.7	56.4	76.8
	SVAMP	75.2	73.6	**77.0**	55.9	76.2	90.0
	Algebra	43.2	35.6	**64.0**	28.4	45.5	67.5
	GSM-hard	27.1	43.1	**43.8**	13.1	20.2	32.7
LLaMA2-13B	GSM8K	27.5	26.4	30.2	29.7	**37.6**	59.9
	SVAMP	40.3	50.4	46.2	43.4	**55.2**	78.6
	Algebra	20.3	26.1	**30.2**	25.2	29.3	53.2
	GSM-hard	8.5	**22.7**	17.4	7.2	9.8	19.7
Mixtral 8x7B Instruct	GSM8K	66.8	13.9	66.0	69.4	**75.4**	88.7
	SVAMP	72.2	6.8	71.2	82.4	**86.2**	92.7
	Algebra	54.9	2.3	54.0	71.1	**71.6**	86.9
	GSM-hard	31.5	8.5	31.4	33.2	**37.7**	50.9

Πίνακας 6.2: *Solution accuracy across various models and math reasoning datasets. The baseline results for Llama2 and GPT 3.5 turbo are taken from [3]. Best performer is shown in bold and second best is underlined (excluding the BloomWise Oracle model).*

In Table 6.2, we report solution accuracy across the four math datasets for the four LLMs we tested. We consider three prompting methods as baselines, namely CoT [5], PoT [7] and XoT [3], and compare with BloomWise Early Stop (BLES), Majority Voting (BLM) and Oracle (BLO). Overall, averaged over all datasets and models, BLM and XoT perform comparably achieving approximately 56.5% accuracy, followed by CoT and BLES

[3] `https://openai.com`

at 50.6% and 49.1%, respectively, while PoT is the worse performer at 40.2%. When using an Oracle, BLO accuracy is significantly higher- at 71% - demonstrating the potential of BloomWise.

Performance per task: BLM is the top performer for the GSM8K and SVAMP tasks, while for Algebra XoT and BLM perform similarly. Specifically for SVAMP average BLM accuracy across the four models is at 76.4% while XoT is at 69.5%. For GSM8K the accuracy is at 63% vs 59.4% for BLM vs XoT. All three BloomWise variants perform relatively poorly on the GSM-hard dataset due to the numerical complexity that characterizes this particular dataset, while XoT performs best followed by PoT. BLO reduces the relative error rate compared to the best performing prompting method (BLM or XoT) by up to 53% for SVAMP, 47% for GSM8K, 30% for Algebra, with no improvement for GSM-hard.

Performance per model: The best performing prompting method is very much model dependent. BLM is the top performer for LLaMA2-13B and mixtral LLMs, while XoT is the top performer for LLaMA2-70B and gpt3.5-turbo. BLES performs well for gpt3.5-turbo and mixtral, but relatively poorly for the LLaMA2 models (see also the next section for a deep dive into early stopping performance). BLO reduces the error rate compared to the best performing prompting method for all models, ranging on average between 15% relative for LLaMA2-70B to 37% for mixtral.

6.3.2 Ablation study and improved self-evaluation

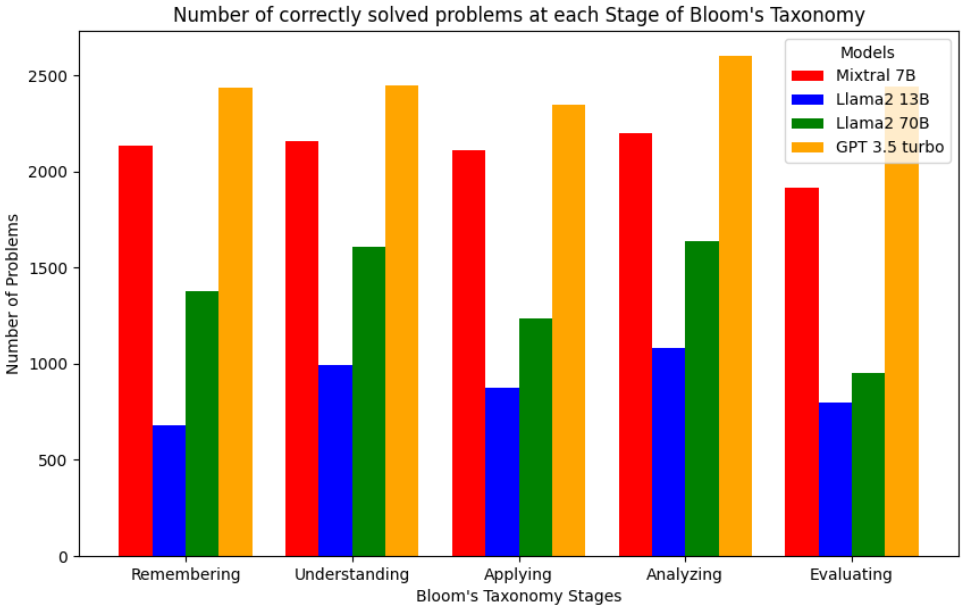
The analysis of the results will be structured along two axes. The first concerns the study and commentary of the results by Bloom taxonomy level. The second focuses on closing the performance gap between the self-evaluation (BLES) and the oracle setting (BLO) of BloomWise by exploring better self-evaluation approaches. Towards this end, we present a variant of our method: the Program of Bloom.

