تطبيقات هندسة أوامر النماذج اللغوية الكبيرة (LLMs) في التدريب على مهام هندسة البرمجيات: مراجعة منهجية

أ. إيمان عبده عوض [1]

(قدم للنشر 1445/06/05هـ -وقبل 22 /11/ 1445هـ)

المستخلص: هدفت هذه الدراسة إلى إجراء مراجعة منهجية للتعرف على النماذج اللغوية الكبيرة التي تم توظيفها في تعليم وأتمتة مهام هندسة البرمجيات، وتحليل اتجاهاتها البحثية، والتعرف على أساليب هندسة الأوامر لتحسين مخرجات النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات، وتحديد أبرز تحديات تطبيقها. ونظراً لحداثة هذا المجال، استهدفت المراجعة جميع الدراسات المنشورة عام 2023م في قاعدة بيانات (Web of Science) ومحرك البحث (Google Scholar)، وباتباع إرشادات (Kitchenham) للمراجعة المنهجية، وتلخيص النتائج وفق القائمة المرجعية ومخطط (PRISMA) تم تحديد (35) ورقة بحثية انطبقت عليها معايير التضمين، وبالتحليل الموضوعي للدراسات، توصلت الدراسة إلى أن النماذج اللغوية الأكثر توظيفاً في أتمتة مهام هندسة البرمجيات مرتبة تصاعدياً كالتالي: (ChatGPT) ثم (GPT 3.5) ثم (Codex) ثم (GPT4) ثم (Copilot)، كما أن مهمة البرمجة من أكثر الاتجاهات البحثية لتطبيقات النماذج اللغوية الكبيرة، يليها مهمة التصميم، ثم مهمة اختبار وصيانة البرمجيات، وأن أسلوب الأوامر الأساسية (Zero-Shot, Few-Shot) من أكثر أساليب هندسة الأوامر للنماذج اللغوية الكبيرة استخداماً في أتمتة مهام هندسة البرمجيات، وتم رصد تحديات توظيف النماذج اللغوية الكبيرة في مجال هندسة البرمجيات وتصنيفها إلى تحديات تقنية ترتبط ببنية وآلية عمل النماذج اللغوية الكبيرة، وتربوية تمثلت في ضرورة تطوير أصول التدريس وأساليب التقويم لتعليم هندسة البرمجيات في ظل انتشار استخدام تطبيقات النماذج اللغوية الكبيرة، وبحثية تمثلت في الحاجة للمزيد من الدراسات التطبيقية، ودراسات الفاعلية، لتوجيه استخدام هذه النماذج بطرق فعالة، وعادلة، وأخلاقية.

الكلمات المفتاحية: الذكاء الاصطناعي التوليدي، التعلم العميق، معالجة اللغة الطبيعية، المحولات، Chat GPT.

Applications of Prompt Engineering for Large Language Models (LLMs) on Training of Software Engineering Tasks: A Systematic Review

Eman A. Awad[1]

(Submitted 18-12-2023 and Accepted on 30-05-2024)

Abstract: This study aimed to conduct a systematic review to identify the large language models (LLMs) that have been employed in teaching and automating software engineering tasks, analyze their research trends, identify prompts engineering techniques to improve the outputs of large language models in automating software engineering tasks, and identify the most prominent challenges of their application. Due to the novelty of the field, the review targeted all studies published in 2023 in the Web of Science database and the search engine (Google Scholar), and by following the (Kitchenham) guidelines for systematic review, and summarizing the results according to the reference list and the (PRISMA) scheme, (35) papers that met the inclusion criteria were identified, and through thematic analysis of the studies, the study concluded that the most widely used large language models in automating software engineering tasks are arranged in ascending order as follows: (ChatGPT, GPT 3.5, Codex, GPT4, Copilot), and the programming task is one of the most research trends for applications of Large language models, followed by the design task, then the task of software testing and maintenance, and that the basic prompt technique (Zero-Shot, Few-Shot) is one of the most widely used prompt engineering techniques for large language models in automating software engineering tasks. The challenges of employing large language models in the field of Software engineering and its classification into technical challenges related to the structure and mechanism of operation of large language models, educational challenges represented by the necessity of developing pedagogy and evaluation methods for teaching software engineering in light of the widespread use of applications of large language models, and research challenges represented by the need for more applied studies and effectiveness studies.

Keywords: Generative Artificial Intelligence, Deep Learning, Natural Language Processing, Transformers, Chat GPT.

(1) PhD student - College of Graduate Studies in Education, King Abdulaziz University

(1) طالبة دكتوراه-كلية الدراسات العليا التربوية جامعة الملك عبدالعزيز

E-mail: eawadh0002@stu.kau.edu.sa

المقدمة

أدى التطور المتسارع في تقنيات الذكاء الاصطناعي (Artificial Intelligence) ووفرة البيانات وتنوعها إلى ظهور أنظمة متخصصة في معالجة اللغة الطبيعية (Natural Language Processing) تهدف إلى فهم اللغات البشرية وخصائصها، وتتنوع مهام هذه الأنظمة من أساليب التحليل البسيطة للغة إلى المعقدة والتي تستخلص المعنى والسياق من النصوص المختلفة. ومؤخراً، شهد مجال معالجة اللغة الطبيعية (NLP) وتطبيقاتها ثورة تقنية بظهور نماذج لغوية متطورة تعرف بنماذج الذكاء الاصطناعي التوليدي (Generated AI) أو النماذج اللغوية الكبيرة (Large Language Models) قائمة على خوارزميات التعلم العميق (Deep Learning) ولديها القدرة على معالجة اللغة الطبيعية البشرية لتوليد أنواع عديدة من المحتوى كالنصوص، والصور، والأصوات، وأكواد لغات البرمجة.

ونظراً للإمكانات المتقدمة للنماذج اللغوية الكبيرة (LLMs) في تطبيقات معالجة اللغة الطبيعة (NLP)، فقد اكتسبت اهتماماً كبيراً من قبل المجتمعات التقنية والتجارية، والأكاديمية، وبرزت العديد من التطبيقات القائمة على النماذج اللغوية الكبيرة (LLMs) لتخصيص استخدامها في مهام محددة، والتي من أشهرها نموذج (GPT) من شركة (Open AI) النموذج الأساس لتطبيق روبوت المحادثة (ChatGPT)، والنموذج الأساس (Gemini) لتطبيق روبوت المحادثة (Bard) من شركة (Google)، فمنذ إطلاق (ChatGPT) في أواخر نوفمبر 2022 م بلغ عدد المستخدمين للتطبيق مليون مستخدم خلال خمسة أيام، وما يقارب مليار ونصف زائر خلال شهر (Duarte, 2023).

وتجدر الإشارة إلى أن العديد من المنظمات والمؤسسات ذات العلاقة بتجويد التعليم والبحث العلمي سارعت باستكشاف وتقنين تطبيقات هذه التقنية في السياقات التعليمية والبحثية، فقد أصدرت منظمة اليونيسكو تقرير إرشادات استخدام نماذج الذكاء الاصطناعي التوليدي في التعليم والبحث (UNESCO, 2023)، وبالمثل أصدرت منظمة الاقتصاد والتعاون والتنمية (OECD) تقرير الاعتبارات السياسية الأولية للذكاء الاصطناعي التوليدي في عدد من المجالات ومنها التعليم، والبحث العلمي، والكتابة الأكاديمية، وحقوق الملكية الفكرية (Lorenz et al., 2023)، ومؤخراً صدر عن الهيئة السعودية للبيانات و الذكاء الاصطناعي تقرير الذكاء الاصطناعي التوليدي في التعليم و الذي تناول أبرز التطبيقات و التحديات والتوجهات المستقبلية لهذه التقنية في السياق التعليمي و الأكاديمي (SDAIA, 2023).

ومن المجالات التي نالت اهتماماً واسعاً لتطبيقات النماذج اللغوية الكبيرة (LLMs)، مجال هندسة البرمجيات (Software Engineering) كامتداد للمجال البحثي "تعلم الآلة (Machine Learning) في هندسة البرمجيات" (ML4SE) والذي شهد العديد من التطورات التطبيقية والبحثية التي استهدفت تطوير البرمجيات المعززة بالذكاء الاصطناعي (AI-Augmentation SELC) و ذلك بأتمتة مهام دورة حياة هندسة البرمجيات (SELC) بمستويات متفاوتة، لتمكين مطوري البرمجيات من تصميم، وتطوير، وصيانة، ونشر البرمجيات بأقل وقت، وجهد، وتكلفة (Kotti et al., 2023; Ozkaya, 2023).

وفي هذا الإطار، تتضمن مهام دورة حياة هندسة البرمجيات عملية تحليل المتطلبات، وتصميم البرمجيات، وكتابة الكود البرمجي، والتحقق من

صحته، وتوليد حالات أو بيانات الاختبار الملائمة له، ونشر البرمجيات، والصيانة الدورية لها، إضافة إلى التطوير التعاوني، والبرمجة التعاونية، وتعزيز جودة البرمجيات، والتي غالباً ما تتطلب مهندسي برمجيات على قدر عالي من الخبرة، والذي بدوره يضيف على تعقيد مهام تطوير البرمجيات تكلفة أكبر في مجال الموارد البشرية، وهو ما دفع الباحثين وكبرى الشركات التقنية إلى الاستفادة من التطور الحالي في النماذج اللغوية الكبيرة لرفع مستوى أتمتة مهام هندسة البرمجيات (Ozkaya, 2023).

