



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  
Πολυτεχνική Σχολή  
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Τομέας Τηλεπικοινωνιών

Τίτλος διπλωματικής

Διπλωματική Εργασία  
του  
Θεοφάνη Θαρρόπουλου

**Επιβλέπων:** Ανδρέας Συμεωνίδης  
Καθηγητής Α.Π.Θ.

3 Ιουνίου 2024

## **Περίληψη**

Αντικείμενο της παρούσας διπλωματικής εργασίας αποτελεί η έρευνα για την αξιολόγηση της ποιότητας του κώδικα που παράγεται από Μεγάλα Γλωσσικά Μοντέλα (LLMs), και πιο συγκεκριμένα από το GitHub Copilot[1]. Η μελέτη εστιάζει στην αξιολόγηση της ποιότητας του κώδικα που παράγεται από το όπilot και στην βελτιστοποίηση των προτροπών (prompts) για την επίτευξη των επιθυμητών αποτελεσμάτων μέσω τεχνικών μηχανικής προτροπής (prompt engineering) και της μηχανικής μάθησης. Τα αποτελέσματα αποδεικνύουν τις δυνατότητες και τους περιορισμούς του όπilot στην παραγωγή ποιοτικού κώδικα και προσφέρουν νέες προσεγγίσεις για την βελτίωση της αλληλεπίδρασης μεταξύ του χρήστη και του εργαλείου μέσω στοχευμένων τεχνικών προτροπής.

## **Abstract**

Empty

# Ευχαριστίες

Άδειο

# Τίτλος διπλωματικής

Θεοφάνης Θαρρόπουλος  
theofact@ece.auth.gr

3 Ιουνίου 2024

# Περιεχόμενα

<b>1</b>	<b>Εισαγωγή</b>	<b>2</b>
1.1	Δημιουργία Κώδικα (Code Generation) - Ολοκλήρωση Κώδικα (Code Completion) . . . . .	3
<b>A'</b>	<b>Ακρωνύμια και συντομογραφίες</b>	<b>8</b>