Performance at each cognitive level

LLMs and cognitive abilities In Figure 6.2 we show the number of correctly solved problems per cognitive level for all four LLMs tested. For this analysis, we executed the prompts corresponding to each of the first five stages of the taxonomy (no early stopping). A problem might be correctly solved in more than one stages.

Dataset	Remembering	Understanding	Applying	Analyzing	Evaluating
GSM8K	72.6	73.2	70.2	78.0	73.7
SVAMP	81.6	83.3	76.5	81.8	81.2
Algebra	75.2	74.7	75.2	85.6	80.2
GSM-hard	37.8	36.5	37.0	42.8	36.6

Πίνακας 6.3: *Accuracy per Bloom's Stage for GPT 3.5 turbo.*



Σχήμα 6.2: *Correctly solved problems per Bloom's Level*

From the histogram, we can draw several conclusions about the relationship between the cognitive abilities of an LLM and its size and sophistication.

First, all models achieve their best performance at 'Analyzing', followed by 'Understanding', although their overall performance varies. Despite the intuitive expectation that smaller less sophisticated models might excel at simpler cognitive abilities this is not supported by the data in the histogram. It shows a clear need for deep processing and comprehension of the problem itself for all models, as the 'Understanding' and 'Analyzing' stages emphasize these aspects. This might also be due to the fact that the prompts associated with 'Understanding' and 'Analyzing' are similar to the prompts of traditional prompting methods like CoT, i.e., despite the lack of sophistication of smaller models they have been exposed to such "thinking" patterns during fine-tuning or alignment directly or via distillation from bigger models.

Second, larger LLMs such as GPT-3.5 Turbo, or more sophisticated models like Mixtral, demonstrate a more homogeneous performance across the cognitive levels, indicating a consistent ability to tackle tasks at any level of cognitive complexity. In contrast, smaller or less sophisticated models, such as Llama2 ranging from 13b to 70b, show more variation in their performance across these levels. This behavior reinforces

the belief that a deep understanding of the problem is sine qua non, especially for smaller LLMs, likely because they faithfully apply what is explicitly stated in the prompt. In contrast, larger models are capable of identifying and addressing any gaps in the understanding of the problem before proceeding to the solution, even when such action is not explicitly requested.This suggests that size and sophistication likely contribute to an LLM's capacity to maintain consistent performance across diverse cognitive challenges. This is another indication to the improved generalization abilities of larger models to less familiar prompting methods, e.g., "Applying".

Problem type and cognitive depth For the sake of simplicity for this analysis, we will focus only on the results concerning the more sophisticated model, GPT-3.5 Turbo results per processing level and math dataset are shown in Table 6.3 (for the interested reader results for the rest of the models are shown in Appendix A). GSM8K, the most challenging dataset from a rationale point of view, achieves a better accuracy in "Analyzing" and "Evaluating", denoting that harder problems require higher stages of the taxonomy in order to be accurately solved. The reverse does not seem to hold true; algebra has its best performance in the last two stages. The SVAMP dataset (the best and second best performing in accuracy, for BLM and BLES, respectively) performs better in understanding and analyzing but records its lowest performance in applying. The most challenging dataset for calculations, GSM-hard, achieves its highest accuracy in "Analyzing" followed by "Remembering", which could indicate a high level of familiarity of this model with the problems.

Self-evaluation

The results regarding the accuracy of the module of self-evaluation can be found in Table 6.4. Self-evaluation has been found to work better for larger or more sophisticated models. These models, with their vast knowledge bases and advanced reasoning capabilities, are better equipped to assess the quality of their own responses and make necessary adjustments. They can identify gaps in their understanding, recognize when a solution is not viable, and iterate upon their initial attempts to arrive at a more accurate and comprehensive answer.