وعلى الرغم من التطبيقات العديدة للنماذج اللغوية الكبيرة في مجال أتمتة مهام هندسة البرمجيات والتي شهدت ارتفاعاً كبيراً مع انطلاق النماذج المصممة خصيصاً لأتمتة بعض مهام هندسة البرمجيات مثل، نموذجي (Codex) و(GitHub Copilot) إضافة إلى نماذج (GPT) وذلك لأتمتة مهمة كتابة الكود البرمجي بتحويل الأوامر المدخلة له إلى تعليمات برمجية، إلا أنه سرعان ما تم رصد جانب من التحديات المرتبطة باستجابات هذه النماذج للأوامر المدخلة لها (Fu et al., 2023; Denny et al., 2023; MacNeil et al, 2023; Kiesler et al., 2023).

ومن هذا المنطلق، تسعى هذه الدراسة إلى استكشاف التوجهات البحثية الحديثة لتطبيقات النماذج اللغوية الكبيرة (LLMs) في مجال هندسة البرمجيات بتقديم مراجعة منهجية للدراسات المنشورة في هذا المجال.

مشكلة الدراسة

ترتبط استجابة النماذج اللغوية الكبيرة بعدد من العوامل منها البيانات التي تم تدريب هذه النماذج عليها، وضبط أوزان معلمات النموذج، والأوامر (Prompts) التي تشكل مدخلات المستخدم و وسيلة اتصاله مع النموذج وذلك لتوليد استجابة محددة (Ozdemir, 2023)، وتعتبر الأوامر عنصر جوهري في تحفيز النماذج اللغوية الكبيرة لتوليد الاستجابة المتوافقة مع أهداف المستخدم، كما أشارت العديد من الدراسات التي استهدفت محاولات تقصي أثر الأساليب المختلفة للأوامر على استجابة النماذج اللغوية الكبيرة إلى ضرورة كتابة الأوامر بالأسلوب الذي يوجه النموذج لتوليد الاستجابة المطلوبة (Alto, 2024; Phoenix & Taylor, 2024).

تُعرف عملية تصميم الأوامر للنماذج اللغوية الكبيرة وتحسينها وتعديلها بمفهوم هندسة الأوامر (Prompts Engineering)، وتتضمن أساليب متنوعة ومستويات من الخبرة بكتابة الأوامر، ومع تطور النماذج اللغوية الكبيرة أصبحت هندسة الأوامر مهارة بالغة الأهمية لتسخير إمكانيات النماذج اللغوية الكبيرة، وتعزيز قدرتها الاستدلالية، للاستفادة منها في مجموعة متنوعة من التطبيقات، مثل أتمتة مهام هندسة البرمجيات (Wei et al., 2023).

وبالرغم من استحواذ مجال الحوسبة عموماً على أغلب تطبيقات وأبحاث النماذج اللغوية الكبيرة (Liu et al., 2023)، إلا أن العديد من الدراسات أشارت إلى أنه مجال لا يزال بحاجة إلى المزيد من التطبيقات والدراسات المستقبلية للتغلب على بعض قيود، وتحيزات، وأخطاء استجابات هذه النماذج (Liu et al., 2023; Denny et al., 2023; Nguyen-Duc et al., 2023)، فعلى سبيل المثال، رصدت دراسة (Lau & Guo, 2023) عدداً من قيود النماذج اللغوية الكبيرة في مهمة توليد الكود البرمجي تمثلت في عدم الدقة، وضعف جودة التعليمات البرمجية وعدم ملائمتها للحالات التي تتطلب مستوى أمن عالي في البرنامج، وعدم ضمان تكرار استجابة النموذج حتى مع تكرار ذات الأمر، إضافة إلى أن مخرجاتها قد تكون معقدة

للمبرمجين المبتدئين، أو أنها تتطلب مستوى معين من الخبرة بهندسة الأوامر .

وعلى الجانب الآخر، أشارت دراسة (Shin et al., 2023) إلى تفوق نموذج (GPT 4) في مهمة توليد الكود البرمجي مع الأوامر المتكررة التي تتضمن تعليمات، وتغذية راجعة، وإضافة تفاصيل أكثر عن سياق المهمة للنموذج مقارنة بالأوامر البسيطة، كما أشارت دراسة (Zhang et al., 2023) إلى تفوق (ChatGPT) في مهمة إصلاح البرمجيات على نموذجي (CodeT5) و(PLBART) بدقة التنبؤ بأخطاء البرمجيات باستخدام ثلاثة أنواع مختلفة من الأوامر.

وعلى اعتبار أن التعليم والتدريب المستمر أحد الجوانب الأساسية لتطوير مهارات مهندسي البرمجيات، تطرقت دراسة كلاً من (MacNeil et al., 2023; Lau & Guo, 2023; Denny et al., 2023) إلى الآثار المستقبلية لاعتماد مطوري البرمجيات، والمبتدئين في تعلم البرمجة على النماذج اللغوية الكبيرة في توليد الأكواد البرمجية والذي قد ينطوي عليه ضعف بعض المهارات الأساسية للمبرمجين كمهارات حل المشكلات، ومهارات التفكير الحاسوبي، ومهارات ما وراء المعرفة.

وتأسيساً على ما سبق ذكره، تسعى الدراسة الحالية إلى تسليط الضوء على التوجهات البحثية الحديثة لتطبيقات النماذج اللغوية الكبيرة (LLMs) في مجال هندسة البرمجيات بجمع وتحليل وتقييم الدراسات المنشورة بطريقة منهجية.

أسئلة الدراسة

- ما النماذج اللغوية الكبيرة التي تم توظيفها في أتمتة مهام هندسة البرمجيات؟
- ما الاتجاهات البحثية لتطبيقات النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات؟
- ما أساليب هندسة الأوامر لتحسين مخرجات النماذج اللغوية الكبيرة في مهام هندسة البرمجيات؟
- ما تحديات توظيف النماذج اللغوية الكبيرة في تعليم هندسة البرمجيات؟

أهداف الدراسة

- التعرف على النماذج اللغوية الكبيرة التي تم توظيفها في مجال أتمتة مهام هندسة البرمجيات.
- الكشف عن الاتجاهات البحثية لتطبيقات النماذج اللغوية الكبيرة في مجال أتمتة مهام هندسة البرمجيات.
- التعرف على أساليب هندسة الأوامر لتحسين مخرجات النماذج اللغوية الكبيرة في مهام هندسة البرمجيات.
- الكشف عن تحديات توظيف النماذج اللغوية الكبيرة في مجال تعليم هندسة البرمجيات.

أهمية الدراسة

تكمن **الأهمية النظرية** للدراسة في أنها تواكب التطورات الحالية في مجال تقنيات الذكاء الاصطناعي التوليدي وتحديداً النماذج اللغوية الكبيرة، كما أنه يتوافق مع جهود المؤسسات التربوية والمنظمات ذات العلاقة بتأطير تطبيقات تقنيات الذكاء الاصطناعي التوليدي وتوظيفها في مجال تعليم هندسة البرمجيات.

وتكمن **الأهمية التطبيقية** في تقديم تطبيقات وتجارب لهندسة أوامر النماذج اللغوية الكبيرة وذلك لتحسين مخرجاتها في سياق هندسة البرمجيات، والتي قد تفيد المعلمين والمطورين أو الباحثين بتطبيقها وتطويرها في ذات السياق أو في مهمات وتطبيقات أخرى.

حدود الدراسة

اقتصرت الدراسة الحالية على الحدود الآتية:

الحدود موضوعية: تمثلت في أساليب هندسة أوامر النماذج اللغوية الكبيرة (LLMs) وتطبيقاتها في

أتمتة مهام هندسة البرمجيات، وأبرز فرص وتحديات توظيفها في تعليم هندسة البرمجيات.

الحدود مكانية: الدراسات والمقالات العلمية المنشورة قاعدة بيانات شبكة العلوم (Web of Science)، ومحرك الباحث العلمي (Google Scholar).

الحدود زمانية: الدراسات والمقالات العلمية المنشورة عام 2023م.

مصطلحات الدراسة

النماذج اللغوية الكبيرة (LLMs): من نماذج الذكاء الاصطناعي القائمة على نوع من خوارزميات التعلم العميق يعرف بالمحولات (Transformers) تم تدريبها على مجموعة كبيرة من البيانات النصية، وتستخدم في إنشاء حلول وتطبيقات معالجة اللغة الطبيعية (Natural Language Process) كتصنيف النص، الترجمة الآلية، توليد النص، ومن أمثلتها (GPT4) و(Gemini) و(Claude) (Blete & Caelen, 2023).

هندسة الأوامر: عملية تكرارية تنطوي على تصميم وتعديل الأوامر المدخلة للنماذج اللغوية الكبيرة لتحسين دقتها في توليد استجابة محددة (Blete & Caelen, 2023).

هندسة البرمجيات: فرع من فروع علوم الحاسب يهتم بتطبيق مبادئ التصميم الهندسي في تصميم، وتطوير، واختبار، وصيانة البرمجيات (Tsui et al., 2022).