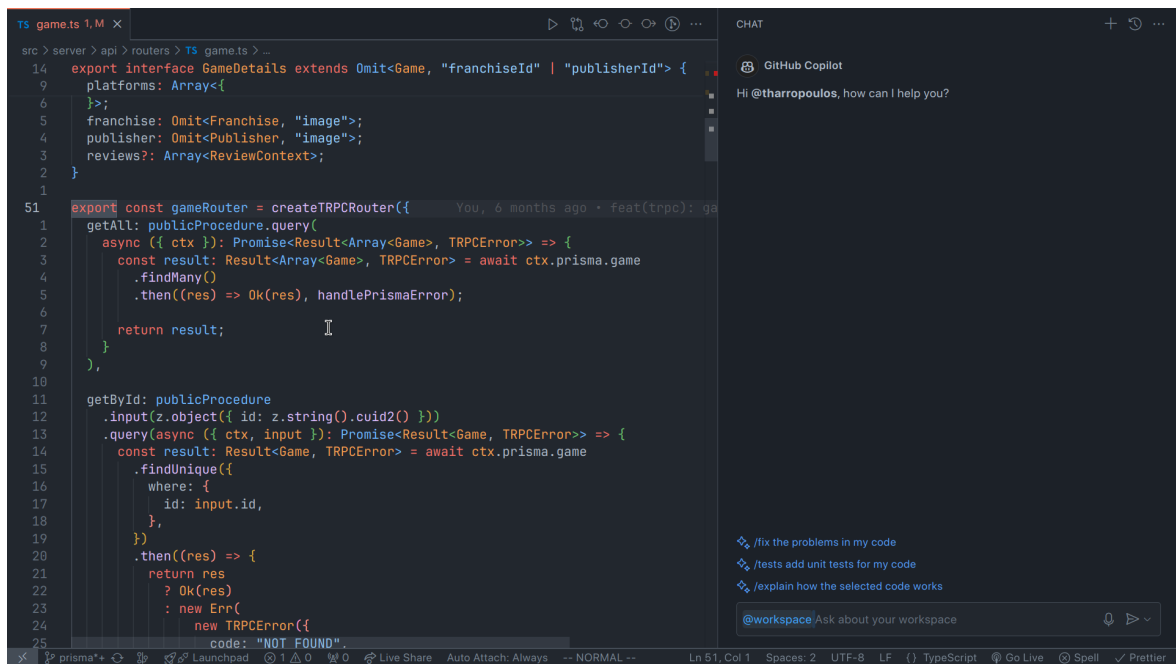
# Κεφάλαιο 1

## Εισαγωγή

Η ανάπτυξη των Μεγάλων Γλωσσικών Μοντέλων (LLM) έχει επιφέρει ριζικές αλλαγές στον τομέα της Τεχνητής Νοημοσύνης και της Επεξεργασίας Φυσικής Γλώσσας (NLP) [8, 47, 48, 22]. Η εισαγωγή αυτών των μοντέλων στην καθημερινή ζωή μέσω του Chat-GPT το 2022 [31] έχει πυροδοτήσει μια επανάσταση στην τεχνολογική αγορά, σηματοδοτώντας την απαρχή του αγώνα για την κυριαρχία στην αγορά της Τεχνητής Νοημοσύνης [37, 29, 35, 26, 3]. Τα μοντέλα αυτά έχουν αποδειχθεί εξαιρετικά αποτελεσματικά στην αντιμετώπιση προβλημάτων επεξεργασίας φυσικής γλώσσας, όπως η αναγνώριση φυσικής γλώσσας, προάγοντας την ανάπτυξη της Γενικής Τεχνητής Νοημοσύνης (AGI) [5, 18].

Μέσα σε αυτή την επανάσταση, η χρήση μοντέλων επεξεργασίας φυσικής γλώσσας στον τομέα του προγραμματισμού και της ανάπτυξης λογισμικού, ονόματι βοηθοί κώδικα (Code Assistants), έχει αναδειχθεί ως ένας από τους πιο υποσχόμενους τομείς της τεχνολογίας. Ένα από τα πιο διαδεδομένα εργαλεία με αυτόν το σκοπό είναι το GitHub Copilot [17, 1]. Αναπτυγμένο σε συνεργασία με την OpenAI, το GitHub Copilot ξεκίνησε χρησιμοποιώντας το μοντέλο ονόματι Codex της OpenAI [11], σχεδιασμένο εξ αρχής αποκλειστικά για τη παραγωγή κώδικα, για να προτείνει κώδικα στον προγραμματιστή κατά την γραφή κώδικα.

Η παροδική απόσυρση του μοντέλου Codex τον Μάρτιο του 2023 και η οριστική του απόσυρση το 2023 [23], οδήγησε στην ανάπτυξη ενός νέου μοντέλου, σε συνεργασία μεταξύ της OpenAI, της Microsoft Azure AI, και της GitHub AI. Το νέο μοντέλο αρχικά βασίστηκε στο GPT-3.5 Turbo [46], με την επόμενη του έκδοση να βασίζεται στο GPT-4 [46], με το κωδικό όνομα GitHub Copilot X [14], δίνοντας την δυνατότητα για μια νέα λειτουργία, του GitHub Copilot Chat, ενός chatbot μοντέλου, παρόμοιο με αυτό του Chat-GPT. Μέσω αυτής της λειτουργίας, ο προγραμματιστής μπορεί μέσα από το περιβάλλον ανάπτυξής του (IDE), να κάνει ερωτήσεις στο μοντέλο, χρησιμοποιώντας φυσική γλώσσα, ενώ το μοντέλο μπορεί να χρησιμοποιήσει τον ήδη υπάρχοντα κώδικα, την τεκμηρίωση, και τις οδηγίες του προγραμματιστή για να απαντήσει στην ερώτηση του προγραμματιστή.



Σχήμα 1.1: Παράδειγμα χρήσης του GitHub Copilot Chat εντός του IDE Visual Studio Code [40, 4]

Το GitHub Copilot Chat αρχικά ήταν διαθέσιμο μόνο μέσω λίστας αναμονής, με την πρόσβαση να δίνεται σε περιορισμένο αριθμό προγραμματιστών, κατόπιν αίτησης, με την ενσωμάτωση του GPT-4 να γίνεται επίσημα τον Νοέμβριο του 2023 [36]. Η δημόσια κυκλοφορία του GitHub Copilot Chat έγινε τον Δεκέμβριο του 2023 [45].