Model	Self-evaluation Accuracy
GPT 3.5 turbo	68.1
Llama2 70B	56.5
Llama2 13B	45.8
Mixtral 8x7B	62.8

Πίνακας 6.4: *Self-evaluation accuracy across models. Larger/more sophisticated LLMs achieve a more accurate self-evaluation.*

Program of Bloom

In order to reduce the errors in calculations, we utilized Python code in our framework. More specifically, we used the same prompts but requested the answer in the form of python code which was safely executed. Handling Python code, especially for self-evaluation, is challenging for smaller LLMs, so we limited our experiments to GPT-3.5 Turbo. For the GSM-hard dataset—the most challenging in terms of calculations—accuracy improved by 13%. For the rest of the datasets, accuracy was lower compared to that achieved with BloomWise. Results can be found in Table 6.5. A possible explanation for the reduced performance of this variant is the fact that the prompts are not suitable for expressing the answer in code format. The textual rationale is a more suitable way of structuring the response because the strict structure required for using code does not always allow for the specific prompts to be followed.

Dataset	BloomWise	Program of Bloom
GSM8K	78.2	73.2
SVAMP	84.7	81.0
Algebra	85.1	67.1
GSM-hard	40.6	53.6

Πίνακας 6.5: *Comparison between Program of Bloom and BloomWise-Results for GPT3.5 turbo*

Κεφάλαιο 7

Conclusions, limitations and future work

7.1 Conclusions

Although LLMs excel in many areas, they struggle with mathematical problem-solving, which requires high precision and complex reasoning.

Researchers are increasingly working to improve this capability, often through in-context learning approaches. Some of the most widely used methods include Chain of Thought (CoT)-which encourages the LLMs to develop a textual rationale, Program of Thought (PoT)-which uses Python functions, and XoT-according to which the most suitable among the techniques of CoT (Chain of Thought), PoT (Program of Thought), and EoT (Equations of Thought) is selected, applied, and verified (iteratively, until a correct solution is found). The latter, which effectively integrates multiple methods, has proven especially successful. Building on this foundation of methodological integration, our approach aims to further enhance self-awareness in problem-solving.

We propose BloomWise, a problem-solving framework that uses prompts inspired by the first five levels of Bloom's Taxonomy to reach the correct solution. We introduce three variants of our method: EarlyStop, which halts the process if a solution is self-evaluated as correct, preventing progression to higher levels of the taxonomy; Program of Bloom, similar to EarlyStop but requiring the answer in the form of Python code; and Majority Voting, where the final solution is determined by a consensus across the outputs. Experiments conducted on four math reasoning datasets demonstrated the efficiency of our method, showcasing not only accuracy but also providing valuable insights into the cognitive abilities of large language models (LLMs). Among the variants, Majority Voting achieved the highest accuracy, while EarlyStop prioritized computational efficiency. Notably, the accuracy gap between EarlyStop and Majority Voting narrows for larger or more sophisticated models. The Program of Bloom variant achieved the best accuracy only on the GSM-hard dataset, while it performed the lowest on the rest of the datasets. Both the oracle setting and Majority Voting demonstrate the potential of our work.

When compared to the baselines, BloomWise demonstrated performance on par with XoT and outperformed simpler prompting methods such as CoT and PoT. The Oracle setting further illustrated the potential of BloomWise by significantly improving performance, emphasizing the value of methodological guidance and iterative refinement.

Our findings also provide valuable insights into the cognitive abilities of LLMs. Across all models, problem-solving was most effective when analyzing problems, followed by understanding, with larger and more sophisticated models showing a more balanced performance across all cognitive skills. Notably, while solving difficult problems requires higher cognitive abilities, the inverse is not necessarily true.

In conclusion, our proposed BloomWise framework demonstrated significant improvements in problem-solving by effectively integrating prompts inspired by Bloom's Taxonomy. These findings offer valuable insights into the cognitive abilities of LLMs, highlighting their strengths in areas such as analysis and understanding, while also revealing how different types of problems demand varying levels of cognitive skills. Furthermore, the results demonstrate the models' capacity to perform more consistently as their sophistication increases, providing a deeper understanding of the relationship between problem complexity and the cognitive capabilities required for effective solutions.

7.2 Limitations

We acknowledge that, while the BloomWise Oracle results show great promise, BloomWise EarlyStop has not yet achieved its full potential. One area requiring further investigation is self-evaluation or other criteria for early stopping. Furthermore, we have only experimented with very basic prompts, prompt tuning may improve performance surpassing the state of the art. Last but not least, adding the LLM replies from the previous levels of Bloom's taxonomy together with the prompt for the current level also has potential for improving the performance for all BloomWise variants.

7.3 Future work

Future research could focus on further enhancing the EarlyStop variant of BloomWise, aiming to approach the potential demonstrated under the oracle setting. By integrating improved self-assessment techniques and dynamic selection of the optimal Bloom's Taxonomy level, the model could achieve a higher accuracy while maintaining an effective balance between efficiency and accuracy.