الخلفية العلمية للدراسة

النماذج اللغوية الكبيرة (Large Language Models):

في عام 2017م تم استحداث نموذج ذكاء اصطناعي متطور قائم على تقنية التعلم العميق (Deep Learning) من قبل فريق (Google Brain) أُطلق عليه اسم المحول أو (Transformer) وتم استخدامه بحثياً وتقنياً في تطبيقات معالجة اللغة الطبيعية (Natural Language Process) مثل، تحليل النصوص، والترجمة، وتحويل النص إلى كلام (Ozdemir, 2023)؛ وفي عام 2018م طورت شركة Open AI نموذج المحول التوليدي المدرب مسبقاً (Generative Pre-trained Transformer) والذي يُعرف بنموذج (GPT) وذلك لإنشاء نصوص تشبه النصوص التي ينشئها الإنسان، و عملت شركة (Open AI) على تطوير نموذج (GPT) بسلسلة من الإصدارات المطورة للنموذج من حيث عدد معلمات النموذج وحجم البيانات المستخدمة في تدريبيه، كما قامت في عام 2022م بإطلاق ChatGPT وهو عبارة عن روبوت محادثة (Chat bot) يستخدم النموذج اللغوي الكبير (GPT) لإنشاء نص يشبه النص الذي ينشئه الإنسان بناءً على أوامر مكتوبة (Blete & Caelen, 2023).

تعرف النماذج اللغوية الكبيرة (Large Language Models) بأنها نماذج ذكاء اصطناعي قائمة على تقنية المحولات (Transformers) يتم تدريبها على كميات هائلة من البيانات النصية حتى تصبح قادرة على الفهم الدلالي للغة البشرية ونمذجتها، ويتم استخدامها في مهام عديدة متعلقة باللغة مثل تصنيف وإنشاء النصوص، وقد أظهرت نتائج مبهرة فيما يتعلق باتساق ودقة سياق النصوص بدرجة تشبه النصوص التي ينشئها الإنسان (Ozdemir, 2023).

تعمل النماذج اللغوية الكبيرة (LLMs) بتقسيم الجملة أو النص إلى وحدات صغيرة تعرف بالرموز أو (Tokens) وتعتبر هذه الرموز أصغر وحدة للمعنى الدلالي والمدخلات الأساسية للنماذج اللغوية الكبيرة (LLMs)، ويتم تدريب هذه النماذج على التنبؤ بالرمز التالي في الجملة بناءً على الرموز المدخلة وذلك لمهام توليد النصوص (Blete & Caelen, 2023).

ويُشار إلى المحتوى الناتج عن الذكاء الاصطناعي بمصطلح (AI-Generated Content) أو (AIGC)، وقد أحدث ثورة هائلة في مشهد المحتوى الرقمي، واكتسب الكثير من الاهتمام البحثي والتجاري نظراً للإمكانيات الكبيرة لنماذج الذكاء الاصطناعي التوليدي في توليد مختلف أنواع المحتوى الرقمي، مثل نموذج (GPT4) لتوليد النصوص، ونموذج (DALL-E2) لتوليد الصور، ونموذج (Codex) لتوليد الكود البرمجي، وتشمل تطبيقات هذه النماذج مهام معالجة اللغة الطبيعة مثل التصنيف والترجمة، إضافة إلى البحث الدلالي واسترجاع المعلومات، وتحويل النص إلى كود برمجي، وتوليد الصور من النصوص، وتوليد النصوص لمهام عديدة مثل، كتابة بريد الكتروني، أو موضوع لمدونة، أو التخطيط للكتابة الأكاديمية، كما يتم استخدامها كأساس لبنية المحادثة وتوليد النصوص في روبوتات المحادثة مثل (ChatGPT) و(Gemini) (Ozdemir, 2023; Becker et al., 2023).

هندسة الأوامر (Prompt Engineering):

مع تسارع وتيرة الابتكار في تقنيات الذكاء الاصطناعي التوليدي والنماذج اللغوية الكبيرة، برزت الحاجة إلى تسخير إمكانيات النماذج اللغوية الكبيرة بشكل فعال في التطبيقات المختلفة لتحقيق الأهداف المرجوة، وبما أن النماذج اللغوية الكبيرة تتعامل مع اللغة الطبيعة كمدخلات لتوليد الاستجابات، فذلك يتطلب صياغة دقيقة للأوامر التي يتم توجيهها للنموذج لضمان توليد مخرجات صحيحة ومتوافقة مع سياق المهمة، والذي بدوره ساهم في ظهور مفهوم هندسة الأوامر (Prompt Engineering) (Alto, 2024).

تُعتبر هندسة الأوامر حجر الأساس لتحقيق الفائدة القصوى من روبوتات المحادثة القائمة على النماذج اللغوية الكبيرة مثل (ChatGPT) (Abedi et al., 2023)، ويُشير مفهوم هندسة الأوامر إلى عملية تصميم وتحسين أوامر الإدخال من قبل المستخدم والتي عادةً ما تكون على شكل أوامر نصية إلى النموذج اللغوي لتوليد مخرجات عالية الجودة، وتتطلب هندسة الأوامر الدقة، والإبداع، والتكرار بالمحاولة والخطأ، للوصول إلى النتائج المثلى لسياق المهمة المطلوبة، وهو ما يسهم في فهم قدرات وقيود النماذج اللغوية الكبيرة (Alto, 2024).

ومن المبادئ الأساسية التي يجب اتباعها عند كتابة الأوامر (Phoenix & Taylor, 2024):

1) أن تتضمن تعليمات واضحة تشتمل على الهدف، تنسيق المخرجات المطلوب، سياق وحدود المهمة.
2) تجزئة المهام المعقدة إلى مهام فرعية أبسط وتوليد مخرجاتها بأكثر من أمر، على سبيل المثال، عندما تكون المهمة تلخيص النصوص الطويلة قد يتم تجزئتها لأكثر من أمر للنموذج بدءاً من استخراج الأفكار الرئيسية من النص، ثم إعادة كتابتها بالربط المنطقي بين الأفكار الرئيسية، ثم إعادة توليدها حسب التنسيق المحدد.
3) توجيه النموذج لتوليد المبررات حول استجابته، تجبر النموذج على إعادة التفكير في مخرجاته وبالتالي تساعده على اكتشاف الأخطاء وتحسين دقة المخرجات، كما أنها توفر لنا نظرة ثاقبة حول كيفية عمل النموذج.
4) توجيه النموذج إلى توليد مخرجات متعددة للمهمة الواحدة، ثم إعادة توجيه أمر له باختيار الاستجابة الأفضل وفق معايير أو سياق محدد للمهمة.
5) استخدام علامات التحديد مثل: "، #، >،] ولذلك لمساعدة النموذج على فهم الأمر وتحديد العلاقات بين أجزاءه.

ومن أساليب هندسة الأوامر، الأوامر الأساسية (Basic Prompts) وتشمل أسلوب التعلم بدون أمثلة

(Zero-shot) وهو عبارة عن أوامر توجه للنموذج بدون أمثلة إضافية، وأسلوب التعلم بأمثلة قليلة (Few-shot) وهو عبارة عن أوامر توجه للنموذج وتشتمل على عدد من الأمثلة (خليفة، 2023)، و يعد أسلوب التعلم بأمثلة قليلة (Few-shot) من الأساليب القوية التي تتيح تخصيص استجابات النماذج دون التدخل في بنيتها العامة، كما يتم استخدام أسلوب التفكير المتسلسل (Chain of Thought) لتحفيز النموذج على إنشاء خطوات استدلالية وسيطة في المهام المعقدة التي تتطلب التفكير قبل الاستجابة، حيث يقوم النموذج بتبرير استجابته بتوليد سلسلة من خطوات التفكير المنطقية، و يمكّن دمج هذا الأسلوب في هندسة الأوامر مع أسلوب الأوامر بأمثلة قليلة (Few-shot) النموذج من توليد أفضل النتائج (Alto, 2024).

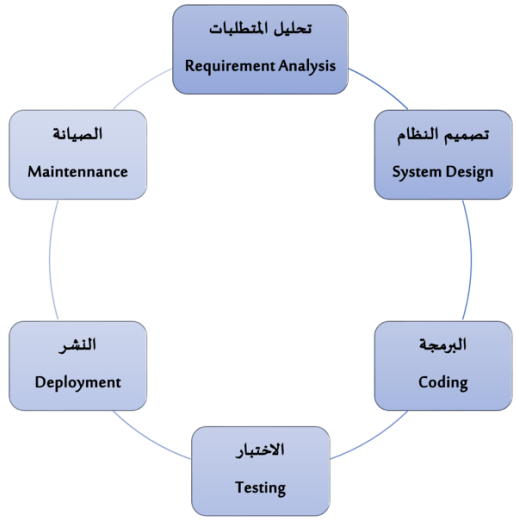
وفي سياق الجمع بين تنفيذ الإجراء والاستدلال، وتمكين النماذج اللغوية الكبيرة من الوصول إلى الموارد الإضافية مثل البحث في الويب أو قواعد البيانات لتحسين استجابتها، أشار (Yao et al., 2022) إلى أسلوب (ReAct) وهو اختصار لجملة (Reasoning and Acting)، حيث تعزز أوامر (ReAct) التي تم دمجها في إطار عمل (LangChain) عملية الاستدلال للنموذج وتكييف استجابته بناءً على المعلومات الخارجية.