Πέρα από το GitHub Copilot, υπάρχουν πολλοί άλλοι code assistants, με αυτούς που λήφθηκαν υπ' όψιν να είναι οι:

- Tabnine [41, 42]
- Codeium [10]
- Amazon Codewhisperer [7]

Η απόφαση για την χρήση του GitHub Copilot ως το εργαλείο, του οποίου η λειτουργία θα εξεταστεί στην παρούσα διπλωματική, έγινε με βάση την ευρεία χρήση του, την ενσωμάτωση του μοντέλου GPT-4, και την δυνατότητα χρήσης του GitHub Copilot Chat, καθώς και την δωρεάν παροχή του σε ενεργούς φοιτητές του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης μέσω του προγράμματος GitHub Student Developer Pack [2].

## 1.1 Δημιουργία Κώδικα (Code Generation) - Ολοκλήρωση Κώδικα (Code Completion)

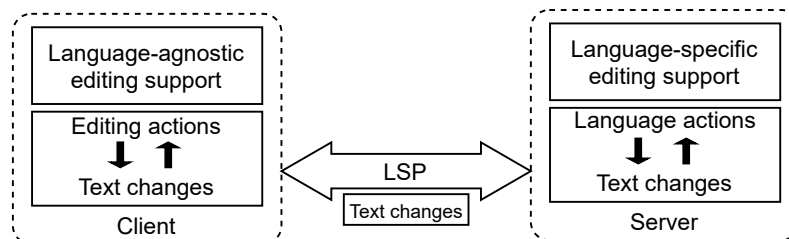
Δύο όροι που συχνά συναντώνται στην βιβλιογραφία είναι η **Δημιουργία Κώδικα** (Code Generation) και η **Ολοκλήρωση Κώδικα** (Code Completion). Και οι δύο όροι έχουν άμεση σχέση με την παραγωγικότητα του προγραμματιστή και αποτελούν σημαντικά



εργαλεία για την ανάπτυξη λογισμικού [28, 25, 6], και ενώ και τα δύο μπορούν να χρησιμοποιήσουν μοντέλα Τεχνητής Νοημοσύνης [39, 33] με την διαφορά να βρίσκεται στον τρόπο λειτουργίας τους.