Beyond refining EarlyStop, several other promising directions warrant exploration. Incorporating a memory mechanism would enable the model to retain and reuse insights from earlier cognitive steps, potentially improving its reasoning capabilities. Likewise, additional prompt-tuning could bolster the method's adaptability across diverse mathematical problems. Finally, extending this approach to a wider range of tasks beyond mathematics could not only enhance performance but also yield deeper insights into the reasoning processes of large language models.

Appendix

Appendix A

Accuracy per Bloom's level for Llama2 and Mixtral

Results for Llama2 and Mixtral can be found in Table A.1. In most cases, the best accuracy is recorded in the 'Analyzing' and 'Understanding' levels. It is worth noting the strong performance of Mixtral in 'Remembering', which, for some datasets, is the best among the five stages. This could indicate a higher level of familiarity with the problems or tasks compared to the other models.

	Dataset	Remembering	Understanding	Applying	Analyzing	Evaluating
LLaMA2-70B	GSM8K	38.7	49.3	35.7	49.8	26.7
	SVAMP	60.1	64.7	52.8	64.7	41.6
	Algebra	34.2	40.5	28.8	41.4	25.6
	GSM-hard	14.5	16.8	13.3	18.6	9.6
LLaMA2-13B	GSM8K	17.3	29.8	24.9	35.3	24.0
	SVAMP	34.5	44.3	40.1	42.7	35.0
	Algebra	14.8	23.8	23.8	27.9	21.6
	GSM-hard	5.5	7.7	7.2	9.6	6.2
Mixtral 8x7B Instruct	GSM8K	62.5	63.9	62.0	65.1	61.8
	SVAMP	77.1	75.4	73.6	76.6	64.0
	Algebra	64.8	62.2	59.5	63.0	58.1
	GSM-hard	30.2	32.0	32.2	32.9	25.4

Table A.1: *Accuracy per Bloom's Stage for each LLM and math task.*

Appendix B

Prompts and Example Outputs for BloomWise

In this section, we provide input and output examples for each level of Bloom's taxonomy in Tables B.1-B.5 and for self-evaluation in Table B.6. The instruction prefix in each prompt, e.g., "Summarize the problem and relevant concepts. Try to solve using this information" in Table B.2 for the "Understanding" level of the taxonomy, is fixed for all inputs for a taxonomy level. Note the simplicity and brevity of the instruction prompt prefix.

Input:	Carefully read the problem. Identify the type (e.g., algebra, geometry) and key concepts. Recall similar problems. If you recall the solution, state it. Problem: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?
Output:	Type: Arithmetic Key concepts: Multiplication, addition Similar problems: Calculating total distance traveled in a given time period Solution: James runs 3 sprints 3 times a week, so he runs a total of 3 x 3 = 9 sprints in a week. Each sprint is 60 meters, so the total distance James runs in a week is 9 x 60 = 540 meters. The final answer is: 540 meters.

Table B.1: *Example of a response corresponding to "Remembering" prompt.*

Input:	Summarize the problem and relevant concepts. Try to solve using this information. Problem: 43 children were riding on the bus. At the bus stop some children got off the bus. Then there were 21 children left on the bus. How many children got off the bus at the bus stop?
Output:	The problem involves 43 children initially on the bus, with some getting off at the bus stop, leaving 21 children on the bus. We need to determine how many children got off the bus at the bus stop. To solve this problem, we can subtract the number of children left on the bus from the initial number of children to find out how many got off.43 - 21 = 22 Therefore, 22 children got off the bus at the bus stop. The final answer is: 22

Table B.2: *Example of a response corresponding to "Understanding" prompt.*

Input:	Design and apply a step-by-step methodology. Alice runs 1.9 times faster than Bob, who runs at 4.7 mph. How fast does Alice run, in mph?
Output:	Step 1: Define the variables. Let A be the speed at which Alice runs in mph. Let B be the speed at which Bob runs in mph. Step 2: Translate the given information into equations. From the problem statement, we know that Alice runs 1.9 times faster than Bob, so we can write the equation: A = 1.9B. Given that Bob runs at 4.7 mph, we can substitute B = 4.7 into the equation above: A = 1.9(4.7). Step 3: Solve for A. A = 1.9(4.7). A = 8.93 mph. Step 4: Final answer. The final answer is: Alice runs at a speed of 8.93 mph.

Table B.3: *Example of a response corresponding to "Applying" prompt.*

Input:	Analyze your thoughts on this problem. Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
Output:	To find out how much profit Josh made, we first need to calculate the new value of the house after the repairs. The increase in value of 150% means the new value of the house is 250% of the original value (100% original value + 150% increase). Original value of the house = $80,000 Increase in value = 150% of $80,000 = 1.5 × $80,000 = $120,000 New value of the house = $80,000 + $120,000 = $200,000 Now, to find out how much profit Josh made, we subtract the total cost (purchase price + repairs) from the new value of the house. Total cost = $80,000 + $50,000 = $130,000 Profit = New value of the house - Total cost = $200,000 - $130,000 = $70,000 Therefore, Josh made a profit of $70,000. The final answer is: $70,000.