هندسة البرمجيات (Software Engineering):

يهدف مجال هندسة البرمجيات إلى تطوير برمجيات فعالة، عالية الجودة، وقابلة للصيانة، وتتضمن هذه العملية مهام متعددة تشمل تحليل المتطلبات، تصميم النظام، تطوير البرمجيات، واختبارها، ونشرها، وصيانتها، إضافة إلى تطبيق معايير الجودة في تطوير البرمجيات (Kotti et al., 2023).

وعلى مدى العقود الماضية، شهد مجال هندسة البرمجيات تطوراً كبيراً فيما يتعلق بتعزيز مستوى الأتمتة في مهام هندسة البرمجيات، وذلك بتطبيق تقنيات الذكاء الاصطناعي على دورة حياة تطوير البرمجيات (Software Development Life Cycle) والذي مكّن المطورين من تصميم وتطوير واختبار ونشر البرمجيات بأقل وقت وجهد وتكلفة، وذلك بمساعدة أدوات الذكاء الاصطناعي (Hou et al., 2023).

ففي دورة حياة تطوير البرمجيات المعززة بالذكاء الاصطناعي (AI-Augmentation SDLC) تتم أتمتة عمليات تحديد المتطلبات، واختبار البرمجيات، والتنبؤ بأخطائها ومدى قابليتها للصيانة، وتوليد الأكواد البرمجية ومراجعتها وإصلاحها، وتقدير تكلفة تطوير البرمجيات، ويعتبر تعزيز مستوى أتمتة مهام هندسة البرمجيات من المجالات النشطة بحثياً في مجتمع هندسة البرمجيات (Ozkaya, 2023).



شكل 1: دورة حياة تطوير البرمجيات (SDLC)

ومؤخراً، تم توظيف النماذج اللغوية الكبيرة (LLMs) في العديد من مهام هندسة البرمجيات، كتحليل المتطلبات، وكتابة الأكواد البرمجية وشرح آلية

عملها، وتوليد التعليقات التوضيحية المصاحبة لها، واختبار البرمجيات وإصلاحها، وقد أثبت نموذج (Codex) القدرة على حل 72.31% من التحديات البرمجية المعقدة التي طرحها عليه المبرمجون بلغة (Python) (Hou et al., 2023).

منهجية الدراسة

تم إجراء هذه المراجعة بناءً على إرشادات ومراحل (Kitchenham & Charters, 2007) للمراجعة المنهجية والتي تم تنظيمها في ثلاثة مراحل هي: 1) التخطيط، 2) التنفيذ، 3) إعداد تقرير المراجعة، حيث تضمنت المرحلة الأولى بلورة الحاجة للمراجعة المنهجية بتحديد أسئلتها وأهدافها، وتطوير بروتوكول المراجعة، وفي المرحلة الثانية، تم البحث عن الدراسات وفق استراتيجية محددة، واختيار الدراسات ذات الصلة، واستخراج بياناتها و توليفها، وأخيراً، في المرحلة الثالثة تم تلخيص و استعراض النتائج وفق القائمة المرجعية ومخطط تدفق (PRISMA) (Sarkis-Onofre et al., 2021)، كما تم أتمتة عمليات المراحل السابقة باستخدام تطبيق (SR-Accelerator)، وفيما يلي وصف لجميع مراحل وإجراءات المراجعة المنهجية في هذه الدراسة.

1) التخطيط

1.1 استراتيجية البحث:

هدفت استراتيجية البحث إلى تحديد الدراسات الحديثة ذات الصلة بتطبيقات هندسة أوامر النماذج اللغوية الكبيرة (LLMs) في مجال هندسة البرمجيات، وتضمنت عملية البحث في قاعدة بيانات شبكة العلوم (Web of Science)، ولتوسيع نطاق البحث، تم أيضاً البحث في الباحث العلمي (Google Scholar) حيث أشار (Martín-Martín et al., 2021) أنه لا زال محرك البحث الأكثر شمولاً في تغطية المصادر العلمية والأكاديمية، واقتصر البحث على قواعد البيانات السابق ذكرها لوفرة الأوراق المنشورة حديثاً و التي تتوافق مع توجه الدراسة، كما أن موضوع الدراسة أحد التخصصات التي تغطيها تلك القواعد، إضافة إلى تنوع الاتجاهات البحثية في الأوراق المنشورة ذات العلاقة بالدراسة والمفتوحة المصدر.

1.2 معايير التضمين والاستبعاد:

تم اختيار الدراسات وفق معايير التضمين الموضحة في الجدول (1) أدناه:

جدول (1)

معايير التضمين والاستبعاد للدراسات في المراجعة المنهجية

المعيار	التضمين	الاستبعاد
الفترة الزمنية للنشر	الدراسات المنشورة في عام 2023م	الدراسات المنشورة قبل عام 2023م
اللغة	الإنجليزية	الدراسات المكتوبة بلغة غير الإنجليزية
نوع الدراسة	الأبحاث والمقالات العلمية المنشورة في المجلات العلمية المحكمة.	المراجعات، التقارير.
موضوع الدراسة	1. تطبيقات النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات. 2. أساليب هندسة أوامر النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات. 3. تطبيقات النماذج اللغوية الكبيرة في تعليم هندسة البرمجيات.	1. استبعاد الدراسات التي لم تستهدف تطبيقات النماذج اللغوية في أتمتة مهام هندسة البرمجيات. 2. استبعاد الدراسات التي لم تستهدف الجانب التطبيقي لهندسة أوامر النماذج اللغوية الكبيرة في سياق هندسة البرمجيات. 3. استبعاد الدراسات التي استهدفت توظيف النماذج اللغوية الكبيرة في السياقات التعليمية لتخصصات أخرى غير هندسة البرمجيات.

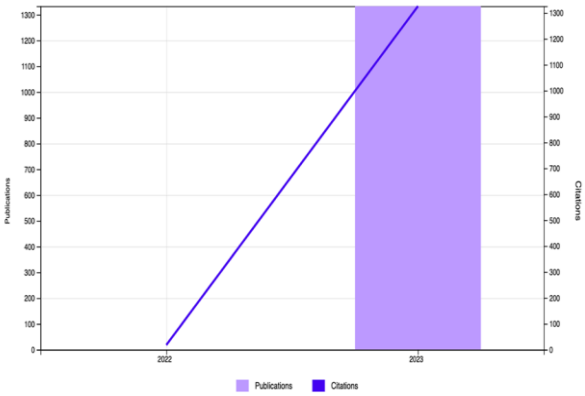
2) التنفيذ

2,1البحث:

تم تنفيذ استراتيجية البحث المحددة أعلاه بتاريخ 12/11/2023، وباستخدام المصطلحات الرئيسية التالية: "النماذج اللغوية الكبيرة"، "هندسة الأوامر"، "تطوير البرمجيات"، "هندسة البرمجيات"، "مهارات التفكير الحاسوبي"، والروابط المنطقية وفق الاستعلام التالي:

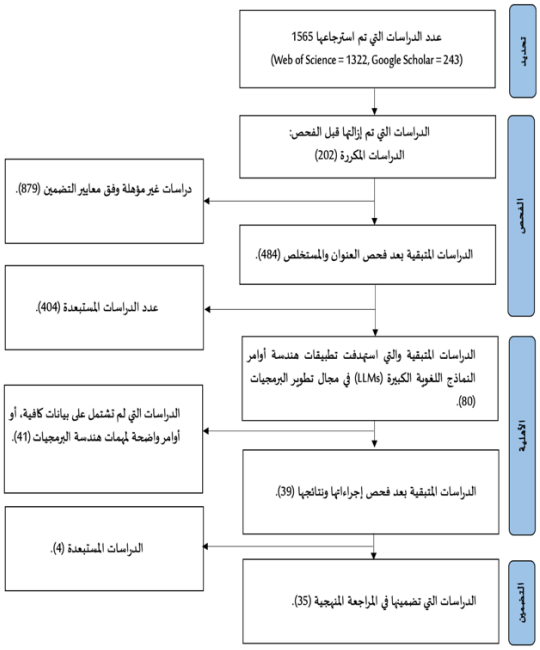
ALL=(English) AND SO=(large language model) AND ALL=(prompt engineering) AND ALL=(software engineering) OR ALL=(software development) OR ALL=(computational thinking)) AND (PY==("2023" OR "2024") AND ((DT==("REVIEW") AND PY=(2023-2023

وأسفرت نتائج عملية البحث عن (1565) دراسة (WOS =1322, Google scholar =243)، حسب ما هو موضح في الشكل (2).



شكل (2): نتيجة البحث في Web of science

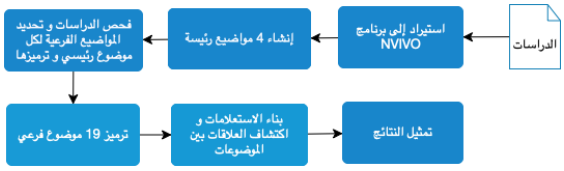
تم إزالة (202) دراسة مكررة، واستبعاد (879) دراسة خارج نطاق الدراسة، ونتج عن ذلك فحص عنوان ومستخلص (484) دراسة، وحسب معايير التضمين والاستبعاد، تم تضمين (80) دراسة استهدفت تطبيقات هندسة أوامر النماذج اللغوية الكبيرة (LLMs) في مجال تطوير البرمجيات لفحصها بالكامل والاطلاع على إجراءاتها ونتائجها، ونتج عن ذلك اختيار (35) دراسة للمراجعة المنهجية، والشكل (3) يوضح مخطط تدفق (PRISMA) لعملية اختيار الدراسات.