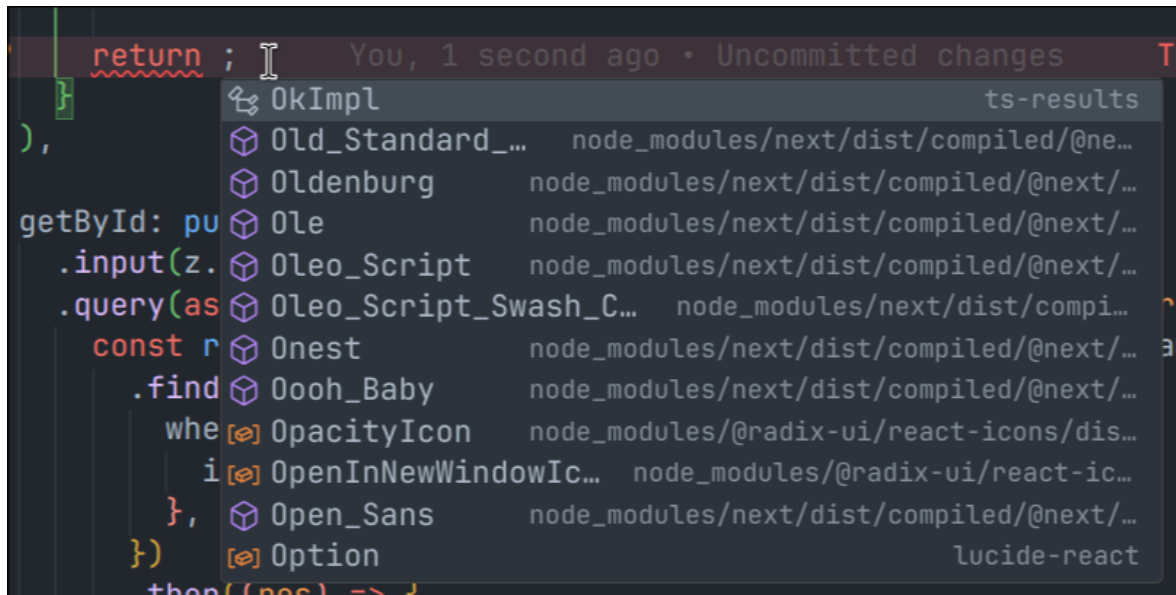
## Ολοκλήρωση Κώδικα

Σύμφωνα με Omar et al. [30], η Ολοκλήρωση Κώδικα αφορά εργαλεία που τα περισσότερα περιβάλλοντα ανάπτυξης παρέχουν μέσω της μορφής πλωτού μενού που περιέχει συμφραζόμενες-σχετικές μεταβλητές, πεδία, μεθόδους, τύπους και άλλα αποσπάσματα κώδικα. Επιλέγοντας από αυτό το μενού, οι προγραμματιστές μπορούν να αποφύγουν πολλά συνηθισμένα ορθογραφικά και λογικά λάθη και να εξαλείψουν τις περιττές πληκτρολογήσεις. Το βασικό εργαλείο που χρησιμοποιείται για την επίτευξη του σκοπού αυτού είναι το Language Server Protocol (LSP) της Microsoft, δίνοντας την δυνατότητα σε κάθε περιβάλλον ανάπτυξης να επικοινωνήσει με την εξωτερική διεργασία του Language Server και να λάβει πληροφορίες για τον κώδικα που γράφεται, όπως τις προτάσεις ολοκλήρωσης κώδικα, τα λάθη, και τις προτάσεις διόρθωσης, σε μια διάταξη χρήστη - διακομιστή [32, 9].



Σχήμα 1.2: Η αρχιτεκτονική του Language Server Protocol, Δανεισμένο από [34]

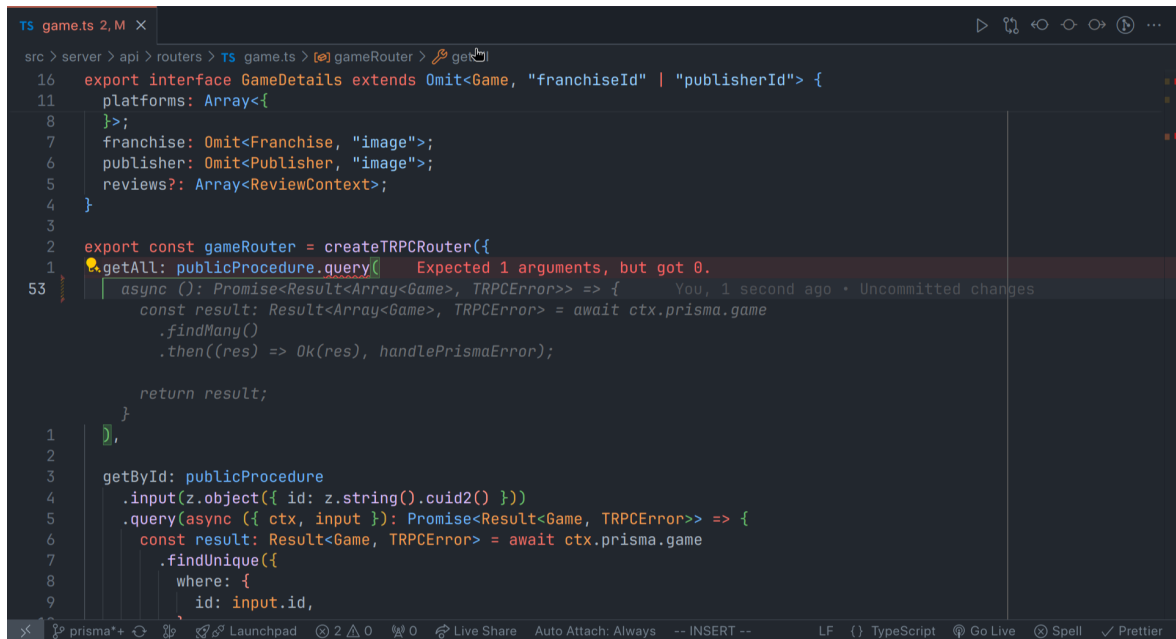
Η Ολοκλήρωση Κώδικα δεν παράγει κώδικα από το μηδέν, αλλά προτείνει συμπληρωματικές λύσεις στον υπάρχοντα κώδικα, βοηθώντας τον προγραμματιστή να ολοκληρώσει τον κώδικα του ταχύτερα.