Table B.4: *Example of a response corresponding to "Analyzing" prompt.*

Input:	Evaluate your solution critically while expressing it. Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?
Output:	To evaluate the solution critically, we need to consider all the costs and revenues involved in planting the lemon tree. Costs: Initial cost of planting the tree: $90 Annual cost of watering and feeding the tree: $3 Revenues: Number of lemons grown per year: 7 Price per lemon: $1.5 To calculate the annual profit, we need to subtract the total cost from the total revenue: Total revenue = 7 lemons/year * $1.5/lemon = $10.5/year Total cost = $3/year Annual profit = Total revenue - Total cost = $10.5 - $3 = $7.5/year To determine how many years it will take before Carlos starts earning money on the lemon tree, we need to consider the initial cost of planting the tree. Since the annual profit is $7.5, it will take $90 / $7.5 = 12 years for Carlos to recoup the initial cost of planting the tree. Therefore, the final answer is: It will take 12 years for Carlos to start earning money on the lemon tree.

Table B.5: *Example of a response corresponding to "Evaluating" prompt.*

Input:	Carefully read the problem and the solution. Try to reproduce the solution in your own words while carefully checking for errors, as the solution is probably wrong (as a rationale or calculation). Provide the numerical result at the end of your answer. Problem: Jill gets paid $20 per hour to teach and $30 to be a cheerleading coach. If she works 50 weeks a year, 35 hours a week as a teacher and 15 hours a week as a coach, what's her annual salary? Solution: Type: Algebra. Key concepts: Rates, hours worked, annual salary. Similar problems: Calculating annual salary based on hourly rates and hours worked. Teacher's salary per week = $20/hour * 35 hours/week = $700/week Coach's salary per week = $30/hour * 15 hours/week = $450/week Total weekly salary = $700 + $450 = $1150/week Annual salary = $1150/week * 50 weeks/year = $57,500/year The final answer is: $57,500.
Output:	Revised Solution: To find Jill's annual salary, we need to calculate her total earnings from teaching and coaching for the entire year. Teacher's salary per week = $20/hour * 35 hours/week = $700/week Coach's salary per week = $30/hour * 15 hours/week = $450/week Total weekly salary = $700 + $450 = $1150/week To find the annual salary, we multiply the weekly salary by the number of weeks in a year: Annual salary = $1150/week * 50 weeks/year = $57,500/year Therefore, Jill's annual salary is $57,500.

Table B.6: *Example of a self-evaluation prompt and reply.*

Appendix C

Exploratory Approaches and Unsuccessful Attempts

Research often involves venturing into uncharted territory, testing various methods, and exploring diverse approaches before arriving at a conclusive result. While not all attempts lead to success, they are valuable steps in the problem-solving process, providing insights into the complexities of the subject and guiding the refinement of subsequent strategies. This appendix documents the methods and approaches we explored during this thesis that ultimately did not yield the desired results.

C.1 Prompt tuning

During our experimentation, we explored a wide range of prompts, ranging from detailed and highly specific instructions to broader, more general queries. Surprisingly, our findings revealed that simpler prompts outperformed their more complex counterparts. To illustrate this, we now provide an example of a more detailed prompt used during our trials, along with the accuracy it achieved. Due to the resource-intensive nature of these experiments, we focused on one dataset to systematically refine our approach and conclude on the prompts we would ultimately use. The prompts used in these experiments can be found in tables C.1 and C.2, while the corresponding accuracies are shown in table C.3.

C.2 Alternative approach on self-evaluation

The potential of our approach became evident through the strong performance demonstrated by both the oracle and majority voting methods, which highlighted its accuracy and robustness. To further enhance accuracy for BLES, we explored an alternative approach to the self-evaluation module, aiming to address its limitations and improve its effectiveness. Although this alternative approach did not yield better results than our initial self-evaluation module (explained in the framework section of this thesis), we include it here for the sake of completeness: At each stage, the LLM is provided with the problem, the current solution, a description of the ongoing stage, and an outline of the subsequent stage. The model evaluates the accuracy and completeness of the solution at the current stage, considering whether it adequately addresses the problem with minimal effort. Based on this evaluation, the LLM decides whether to proceed to the next stage or conclude that the solution is correct and complete (the prompts can be found in table C.4). We initially decided to experiment with smaller language models (LLMs), such as Llama2 13b, due to their relatively limited ability to follow instructions