شكل (3):مخطط تدفق (PRISMA) لعملية اختيار الدراسات

2,2 استخراج البيانات وتحليلها

تم تحديد (35) دراسة ذات صلة بأهداف هذه المراجعة المنهجية، تم استيرادها إلى برنامج (NVIVO) لتنظيمها، وترميزها (Coding) تمهيداً لفهم التطبيقات البحثية، واستخلاص الرؤى ذات الأهمية للإجابة على أسئلة الدراسة، والشكل (4) التالي يوضح نظرة عامة على سير عمل مراجعة الدراسات في برنامج (NVIVO)، وسحابة الكلمات في الشكل (5) توضح أبرز الموضوعات التي تناولتها الدراسات المضمنة في المراجعة المنهجية وارتباطها بأهدافها.



شكل 4: سير العمل في برنامج NVIVO

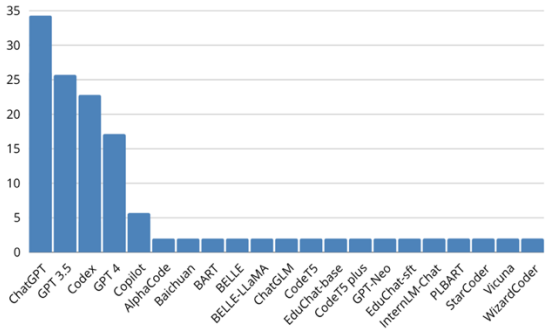
شكل(5): سحابة الكلمات لموضوعات الدراسات

نتائج المراجعة النهجية

- الإجابة عن السؤال الأول والذي نص على:

ما النماذج اللغوية الكبيرة التي تم توظيفها في أتمتة مهام هندسة البرمجيات؟

خلصت نتائج المراجعة المنهجية إلى أن توظيف النماذج اللغوية الكبيرة (LLMs) في مهام هندسة البرمجيات مجال بحثي متسارع النمو، و على الرغم من أن توظيف تقنيات الذكاء الاصطناعي المتمثلة في التعلم الآلي مجال بحثي شهد العديد من التطبيقات والتطورات على مدى الأعوام الماضية، إلا أن التطور الذي نشهده مؤخراً في تقنيات النماذج اللغوية الكبيرة (LLMs) وهندسة الأوامر، شجع الباحثين والمطورين لإجراء المزيد من عمليات الأتمتة لمهام هندسة البرمجيات باستخدام النماذج اللغوية الكبيرة، والتي كان من أكثرها استخداماً (**ChatGPT**) بنسبة (34.3 %) و في دراسة كلاً من (Ji et al., 2023; Yilmaz & Yilmaz, 2023; Lau & Guo, 2023; Scoccia, 2023; Kiesler et al., 2023; Nguyen-Duc et al., 2023; Zhang et al., 2023; Spasić et al., 2023; Denny et al., 2023; Wang et al., 2023; Yilmaz & Yilmaz, 2023)، يليه نموذج (**GPT 3.5**) بنسبة (25.7 %) وفي دراسة كلاً من (Huang et al., 2023; Roest, 2023; Ji et al., 2023; Martínez et al., 2023; Fu et al., 2023; MacNeil et al., 2023; Nam et al., 2023; Zelikman et al., 2023; Shin et al., 2023; G. Fan et al., 2023; A. Fan et al., 3023; Savelka et al., 2023)، ثم نموذج (**Codex**) بنسبة (22.8 %) و في دراسة كلاً من (MacNeil et al., 2023; Denny et al., 2023; Reeves et al., 2023; Liang et al., 2023; Kazemitabaar et al., 2023; Zelikman et al., 2023; Wang et al., 2023; Finnie-Ansley et al., 2022)، ونموذج (**GPT4**) بنسبة (17.14%) و في دراسة كلاً من (Ji et al., 2023; Nam et al., 203; Fan et al., 2023; Zelikman et al., 2023; Shin et al., 2023; Savelka et al., 2023)، وأخيراً نموذج (**Copilot**) بنسبة (5.7%) في دراسة (Lau &Guo, 2023; Prather et al., 2023)، كما تناولت دراسة (Fu et al., 2023) مقارنة نموذج (GPT 3.5) مع (14) نماذج أخرى في دقة توليد الكود البرمجي، و تفوق أداء (GPT 3.5) عليها جميعاً في دقة توليد الأكواد البرمجية، والشكل (5) التالي يوضح جميع النماذج اللغوية الكبيرة التي تم توظيفها في مجال هندسة البرمجيات:



شكل(6): النماذج اللغوية الكبيرة LLMs التي تم توظيفها في هندسة البرمجيات

- الإجابة عن السؤال الثاني والذي نص على:

ما الاتجاهات البحثية لتطبيقات النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات؟

توصلت نتائج المراجعة المنهجية لتطبيقات النماذج اللغوية الكبيرة في مجال هندسية البرمجيات إلى عدد من التطبيقات النوعية وذات القيمة للمجال، كما شملت غالبية مهام هندسة البرمجيات، إلا أن مهمة البرمجة كانت المهمة الأكبر من بين التطبيقات وبنسبة (55%)، شملت استخدام النماذج اللغوية الكبيرة في توليد الكود البرمجي (Code Generation)، وإكمال الكود البرمجي (Code Completion)، وتبسيط الكود البرمجي (Code Simplification)، وتوضيح أو تلخيص الكود البرمجي (Code Explanations) ، وتوليد تعليقات للكود البرمجي (Comments)، و توليد اقتراحات توليد الكود البرمجي (Hint Generation)، والبرمجة التعاونية (Collaborative Programming)، وبرمجة الروبوت (Robot Coding)، إضافة إلى اختبار مدى دقة النماذج اللغوية الكبيرة في توليد الكود البرمجي (Prather et al., 2023; Savelka et al., 2023; Denny et al., 2023; shin et al., 2023; Becker et al., 2023; Zelikman et al., 2023; Kazemitabaar et al., 2023; Liang et al., 2023; Scoccia, 2023; Denny et al., 2023; Yen et al., 2023; Fu et al., 2023; Lau & Guo, 2023; Ji et al., 2023; Zhang et al., 2023; MacNeil, 2023; Nam et al., 2023; Feng et al., 2023; Roest, 2023; Singh, 2023;).

وفي ذات السياق، كانت مهمة تصميم وإدارة تطوير البرمجيات من تطبيقات النماذج اللغوية الكبيرة في مجال هندسة البرمجيات و بنسبة (20%)، شملت توظيف النماذج اللغوية الكبيرة في أتمتة عمليات تحليل متطلبات البرمجيات، وأتمتة مهام تطوير البرمجيات وفق منهجية (Agile)، و أتمتة الدعم في اتخاذ قرارات التصميم، و أتمتة عمليات تهيئة البيانات لعمليات التحليل، وقد تطرقت لها دراسة كلاً من (Nguyen-Duc et al., 2023; Zhang et al., 2023; Shin et al., 2023; Fan et al., 2023; Gore et al., 2023; Tu et al., 2023; Huang et al., 2023; Zelikman et al., 2023)، يليها مهمة اختبار وصيانة البرمجيات بنسبة (25%)، و اشتملت توليد بيانات الاختبار (Test Data)، وعملية اختبار خلو البرمجيات من الأخطاء (Bugs) بكافة أنواعها سواء الأخطاء المنطقية (Logic Errors)، أو الأخطاء في بناء جملة الكود (Syntax Errors)، أو الأخطاء الوظيفية (Functional Errors)، وعملية تصحيح البرمجيات (Debugging) (Hung et al., 2023; Lau & Guo, 2023; Fu et al., 2023; Kiesler et al, 2023; Nam et al., 2023; Zhang et al., 2023; Sheese et al., 2023; Wang et al., 2023; Fan et al., 2023; Gore et al., 2023)، والشكل (6) التالي يوضح جميع الاتجاهات الحالية في دراسات المراجعة المنهجية.

شكل(7): تطبيقات النماذج اللغوية الكبيرة LLMs في أتمتة مهام هندسة البرمجيات

- الإجابة عن السؤال الثالث والذي نص على:

ما أساليب هندسة الأوامر لتحسين مخرجات النماذج اللغوية الكبيرة في مهام هندسة البرمجيات؟

تعتبر هندسة الأوامر للنماذج اللغوية الكبيرة العنصر الجوهري للاستفادة من قدرات النماذج اللغوية الكبيرة في أي مهمة، وفي مهمات هندسة البرمجيات تم تناول هندسة الأوامر بمختلف أساليبها، ففي دراسة (Martínez et al., 2023) تمت المقارنة بين اسلوبين من الأوامر لمهمة تحليل الكود بلغتي (C++)، (C) باستخدام نموذج (GPT 3.5)، الأسلوب الأول الأمر الصفري (Zero-

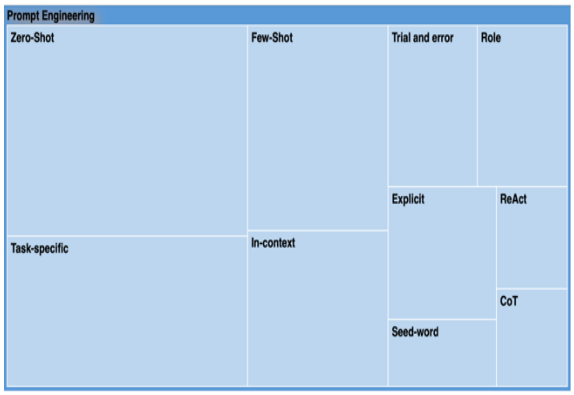
Shot) ولم تكن دقة النموذج عالية في تحليل الكود المعطى له حيث تراوحت بين (79.2% - 22.3%)، والأسلوب الثاني أمرين متصلة، الأمر الأول يشرح وظيفة الكود المعطى له، وبناءً على استجابته في الأمر الأول يحلل مكونات الكود في الأمر الثاني، وبهذه الطريقة حقق دقة استجابة بنسبة 100%.