Σχήμα 1.3: Παράδειγμα Ολοκλήρωσης Κώδικα εντός του IDE Visual Studio Code

## Δημιουργία Κώδικα

Η Δημιουργία Κώδικα αφορά την άμεση παραγωγή κώδικα από ένα μοντέλο στοχαστικά, με βάση το συγκεκριμένο κώδικα του προγραμματιστή. Το μοντέλο που χρησιμοποιείται για την παραγωγή κώδικα είναι εκπαιδευμένο σε μεγάλα σύνολα δεδομένων και μπορεί να παράγει κώδικα από το μηδέν. Το μοντέλο αυτό μπορεί να παράγει κώδικα σε πολλές γλώσσες προγραμματισμού, όπως Python, JavaScript, C++ κ.α. Η παραγωγή κώδικα μπορεί να γίνει μέσω μιας διεπαφής (API) που παρέχεται από το μοντέλο, ή μέσω μιας επέκτασης ενός ειδικού περιβάλλοντος ανάπτυξης, όπως στην περίπτωση του GitHub Copilot.



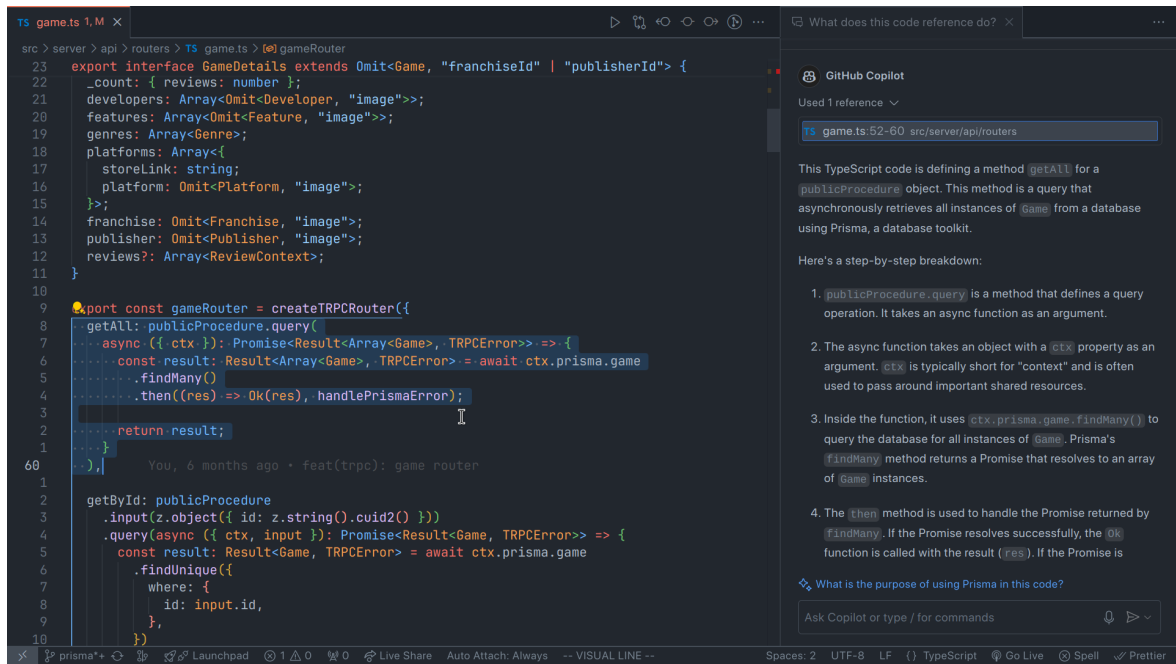
```
TS game.ts 2, M x
src > server > api > routers > TS game.ts > gameRouter > get
16 export interface GameDetails extends Omit<Game, "franchiseId" | "publisherId"> {
11   platforms: Array<{
8     }>;
7   franchise: Omit<Franchise, "image">;
6   publisher: Omit<Publisher, "image">;
5   reviews?: Array<ReviewContext>;
4 }
3
2 export const gameRouter = createTRPCRouter({
1   getAll: publicProcedure.query(() => {
53   |   async (): Promise<Result<Array<Game>, TRPCError>> => {
      const result: Result<Array<Game>, TRPCError> = await ctx.prisma.game
        .findMany()
        .then((res) => Ok(res), handlePrismaError);

      return result;
    }
  },
  getById: publicProcedure
    .input(z.object({ id: z.string().cuid2() }))
    .query(async ({ ctx, input }): Promise<Result<Game, TRPCError>> => {
      const result: Result<Game, TRPCError> = await ctx.prisma.game
        .findUnique({
          where: {
            id: input.id,
          }
        })
    })
});
```