Bloom's Stage	**Description**
Remembering	Read the problem carefully and identify its type (e.g., algebra, geometry). List the key concepts and terms involved. Recall any similar problems you have solved before and describe how they were solved. If you remember a specific solution or formula, state it clearly and explain why it is applicable here. Provide the solution based on this recall.
Understanding	Summarize the problem in your own words, highlighting the main concepts and key information. Explain how these concepts connect and why they are important for solving the problem. Based on this understanding, solve the problem.
Applying	Design a detailed step-by-step methodology to solve the problem. Apply this methodology systematically, detailing each step clearly and logically. Ensure that every step is explained thoroughly and contributes towards finding the solution.
Analyzing	Analyze the problem deeply and break it down into its fundamental parts. Examine the relationships and patterns among these parts. Identify any assumptions or potential errors. Use this analysis to construct a solution, detailing each part of your thought process and how it contributes to the overall solution.
Evaluating	Critically evaluate the problem and potential solutions. Consider multiple approaches and compare their effectiveness. Assess the accuracy and efficiency of each method. Choose the best solution based on this evaluation, providing a detailed explanation of why this approach is superior and how it effectively solves the problem.

Table C.1: *Prompts' version 1*

Bloom's Stage	**Description**
Remembering	Carefully read the problem. Identify the type of problem (e.g., algebra, geometry) and key concepts involved. Recall any similar problems you have encountered before. If you recall the solution or a similar approach, state it clearly.
Understanding	Summarize the problem, highlighting the main concepts and relevant information. Explain how these concepts relate to each other within the context of the problem. Based on this understanding, attempt to solve the problem and provide a clear explanation of your approach.
Applying	Design a step-by-step methodology to solve the problem. Apply this methodology systematically, detailing each step clearly. Ensure that each step logically follows from the previous one and contributes towards solving the problem.
Analyzing	Analyze the solution and thought process. Break down the problem into its component parts and examine the relationships and patterns among these parts. Provide a detailed analysis of your approach and how each part contributes to the overall solution.
Evaluating	Critically evaluate the solution while expressing it. Assess its accuracy and efficiency. Consider alternative methods and compare them to your approach. Discuss the strengths and weaknesses of your solution, and suggest any improvements or refinements that could be made.

Table C.2: *Prompts' version 2*

Prompts' version 1	Prompts' Version 2	Final Prompts' Version
81.1	81.3	82.6

Table C.3: *Comparison of BLM's Accuracy Across Different Prompt Versions on the GSM8K Dataset (1,319 Samples) using GPT-3.5-turbo*

compared to larger and more sophisticated models. The rationale behind this decision was that if smaller LLMs are capable of adhering to the designed prompts and achieving satisfactory accuracy, it is highly likely that larger and more advanced models would also be capable of doing so, often with even better results. By starting with smaller models, we aimed to reach a decision on the viability of the approach without expending significant computational resources, before conducting full-scale experiments with the chosen approach. The accuracy achieved during these experiments can be found in Table C.5. Although the difference in accuracy between Approaches 1 and 2 is not significant, Llama2 13b struggles to follow the instructions and format specified in the prompt of Approach 1, even after prompt tuning was performed. As a result, Approach 2 appears to be more suitable for smaller LLMs, as it aligns better with their capabilities and limitations. Based on these observations, we chose Approach 2 for further experiments and evaluations, as it demonstrates greater suitability and practicality for models like Llama2 13b while maintaining reasonable accuracy.

Prompt:
You are an experienced mathematician. Please review the following math problem and evaluate the solution.
Problem: problem
I current_stage_desc and reached the following solution, which is probably incorrect: model_output_text
Consider if the solution addresses the problem correctly and completely, with the objective of achieving a correct solution with the minimum number of efforts.
Evaluate the solution:
If the solution is incorrect or incomplete, and further attempts are necessary, grade it as '0'. This grade indicates that I should next_stage_desc.
If the solution is correct and meets all requirements of the problem, grade it as '1'. This grade indicates that no further efforts are needed and the solution is correct.
Provide your grade at the end of your response, ensuring to stick to the format: Grade: X.

Table C.4: *Prompts for the alternative approach of self-evaluation*

Approach	Accuracy
Self-Evaluation Approach 1	20.10
Self-Evaluation Approach 2 (chosen)	23.87

Table C.5: *Accuracy Comparison of Self-Evaluation Approaches on Llama2 13b for the Algebra Dataset (222 Samples)*

Bibliography

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent και Christian Janvin. *A Neural Probabilistic Language Model*. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] Alec Radford και Karthik Narasimhan. *Improving Language Understanding by Generative Pre-Training*. *OpenAI Blog*, 2018.