وفي دراسة (Singh et al., 2023) تم استخدام نموذج (GPT 3) في توليد كود برمجي بلغة (Python) لسلسلة إجراءات تنفيذ مهمة منزلية محددة للروبوت (Robot) تحديداً التحكم في ذراع الروبوت، باستخدام الأوامر الصفرية (Zero-Shot)، وحيث أن تنفيذ المهمة من قبل الروبوت تتطلب معرفة كبيرة و دقيقة بالعالم المادي الذي يحيط به، إضافة إلى ضرورة تجزئة المهمة إلى مهمات فرعية صغيرة، تم استخدام الأوامر بأسلوب التعلم بأمثلة قليلة (Few-Shot) مع كتابة التعليقات (Comment) لتشجيع النموذج على تحسين أداءه من خلال التفكير المتسلسل (CoT)، وقد نجح النموذج في توليد الأكواد البرمجية الصحيحة.

و في ذات السياق، هدفت دراسة (Ji et al., 2023) إلى تحليل العلاقة السببية بين أسلوب الأوامر و دقة توليد الكود البرمجي بلغة (Python) في ثلاثة نماذج (GPT4)، (GPT 3.5)، (GPT-Neo)، وتم استخدام ثلاثة أساليب للأوامر، التعليمات المباشرة الأساسية، والأوامر بتحديد الدور (Role) للنموذج، والأوامر بتحديد سياق (In-Context) للنموذج، وكانت النتائج مع الأوامر الأساسية تشير إلى معدل نجاح عالي لنموذج (GPT 4) في دقة توليد الكود البرمجي، بينما كانت النتائج أكثر إيجابية مع النوع الثاني والثالث بتحديد دور و سياق في الأوامر مع نموذجي (GPT 4)، (GPT 3.5).

وبهدف توليد الكود البرمجي لتطوير مواقع الويب بلغة (JavaScript)، قارنت دراسة (MacNeil et al., 2023) بين دقة نموذجي (Codex)، (GPT3) في توليد الكود، وفي توليد توضيح للكود، باتباع أسلوب الأوامر بتجارب المحاولة والخطأ، وكشفت النتائج عن تفوق نموذج (GPT 3) على نموذج (Codex) في توليد الأكواد البرمجية من اللغة الطبيعية، وتوليد التفسيرات الضرورية للكود.

بينما قارنت دراسة (Fan et al., 2023) بين أسلوب الأوامر الصفرية (Zero-Shot) والأوامر بأمثلة قليلة (Few-Shot) في مهمة توليد أسئلة تتبع الكود البرمجي (Code Tracing) بلغة (Java) في نموذجي (GPT 4)، (GPT 3.5)، و قد حقق نموذج (GPT 4) دقة استجابة أكبر من نموذج (GPT 3.5) مع الأوامر الصفرية (Zero-Shot)، في حين أن كلا النموذجين كان لديهم تحيز أكبر نحو محاكاة الأمثلة المقدمة في أوامر (Few-Shot) و نتج عن ذلك استجابات أقل تنوعاً من الاستجابات مع الأوامر الصفرية، و هذا يدل على ضرورة استخدام أسلوب الأمر الملائم للمهمة، و تحسينه، و تكراره حتى يتمكن النموذج من تحقيق الهدف، و الشكل (7) التالي يوضح جميع أساليب هندسة الأوامر التي تم توظيفها في دراسات المراجعة المنهجية.



شكل (8): أساليب هندسة الأوامر للنماذج اللغوية الكبيرة

- الإجابة عن السؤال الرابع والذي نص على:

ما تحديات توظيف النماذج اللغوية الكبيرة في تعليم هندسة البرمجيات؟

تطرقت عدد من الدراسات إلى جانب تعليم هندسة البرمجيات وكيفية توظيف النماذج اللغوية الكبيرة في

تطوير المهارات الضرورية لمطوري البرمجيات، مثل مهارات التفكير الحاسوبي، ومهارة اتخاذ القرار، ومهارة تحويل البرمجيات من لغة لأخرى، كما تم توظيفها في توليد مشكلات بارسونز (Parsons Problems) (Reeves et al., 2023)، إلا أنه على الجانب الآخر رصدت الدراسات العديد من التحديات لهذه التقنيات الناشئة وعملية توظيفها في السياق التعليمي، منها ما هو مرتبط ببنية وطبيعة هذه التقنية، ومنها ما هو مرتبط برؤى ومبررات التطوير اللازم اتخاذها من قبل أصحاب القرار في المؤسسات التعليمية في ظل انتشار التطبيقات القائمة على النماذج اللغوية الكبيرة وارتفاع عدد مستخدميها (Lau & Guo)، ومن خلال المراجعة المنهجية في هذه الدراسة تم التوصل إلى عدد من التحديات يمكن تصنيفها إجمالاً ضمن ثلاثة محاور: 1) تحديات تقنية، 2) تحديات تربوية، 3) تحديات بحثية.

ارتبطت التحديات التقنية ببنية وآلية عمل النماذج اللغوية الكبيرة، وأنها قد تولد استجابة من الأوامر قد لا تتوافق دائماً مع هدف المبرمج، كما أنها تتأثر بضبط أوزان معلمات النماذج، وبالتالي يتوجب عليه تكرار تحسين الأمر للنموذج، و بناءً على ذلك، هدفت دراسة (Yen et al., 2023) إلى تقليل العبء المعرفي الذي يتعرض له المبرمجين عند محاولاتهم المستمرة في تعديل الأوامر باللغة الطبيعية للنماذج اللغوية الكبيرة في مهمة إنشاء الكود البرمجي، والتحقق من دقة استجابة النموذج وصحة الكود البرمجي، وذلك بتصميم نظام التوليد الهرمي للكود البرمجي، بهدف مساعدة المبرمجين على نمذجة أفكارهم في تحليل المهمة وتجزئتها إلى مهمات أصغر، وتصميم الأوامر بشكل هيكل هرمي لتوليد الكود البرمجي، بحيث يمثل كل أمر كتلة برمجية (Modular Block) يستطيع المبرمج تعديلها بسهولة في حال كان هناك خطأ في الكود المنشأ المقابل لهذا الأمر، وبتجربة النظام في دراسة مستخدم شملت (12) مبرمج من ذوي الخبرة في استخدام النماذج اللغوية الكبيرة لتوليد الكود البرمجي، ثبت أن النظام كان له دور كبير في تخفيف العبء على المبرمجين في عملية نمذجة أفكارهم إلى أوامر.

وفي ذات الإطار، أشارت دراسة كلاً من (Denny et al., MacNeil et al., 2023; Kiesler et al., 2023; Zelikman et al., 2023; Wang et al., 2023) إلى قضية التحيز (Bias) في النماذج اللغوية الكبيرة، وأن النماذج المخصصة لتوليد الكود البرمجي تم تدريبها على عدد كبير من الأكواد البرمجية والمكتوبة غالباً من قبل خبراء في البرمجة، وبالتالي قد تُنشئ هذه النماذج كود برمجي معقد وصعب الفهم أو التعديل من قبل المبتدئين في البرمجة، إضافة إلى مشكلة الهلوسة (Hallucination) في النماذج اللغوية الكبيرة والتي قد ينتج منها توليد أكواد خاطئة تؤثر على جودة البرمجيات.

وتربوياً، تم مناقشة آثار الاعتماد المفرط على النماذج اللغوية الكبيرة في توليد الأكواد البرمجية على عدد من المهارات الضرورية للمبرمجين، إضافة إلى سُبل تطوير أصول التدريس وأساليب التقويم لتعليم هندسة البرمجيات بما يتلاءم مع استخدام هذه النماذج، وفي هذا الجانب، أشارت دراسة (Lau & Guo, 2023) إلى ضرورة تخطيط مقررات هندسة البرمجيات في ضوء التطور الحالي في نماذج توليد الكود البرمجي، والاستفادة من دمجها في بيئات التطوير المتكاملة (IDEs) للبرمجيات بطرق مبتكرة، على سبيل المثال، تصميم المهام البرمجية المقاومة للذكاء الاصطناعي (AI-Proof Assignments) بحيث تكون أقل اعتماداً على نماذج الذكاء الاصطناعي التوليدي، و بما يضمن إعداد المبرمجين بالمعارف والمهارات التي لا يمكن استبدالها بواسطة نماذج الذكاء الاصطناعي التوليدي، والتي قد يكون أحد الأمثلة عليها إضافة السياق المحلي أو الثقافي على المهمة، أو التركيز على مهارة تقييم الكود البرمجي المُنشأ بواسطة نماذج الذكاء الاصطناعي، وهذا بالضرورة يستلزم أن يمتلك المبرمج المهارات والمعارف الأساسية بالبرمجة التي تجعله

قادراً على فهم وتقييم الكود البرمجي، إضافة إلى التركيز على التقييم القائم على العمليات والتي تتضمن التقويم التكويني للعمليات التفصيلية في كتابة الكود البرمجي وليس على المخرج النهائي.