Σχήμα 1.4: Παράδειγμα Δημιουργίας Κώδικα σε πραγματικό χρόνο (real time code generation) εντός του IDE Visual Studio Code από το GitHub Copilot

Η παραγωγή κώδικα γίνεται με διάφορους βαθμούς λεπτομέρειας. Αρχικά, το μοντέλο υπολογίζει το επόμενο σύμβολο του κώδικα χρησιμοποιώντας **next token prediction** [20, 24, 43, 15, 13, 12]. Στο στάδιο της ολοκλήρωσης ολόκληρης σειράς κώδικα, το μοντέλο χρησιμοποιεί ένα κομμάτι συγκεκριμένου κώδικα. [20, 19, 38, 27]. Στο μέγιστο δυνατό βαθμό, το μοντέλο μπορεί να παράγει ολόκληρες συναρτήσεις ή κλάσεις. [16, 19, 1].

Η παραγωγή κώδικα επίσης μπορεί να λάβει υπ' όψιν αποκλειστικά το συγκεκριμένο κώδικα πριν την θέση του κέρσορα στο αρχείο ή και τον κώδικα που ακολουθεί την θέση του κέρσορα. [21] Το μοντέλο του GitHub Copilot χρησιμοποιεί την δεύτερη προσέγγιση, παράγοντας κώδικα που συμπληρώνει τον υπάρχοντα κώδικα του προγραμματιστή. [1, 16, 44]. Κατά την έρευνα λήφθηκε η απόφαση να χρησιμοποιηθεί, ως επί το πλείστον, η λειτουργία GitHub Copilot Chat για την έρευνα και τα πειράματα, κυρίως γιατί ο συγκεκριμένος κώδικας που χρησιμοποιεί το μοντέλο κατά την λειτουργία αυτή, μπορεί να επιλεγεί είτε από το μοντέλο, είτε συγκεκριμένα από τον προγραμματιστή. Ένας ακόμα λόγος που επιλέχθηκε η χρήση του GitHub Copilot Chat, είναι ότι η καταγραφή του κώδικα που παράχθηκε κατά την έρευνα από το μοντέλο ήταν πολύ δυσκολότερη με την χρήση του GitHub Copilot, καθώς η δημιουργία κώδικα σε πραγματικό χρόνο (**real time code generation**) ήταν πολύ δύσκολη αυτής του GitHub Copilot Chat.



Σχήμα 1.5: Παράδειγμα Δημιουργίας Κώδικα σε πραγματικό χρόνο (real time code generation) εντός του IDE Visual Studio Code από το GitHub Copilot