[3] Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu και Zheng Zhang. *Plan, Verify and Switch: Integrated Reasoning with Diverse X-of-Thoughts*. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*Houda Bouamor, Juan Pino και Kalika Bali, επιμελητές, σελίδες 2807–2822, Singapore, 2023. Association for Computational Linguistics.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever και Dario Amodei. *Language Models are Few-Shot Learners*. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. NeurIPS 2020, December 6-12, 2020, virtual conference.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le και Denny Zhou. *Chain-of-thought Prompting Elicits Reasoning in Large Language Models*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Presented at NeurIPS 2022.

[6] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan και Graham Neubig. *PAL: Program-Aided Language Models*. *CoRR*, abs/2211.10435, 2022. Preprint available at arXiv.

[7] Wenhu Chen, Xueguang Ma, Xinyi Wang και William W. Cohen. *Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks*. *CoRR*, abs/2211.12588, 2022. Preprint available at arXiv.

[8] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari και Vedant Misra. *Solving quantitative reasoning problems with language models*. *NeurIPS*, 2022.

[9] Daniel Jurafsky και James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

[10] Mary Forehand και others. *Bloom's taxonomy: Original and revised. Emerging perspectives on learning, teaching, and technology*, 8:41–44, 2005.

[11] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo και Yusuke Iwasawa. *Large Language Models are Zero-Shot Reasoners. Advances in Neural Information Processing Systems (NeurIPS)*, 2022. NeurIPS 2022.

[12] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao και Karthik Narasimhan. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models. CoRR*, abs/2305.10601, 2023. Preprint available at arXiv.

[13] John H. Flavell. *Metacognition and Cognitive Monitoring: A New Area of Cognitive-Developmental Inquiry. American Psychologist*, 34:906–911, 1979.

[14] Yuqing Wang και Yun Zhao. *Metacognitive Prompting Improves Understanding in Large Language Models. CoRR*, abs/2308.05342, 2024. Presented at NAACL 2024. Subjects: Computation and Language (cs.CL); Artificial Intelligence (cs.AI).

[15] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse και John Schulman. *Training verifiers to solve math word problems. CoRR*, abs/2110.14168, 2021.

[16] Arkil Patel, Satwik Bhattamishra και Navin Goyal. *Are NLP Models Really Able to Solve Simple Math Word Problems? Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, σελίδες 2080–2094, Online, 2021. Association for Computational Linguistics.

[17] Joy He-Yueya, Gabriel Poesia, Rose E. Wang και Noah D. Goodman. *Solving Math Word Problems by Combining Language Models with Symbolic Solvers. CoRR*, abs/2304.09102, 2023.

[18] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin

Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov και Thomas Scialom. *Llama 2: Open foundation and fine-tuned chat models*. *CoRR*, abs/2307.09288, 2023.

[19] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diegode las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix και William El Sayed. *Mixtral 8x7B: A Sparse Mixture of Experts Language Model*. *arXiv*, 2401.04088, 2024. Comments: See more details at the provided URL.

[20] Ian J. Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[21] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag, Berlin, Heidelberg, 2006.

[22] Sergios Theodoridis και Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, Inc., USA, 4thη έκδοση, 2008.

[23] Olivier Chapelle, Bernhard Schölkopf και Alexander Zien. *Semi-Supervised Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

[24] David E. Rumelhart, Geoffrey E. Hinton και Ronald J. Williams. *Learning Representations by Back-propagating Errors*. *Nature*, 323(6088):533–536, 1986.

[25] Yann LeCun, Patrick Haffner και Y. Bengio. *Object Recognition with Gradient-Based Learning*. *Journal Placeholder*, 2000.

[26] Christopher Olah. *Understanding LSTM Networks*, 2015. Accessed: 2024-10-02.

[27] David E. Rumelhart, Geoffrey E. Hinton και Ronald J. Williams. *Learning Representations by Back-Propagating Errors*. *MIT Press*, σελίδες 696–699, Cambridge, MA, USA, 1988.

[28] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[29] Sepp Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. Διδακτορική Διατριβή, Technische Universität München, 1991. Available at: `https://people.idsia.ch/~juergen/SeppHochreiter1991ThesisAdvisorSchmidhuber.pdf`.

[30] Sepp Hochreiter και Jürgen Schmidhuber. *Long Short-Term Memory*. *Neural Computation*, 9(8):1735–1780, 1997.

[31] Dzmitry Bahdanau, Kyunghyun Cho και Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473*, 2014.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser και Illia Polosukhin. *Attention Is All You Need. arXiv preprint arXiv:1706.03762*, 2017.

[33] H. P. Luhn. *A Statistical Approach to Mechanized Encoding and Searching of Literary Information. IBM Journal of Research and Development*, 1(4):309–317, 1957.

[34] Usman Naseem, Imran Razzak, Shah Khalid Khan και Mukesh Prasad. *A Comprehensive Survey on Word Representation Models: From Classical to State-Of-The-Art Word Representation Language Models. CoRR*, abs/2010.15036, 2020.

[35] John Hancock και Taghi Khoshgoftaar. *Survey on categorical data for neural networks. Journal of Big Data*, 7, 2020.

[36] Zhiyuan Liu, Yankai Lin και Maosong Sun. *Representation Learning for Natural Language Processing. CoRR*, abs/2102.03732, 2021.

[37] Lorenzo Ferrone και Fabio Massimo Zanzotto. *Symbolic, Distributed and Distributional Representations for Natural Language Processing in the Era of Deep Learning: a Survey. CoRR*, abs/1702.00764, 2017.

[38] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR*. 2013, 2013.

[39] Edgar Altszyler, Mariano Sigman και Diego Fernández Slezak. *Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. CoRR*, abs/1610.01520, 2016.

[40] Zhiyuan Liu, Yankai Lin και Maosong Sun. *Representation Learning for Natural Language Processing*. 2020.

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

[42] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee και Luke Zettlemoyer. *Deep contextualized word representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2018.

[43] Wei Xu και Alex Rudnicky. *Can artificial neural networks learn language models? INTERSPEECH*, 2000.

[44] Kun Jing και Jungang Xu. *A Survey on Neural Network Language Models*. *CoRR*, abs/1906.03591, 2019.

[45] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký και Sanjeev Khudanpur. *Recurrent Neural Network Based Language Model*. *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, τόμος 2, σελίδες 1045–1048, 2010.

[46] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan H. Cernocky και Sanjeev Khudanpur. *Extensions of Recurrent Neural Network Language Model*. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, σελίδες 5528–5531, 2011.

[47] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei και Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. *OpenAI Blog*, 2019.

[48] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever και Dario Amodei. *Language Models are Few-Shot Learners*. *Advances in Neural Information Processing Systems*H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan και H. Lin, επιμελητές, τόμος 33, σελίδες 1877–1901. Curran Associates, Inc., 2020.

[49] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever και Dario Amodei. *Language Models are Few-Shot Learners*. *Advances in Neural Information Processing Systems*H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan και H. Lin, επιμελητές, τόμος 33, σελίδες 1877–1901. Curran Associates, Inc., 2020.

[50] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar και others. *LLaMA: Open and Efficient Foundation Language Models*. *arXiv preprint arXiv:2302.13971*, 2023.

[51] Biao Zhang και Rico Sennrich. *Root mean square layer normalization*. *Advances in Neural Information Processing Systems*, τόμος 32, 2019.

[52] Noam Shazeer. *Gated Linear Units (GLUs)*. *arXiv preprint arXiv:2002.05202*, 2020. Available at https://arxiv.org/abs/2002.05202.

[53] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen και Yunfeng Liu. *Roformer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864*, 2021.

[54] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diegode las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix και William El Sayed. *Mistral 7B: A High-Performance and Efficient Language Model. arXiv preprint arXiv:XXXX.XXXXX*, 2023.

[55] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch και others. *Mixtral of Experts: A Sparse Mixture of Experts Language Model. arXiv preprint arXiv:XXXX.XXXXX*, 2023.

[56] Sylvestre Alvise Rebuffi, Hakan Bilen και Andrea Vedaldi. *Learning multiple visual domains with residual adapters. CoRR*, abs/1705.08045, 2017.

[57] Xiang Lisa Li και Percy Liang. *Prefix-Tuning: Optimizing Continuous Prompts for Generation. arXiv preprint arXiv:2101.00190*, 2021.

[58] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan και Sylvain Gelly. *Parameter-Efficient Transfer Learning for NLP. Proceedings of the 36th International Conference on Machine Learning*Kamalika Chaudhuri και Ruslan Salakhutdinov, επιμελητές, τόμος 97 στο *Proceedings of Machine Learning Research*, σελίδες 2790–2799. PMLR, 2019.

[59] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho και Iryna Gurevych. *AdapterFusion: Non-Destructive Task Composition for Transfer Learning. CoRR*, abs/2005.00247, 2020.

Abbreviations - Acronyms

βλ.	βλέπε
κ.λπ.	και λοιπά
δηλ.	δηλαδή
ΜΓΜ	Μεγάλα Γλωσσικά Μοντέλα
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BLES	BloomWise Early Stopping
BLM	BloomWise Majority Voting
BLO	BloomWise Oracle
CoT	Chain of Thought
CNN	Convolutional Neural Network
DL	Deep Learning
EoT	Equations of Thought
FFN	Feedforward Neural Network
LLM	Large Language Models
LSTM	Long short-term memory
ML	Machine Learning
NLP	Natural Language Processing
PoT	Program of Thought
RNN	Recurrent Neural Network
XoT	X of Thought
ToT	Tree of Thought