وفي ذات السياق، أكدت دراسة (Denny et al., 2023) إلى أن إفراط المبرمجين في الاعتماد على نماذج الذكاء الاصطناعي التوليدي في كتابة الكود البرمجي يُعيق تطوير مهارات ما وراء المعرفة والتي تعتبر العنصر الجوهري لتعزيز مهارات التفكير الحاسوبي، وللحد من ذلك، اقترحت دراسة (Reeves et a., 2023) اعتماد وتطوير أساليب التقويم القائمة على مشكلات بارسونز (Parsons Problems) وذلك بعد اختبار نموذج (Codex) في حلها وعدم تمكنه من حل المشكلات المتقدمة بشكل صحيح.

وبحثياً، اتفقت العديد من الدراسات (Fu et al., 2023; Zhang et al., 2023; Wang et al., 2023; Denny et al., 2023; Kiesler et al., 2023; Prather et al., 2023) إلى أن توظيف النماذج اللغوية الكبيرة في تعليم هندسة البرمجيات، وتحديداً توليد الكود البرمجي، لا زال مجال بحثي بحاجة للمزيد من الدراسات التطبيقية، ودراسات الفاعلية، لتوجيه استخدام هذه النماذج بطرق فعالة، وعادلة، وأخلاقية، كما أن توظيفها في تعليم هندسة البرمجيات ينطوي على العديد من القيود والتي تتطلب على الباحثين أخذها بعين الاعتبار، على سبيل المثال، مدى قابلية الوصول والاستخدام لهذه النماذج، قيود مرتبطة بالخبرة بهندسة أوامر النماذج اللغوية الكبيرة وتقييم استجاباتها، وكيفية استخدامها من قبل المبتدئين والدعائم التربوية التي تمكنهم من فهم أداء النماذج اللغوية الكبيرة في توليد الكود البرمجي، وحيث أن هذه النماذج مصممة لتوليد الكود البرمجي وليس لتعليم بنية الكود البرمجي فهذا يشكل تحدي في كيفية تخصيصها للسياق التعليمي، ويتطلب خبرة لتوظيفها وفق الأسس و النماذج التربوية، بما يضمن تأهيل مطوري البرمجيات بمتطلبات وظائف المستقبل في ضوء التطور المتسارع لتقنيات الذكاء الاصطناعي التوليدي.

المناقشة

استناداً على ما تم التوصل إليه في نتائج المراجعة المنهجية، فقد تم توظيف النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات وذلك بتخفيف الجهود على المبرمجين في كتابة الكود البرمجي، أو توليد التعليقات التوضيحية لوظيفة الكود البرمجي، واختبار سلامته من الأخطاء، واكتشاف الثغرات الأمنية فيه وإصلاحها، وقد أظهرت أداءً جيداً في ذلك (Wang et al., 2023; Denny et al., 2023)، إلا أنه بالمقابل هناك عدد من التحديات التي ترتبط بخصائص استجابات النماذج اللغوية الكبيرة، كالعشوائية، أو الحالات النمطية التي يتم توليدها بناء على البيانات التي تم تدريب هذه النماذج عليها، أو الأخطاء، وهذا يتطلب توظيفها بحذر على وجه الخصوص مع المبرمجين المبدئين، إضافة إلى ضرورة تقييم دقة مخرجاتها (Fan et al., 2023)، وتجدر الإشارة إلى أن عملية تطوير البرمجيات لا تتطلب مجرد تلقي استجابة من نماذج الذكاء الاصطناعي على أوامر محددة، وإنما من الضروري فهم كيفية دمج هذه النماذج في مراحل وقرارات تصميم البرمجيات، لتحديد مستوى التفاعل المناسب بين مطوري البرمجيات ونماذج الذكاء الاصطناعي، فنجاح نماذج الذكاء الاصطناعي في حل مشكلة من مشاكل تصميم البرمجيات بدقة جيدة لا يعني بالضرورة دقته في حل المشكلات الحقيقة في بيئة تصميم البرمجيات واتخاذ القرارات الملائمة لها (Liu et al., 2023; Denny et al., 2023).

وبالرغم من أن النماذج اللغوية الكبيرة حققت مرونة عالية في الاستجابة لأوامر اللغة الطبيعية، إلا أن

تأثير هذه الأوامر على النماذج اللغوية الكبيرة غير مفهوم بشكل دقيق، وبالتالي هناك صعوبة نوعاً ما في تحديد معايير جودة الأوامر لتحقيق استجابة محددة، إضافة إلى صعوبة التنبؤ بنوعية استجابة النماذج اللغوية الكبيرة بناء على تصميم الأوامر فقط (Liu et al, 2023; Kiesler et al., 2023)، وفي حين أظهرت بعض الدراسات نتائج تُبرز الدور الكبير لهندسة الأوامر في مهام هندسة البرمجيات، مثل توليد الكود البرمجي، إلا أن نتائج دراسات أخرى أكدت أن هندسة الأوامر تعتمد بشكل كبير على الخبرة البشرية، وفهم ومراقبة آلية استجابة النماذج اللغوية الكبيرة، و هو ما يجعل عملية تقييم تصميم الأوامر مجال بحاجة للمزيد من الدراسة والتطبيق (Fu et al., 2023; Zhang et al., 2023; Wang et al., 2023; Kiesler et al., 2023; Prather et al., 2023).

وتجدر الإشارة إلى أن انتشار تطبيقات النماذج اللغوية الكبيرة في مجال هندسة البرمجيات، سيرفع مستوى الحاجة إلى المهارات العقلية والبرمجية المطلوبة لدى مهندسي البرمجيات، على سبيل المثال، مهمة تحديد المتطلبات، وتحليل النظام، وصياغة المواصفات الدقيقة للنظام والتحقق منه، جميعها مهام تتطلب خبرة بشرية و وعي بتحديات تطوير البرمجيات في البيئة الحقيقية، وهو ما قد يصعب أتمتة كلياً وتنفيذه بواسطة التطبيقات القائمة النماذج اللغوية الكبيرة (Alto, 2024)، إضافة إلى أن انتشار أدوات توليد الأكواد البرمجية مثل (ChatGPT)، (Copilot) يبرز الحاجة الكبيرة إلى أهمية إعداد المبرمجين بالمعارف والمهارات الضرورية والتي من جهة تمكنهم من فهم وتحليل وتقييم مخرجات هذه الأدوات، ومن جهة أخرى تجعلهم قادرين على كتابة الأكواد البرمجية المعقدة التي لا يمكن للغة الطبيعية أن تعالج تفاصيلها و تحفز النموذج على توليدها (Fan et al., 2023).

ومما سبق، تبرز الأدوار الجديدة للتعليم وإعداد مطوري البرمجيات، وحاجة المعلمين إلى التكيف مع سرعة انتشار تطبيقات توليد الأكواد البرمجية، وذلك بالاستفادة منها في تطوير أساليب التقويم، واعتبار المخرجات الخاطئة للنماذج اللغوية الكبيرة نقطة انطلاق لتعزيز فهم البرمجة، ومهارات التفكير الناقد، ومهارات التفكير الحاسوبي، إضافة إلى تسخير هذه التطبيقات كمساعد تعليمي يمكن تخصيصه حسب الحاجة، وذلك لتوضيح مفاهيم هندسة البرمجيات، أو إنشاء الموارد التعليمية والمهام التي تركز على العمليات بدلاً من المخرج النهائي، كما يمكن الاستفادة منها لزيادة إنتاجية مطوري البرمجيات وليس استبدالهم.

التوصيات:

تأسيساً على نتائج هذه المراجعة المنهجية، يمكن تقديم التوصيات التالية:

- تصميم التطبيقات القائمة على النماذج اللغوية الكبيرة والتي تستهدف أتمتة مهمة محددة من مهام هندسة البرمجيات.
- تعزيز القدرات الإبداعية البشرية في إنتاج محتوى الذكاء الاصطناعي التوليدي (AIGC) في مجال هندسة البرمجيات بإكساب مهندسي البرمجيات المهارات المتقدمة في هندسة الأوامر.
- إجراء المزيد من التجارب البحثية للتحقق من أداء النماذج اللغوية الكبيرة في أتمتة مهام هندسة البرمجيات.
- إجراء دراسات التحقق من فاعلية النماذج اللغوية الكبيرة في تنمية مهارات التفكير الحاسوبي ومهارات ما وراء المعرفة.

المقترحات

- اتخاذ جميع التدابير التي تضمن نشر الوعي الأخلاقي والاستخدام المسؤول للنماذج اللغوية الكبيرة في مجال هندسة البرمجيات.

- المساهمة في تطوير أصول تدريس، وأساليب تقويم مقرر هندسة البرمجيات في ضوء التطورات الحالية لتطبيقات النماذج اللغوية الكبيرة.
- مواجهة الفجوة الرقمية المتوقعة من تفاوت المعرفة والاستخدام لتطبيقات النماذج اللغوية الكبيرة في المجتمع التربوي بعقد البرامج التدريبية وورش العمل حول هذه التطبيقات وفرص توظيفها في التعليم.