# Παράρτημα Α΄

## Ακρωνύμια και συντομογραφίες

**LLM** Large Language Model

**NLP** Natural Language Processing

**IDE** Integrated Development Environment

**LSP** Language Server Protocol

**API** Application Programming Interface

# Bibliography

- [1] “Github copilot.” [Online]. Available: <https://github.com/features/copilot>
- [2] “Github student developer pack.” [Online]. Available: <https://education.github.com/pack>
- [3] “Trendforce says with cloud companies initiating ai arms race, gpu demand from chatgpt could reach 30,000 chips as it readies for commercialization,” *TrendForce*, Nov 2023, archived from the original on May 30, 2024. Retrieved November 2, 2023.
- [4] “Visual studio code.” [Online]. Available: <https://code.visualstudio.com/>
- [5] S. Adams, I. Arel, J. Bach, R. Coop, R. Furlan, B. Goertzel, J. S. Hall, A. Samsonovich, M. Scheutz, M. Schlesinger *et al.*, “Mapping the landscape of human-level artificial general intelligence,” *AI Magazine*, vol. 33, no. 1, pp. 25–42, 2012.
- [6] M. Asaduzzaman, C. K. Roy, K. A. Schneider, and D. Hou, “Csc: Simple, efficient, context sensitive code completion,” in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 71–80.
- [7] J. Bays, “Aws announces amazon codewhisperer (preview),” Jun 2022. [Online]. Available: <https://aws.amazon.com/about-aws/whats-new/2022/06/aws-announces-amazon-codewhisperer-preview/>
- [8] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, “On the opportunities and risks of foundation models,” 2021.
- [9] H. Bunder, “Decoupling language and editor - the impact of the language server protocol on textual domain-specific languages,” in *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development - MODELSWARD*, INSTICC. SciTePress, 2019, pp. 131–142.
- [10] S. Carter, “Exponential baby! navigating the ai convergence of tech with nvidia,” *Forbes Digital Assets*, May 2024, retrieved May 30, 2024. [Online]. Available: <https://www.forbes.com/sites/digital-assets/2024/05/11/exponential-baby-navigating-the-ai-convergence-of-tech-with-nvidia/>
- [11] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H.

- Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, “Evaluating large language models trained on code,” 2021.
- [12] M. Ciniselli, N. Cooper, L. Pascarella, A. Mastropaolo, E. Aghajani, D. Poshyanyk, M. Di Penta, and G. Bavota, “An empirical study on the usage of transformer models for code completion,” *IEEE Transactions on Software Engineering*, 2021.
- [13] M. Ciniselli, N. Cooper, L. Pascarella, D. Poshyanyk, M. Di Penta, and G. Bavota, “An empirical study on the usage of bert models for code completion,” in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021, pp. 108–119.
- [14] T. Dohmke, “Github copilot x: The ai-powered developer experience,” *GitHub Blog*, March 2023, retrieved May 30, 2024. [Online]. Available: <https://github.blog/2023-03-22-github-copilot-x-the-ai-powered-developer-experience/>
- [15] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, “Codebert: A pre-trained model for programming and natural languages,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov 2020, pp. 1536–1547.
- [16] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, S. Yih, L. Zettlemoyer, and M. Lewis, “InCoder: A generative model for code infilling and synthesis,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [17] GitHub, “Introducing github copilot: your ai pair programmer,” *GitHub Blog*, June 2021, retrieved May 30, 2024. [Online]. Available: <https://github.blog/2021-06-29-introducing-github-copilot-ai-get-code-done-faster/>
- [18] B. Goertzel, “Artificial general intelligence: Concept, state of the art, and future prospects,” *Journal of Artificial General Intelligence*, vol. 5, no. 1, pp. 1–46, 2014, submitted 2013-2-12, Accepted 2014-3-15.
- [19] D. Guo, S. Lu, N. Duan, Y. Wang, M. Zhou, and J. Yin, “Unixcoder: Unified cross-modal pre-training for code representation,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7212–7225.
- [20] M. Izadi, R. Gismondi, and G. Gousios, “Codefill: Multi-token code completion by jointly learning from structure and naming sequences,” in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE ’22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 401–412.
- [21] M. Izadi, J. Katzy, T. van Dam, M. Otten, R. M. Popescu, and A. van Deursen, “Language models for code completion: A practical evaluation,” in

2024 *IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. New York, NY, USA: ACM, April 14–20 2024, pp. 1–13.

- [22] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson, 2009.
- [23] J. Kemper. (2023, March) Openai kills its codex code model, recommends gpt3.5 instead. Retrieved May 27, 2024. [Online]. Available: <https://thedecoder.com/openai-kills-its-codex-code-model-recommends-gpt3-5-instead/>
- [24] S. Kim, J. Zhao, Y. Tian, and S. Chandra, “Code prediction by feeding trees to transformers,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 150–162.
- [25] H. H. Koester and S. Levine, “Effect of a word prediction feature on user performance,” *Augmentative and alternative communication*, vol. 12, no. 3, pp. 155–168, 1996.
- [26] Z. Liu, “Chatgpt will command more than 30,000 nvidia gpus: Report,” *Tom’s Hardware*, Mar 2023, archived from the original on May 30, 2024. Retrieved November 2, 2023.
- [27] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, G. Li, L. Zhou, L. Shou, L. Zhou, M. Tufano, M. Gong, M. Zhou, N. Duan, N. Sundaresan, S. K. Deng, S. Fu, and S. Liu, “Codexglue: A machine learning benchmark dataset for code understanding and generation,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.
- [28] C. Luo, “A report on automatic code completion,” 03 2017.
- [29] C. Metz and T. Mickle, “Openai completes deal that values the company at \$80 billion,” *The New York Times*, Feb 2024, retrieved May 30, 2024.
- [30] C. Omar, Y. Yoon, T. LaToza, and B. Myers, “Active code completion,” in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 859–869.
- [31] OpenAI, “Chatgpt: Optimizing language models for dialogue,” 2022. [Online]. Available: <https://web.archive.org/web/20221130180912/https://openai.com/blog/chatgpt/>
- [32] J. K. Rask, F. P. Madsen, N. Battle, H. D. Macedo, and P. G. Larsen, “The specification language server protocol: A proposal for standardised lsp extensions,” 2022.
- [33] V. Raychev, M. Vechev, and E. Yahav, “Code completion with statistical language models,” in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2014, pp. 419–428.
- [34] R. Rodriguez-Echeverria, J. L. Canovas Izquierdo, M. Wimmer, and J. Cabot, “Towards a language server protocol infrastructure for graphical modeling,” in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven*



*Engineering Languages and Systems*, ser. MODELS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 370–380.

- [35] E. Roth, “Microsoft spent hundreds of millions of dollars on a chatgpt super-computer,” *The Verge*, Mar 2023, archived from the original on May 30, 2023. Retrieved March 30, 2023.
- [36] Staff, “Github copilot – november 30th update,” *GitHub Blog*, November 2023, retrieved May 30, 2024. [Online]. Available: <https://github.blog/changelog/2023-11-30-github-copilot-november-30th-update/>
- [37] K. Staff, “Microsoft-backed openai valued at \$80bn after company completes deal,” *The Guardian*, Feb 2024, retrieved May 30, 2024.
- [38] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, “Intellicode compose: Code generation using transformer,” in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1433–1443.
- [39] A. Svyatkovskoy, S. Lee, A. Hadjitofi, M. Riechert, J. Franco, and M. Alamanis, “Fast and memory-efficient neural code completion,” *arXiv preprint arXiv:2004.13651*, 2020.
- [40] T. V. C. Team, “Visual studio code 1.0!” *Visual Studio Code Blog*, April 2016, retrieved June 1, 2024.
- [41] Unknown, “Tabnine: Coding in vs code with the help of an ai assistant,” *learn.microsoft.com*, March 2021, retrieved April 19, 2024.
- [42] J. Vincent, “This ai-powered autocompletion software is gmail’s smart compose for coders,” *The Verge*, July 2019, retrieved May 19, 2024.
- [43] Y. Wang and H. Li, “Code completion by modeling flattened abstract syntax trees as graphs,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 14 015–14 023.
- [44] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, “Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.
- [45] S. Zhao, “Github copilot chat now generally available for organizations and individuals,” *GitHub Blog*, December 2023, retrieved May 30, 2024. [Online]. Available: <https://github.blog/2023-12-29-github-copilot-chat-now-generally-available-for-organizations-and-individuals/>
- [46] S. Z. Zhao, “Smarter, more efficient coding: Github copilot goes beyond codex with improved ai model,” *GitHub Blog*, July 2023, retrieved May 27, 2024. [Online]. Available: <https://github.blog/2023-07-28-smarter-more-efficient-coding-github-copilot-goes-beyond-codex-with-improved-ai-model/>
- [47] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, “A survey of large language models,” 2023.

- [48] Q. Zhou, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He *et al.*, “A comprehensive survey on pretrained foundation models: A history from bert to chatgpt,” *arXiv preprint arXiv:2302.09419*, 2023.