❋❋❋❋

المراجع العربية:

الخليفة، هـ. س. (2023). *مقدمة في الذكاء الاصطناعي التوليدي*. https://www.researchgate.net/publication/371790205_mqdmt_fy_aldhka_alastnay_altwlydy

سدايا. (2023). *الذكاء الاصطناعي في التعليم*. https://sdaia.gov.sa/ar/MediaCenter/KnowledgeCenter/ResearchLibrary/GenAIE.pdf

المراجع الأجنبية:

Abedi, M., Alshybani, I., Shahadat, M. R. B., & Murillo, M. (2023). Beyond Traditional Teaching: The Potential of Large Language Models and Chatbots in Graduate Engineering Education. *Qeios*.

al-Khalīfah, H. S. (2023).*Muqaddimah fī al-dhakā' alāṣṭnā'y al-tawlīdī*. https://www.researchgate.net/publication/371790205_mqdmt_fy_aldhka_alastnay_altwlydy

Alto, V. (2024). *Building LLM Apps : Create Intelligent Apps and Agents with Large Language Models* (1st ed., pp. 13-14). Packt Publishing.

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023, March). Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 500-506).

Caelen, O., & Blete, M. (2023). *Developing Apps with GPT-4 and ChatGPT* .O'Reilly Media.

Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2023). Promptly: Using prompt problems to teach learners how to effectively utilize ai code generators. *arXiv preprint arXiv:2307.16364*.

Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., ... & Sarsa, S. (2023). Computing Education in the Era of Generative AI. *arXiv preprint arXiv:2306.02608*.

Duarte, F. (2023, November 30). *Number of ChatGPT Users (Dec 2023)*. Https://Explodingtopics.com/. https://explodingtopics.com/blog/chatgpt-users

Fan, A. X., Zhang, R. H., Paquette, L., & Zhang, R. (2023). Exploring the Potential of Large Language Models in Generating Code-Tracing Questions for Introductory Programming Courses. *arXiv preprint arXiv:2310.15317*.

Fan, G., Xie, X., Zheng, X., Liang, Y., & Di, P. (2023). Static Code Analysis in the AI Era: An In-depth Exploration of the Concept, Function, and Potential of Intelligent Code Analysis Agents. *arXiv preprint arXiv:2310.08837*.

Feng, F. L., Yen, R., You, Y., Fan, M., Zhao, J., & Lu, Z. (2023). CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming. *arXiv preprint arXiv:2310.09235*.

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022, February). The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference* (pp. 10-19).

Fu, L., Chai, H., Luo, S., Du, K., Zhang, W., Fan, L., ... & Yu, Y. (2023). CodeApex: A Bilingual Programming Evaluation Benchmark for Large Language Models. *arXiv preprint arXiv:2309.01940*.

Gore, D. V., Binoj, M., Borate, S., Devnani, R., & Gopale, S. (2023). Syntax Error Detection and Correction in Python Code using ML. *Grenze International Journal of Engineering & Technology (GIJET)*, *9*(2).

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., ... & Wang, H. (2023). Large language models for software engineering: A systematic literature review. *arXiv preprint arXiv:2308.10620*.

Huang, D., Nan, Z., Hu, X., Jin, P., Peng, S., Wen, Y., ... & Chen, Y. (2023). ANPL: Compiling Natural Programs with Interactive Decomposition. *arXiv preprint arXiv:2305.18498*.

Ji, Z., Ma, P., Li, Z., & Wang, S. (2023). Benchmarking and Explaining Large Language Model-based Code Generation: A Causality-

comparison of coverage via citations. *Scientometrics*, *126*(1), 871-906.

Martínez, P. A., Bernabé, G., & García, J. M. (2023). Code Detection for Hardware Acceleration Using Large Language Models. *arXiv preprint arXiv:2307.10348*.

Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., & Myers, B. (2023). In-IDE Generation-based Information Support with a Large Language Model. *arXiv preprint arXiv:2307.08177*.

Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., ... & Abrahamsson, P. (2023). Generative Artificial Intelligence for Software Engineering--A Research Agenda. *arXiv preprint arXiv:2310.18648*.

Ozdemir, S. (2023). *Quick Start Guide to Large Language Models: Strategies and Best Practices for Using ChatGPT and Other LLMs*. Addison-Wesley Professional. O'Reilly Media.

Ozkaya, I. (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software*, *40*(3), 4-8.

Phoenix, J., & Taylor, M. (2024). *Prompt Engineering for Generative AI Write the first review* (1st ed., pp. 13-14). O'Reilly Media.

Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction*, *31*(1557-7325). https://doi.org/10.1145/3617367

Reeves, B., Sarsa, S., Prather, J., Denny, P., Becker, B. A., Hellas, A., ... & Leinonen, J. (2023, June). Evaluating the performance of code generation models for solving Parsons problems with small prompt variations. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 299-305).

Roest, L. (2023). *Automated Next-Step Hint Generation For Introductory Programming Using Large Language Models* [Master's thesis, Utrecht University].

Sarkis-Onofre, R., Catalá-López, F., Aromataris, E., & Lockwood, C. (2021). How to properly use the PRISMA Statement. *Systematic Reviews*, *10*(1), 1-3.

Savelka, J., Agarwal, A., An, M., Bogart, C., & Sakr, M. (2023). Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Centric Approach. *arXiv preprint arXiv:2310.06680*.

Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). How Novices Use LLM-Based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment. *arXiv preprint arXiv:2309.14049*.

Kiesler, N., Lohr, D., & Keuning, H. (2023). Exploring the potential of large language models to generate formative programming feedback. *arXiv preprint arXiv:2309.00029*.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kotti, Z., Galanopoulou, R., & Spinellis, D. (2023). Machine learning for software engineering: A tertiary study. *ACM Computing Surveys*, *55*(12), 1-39.

Lau, S., & Guo, P. (2023, August). From" Ban it till we understand it" to" Resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1* (pp. 106-121).

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., ... & Zeng, A. (2023, May). Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9493-9500). IEEE.

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., ... & Ge, B. (2023). Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, 100017.

Lorenz, P., K. Perset and J. Berryhill (2023), "Initial policy considerations for generative artificial intelligence", *OECD Artificial Intelligence Papers*, No. 1, OECD Publishing, Paris, https://doi.org/10.1787/fae2d1e6-en.

MacNeil, S., Tran, A., Hellas, A., Kim, J., Sarsa, S., Denny, P., ... & Leinonen, J. (2023, March). Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 931-937).

Martín-Martín, A., Thelwall, M., Orduna-Malea, E., & Delgado López-Cózar, E. (2021). Google Scholar, Microsoft Academic, Scopus, Dimensions, Web of Science, and OpenCitations' COCI: a multidisciplinary

Information Processing Systems, *35*, 24824-24837.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yen, R., Zhu, J., Suh, S., Xia, H., & Zhao, J. (2023). CoLadder: Supporting Programmers with Hierarchical Code Generation in Multi-Level Abstraction. *arXiv preprint arXiv:2310.08699*.

Yilmaz, R., & Yilmaz, F. G. K. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Computers in Human Behavior: Artificial Humans*, *1*(2), 100005.

Zelikman, E., Huang, Q., Poesia, G., Goodman, N., & Haber, N. (2023, November). Parsel: Algorithmic Reasoning with Language Models by Composing Decompositions. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhang, Q., Zhang, T., Zhai, J., Fang, C., Yu, B., Sun, W., & Chen, Z. (2023). A Critical Review of Large Language Model on Software Engineering: An Example from ChatGPT and Automated Program Repair. *arXiv [Cs.SE]*. Retrieved from http://arxiv.org/abs/2310.08879

Programming Courses. *arXiv preprint arXiv:2306.10073*.

Scoccia, G. L. (2023, September). Exploring Early Adopters' Perceptions of ChatGPT as a Code Generation Tool. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)* (pp. 88-93). IEEE.

Sdāyā. (2023). *al-Dhakā' alāṣṭnā'y fī al-Ta'līm*. https://sdaia.gov.sa/ar/MediaCenter/KnowledgeCenter/ResearchLibrary/GenAIE.pdf

Sheese, B., Liffiton, M., Savelka, J., & Denny, P. (2023). Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant. *arXiv preprint arXiv:2310.16984*.

Shin, J., Tang, C., Mohati, T., Nayebi, M., Wang, S., & Hemmati, H. (2023). Prompt Engineering or Fine Tuning: An Empirical Assessment of Large Language Models in Automated Software Engineering Tasks. *arXiv preprint arXiv:2310.10508*.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., ... & Garg, A. (2023). ProgPrompt: Program generation for situated robot task planning using large language models. *Autonomous Robots*, 1-14.

Spasić, A. J., & Janković, D. S. (2023, June). Using ChatGPT standard prompt engineering techniques in lesson preparation: role, instructions and seed-word prompts. In *2023 58th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)* (pp. 47-50). IEEE.

Tsui, F., Karam, O., & Bernal, B. (2022). *Essentials of software engineering*. Jones & Bartlett Learning.

Tu, X., Zou, J., Su, W. J., & Zhang, L. (2023). What Should Data Science Education Do with Large Language Models?. *arXiv preprint arXiv:2307.02792*.

UNESCO. (2023). *Guidance for generative AI in education and research* (1st ed.). https://unesdoc.unesco.org/ark:/48223/pf0000386693

Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2023). Software testing with large language model: Survey, landscape, and vision. *arXiv preprint arXiv:2307.07221*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural*