BERT: попереднє навчання глибоких двонапрямних трансформерів для розуміння мови

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Анотація

Ми представляємо нову модель мовних подань під назвою **BERT** — **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (двонапрямні подання кодувальника на основі трансформерів). На відміну від нещодавніх моделей мовних подань (Peters et al., 2018a; Radford et al., 2018), BERT призначено для попереднього навчання (pre-training) глибоких двонапрямних подань на нерозміченому тексті зі спільним урахуванням лівого й правого контексту на всіх шарах. Завдяки цьому попередньо навчену BERT можна донавчити (fine-tuning), додавши лише один вихідний шар, і отримати найкращі на сьогодні моделі для широкого кола завдань, зокрема відповідей на запитання й мовного висновування, без істотної перебудови архітектури під конкретне завдання.

BERT концептуально проста й емпірично потужна. Вона встановлює нові найкращі результати в одинадцяти завданнях оброблення природної мови: оцінка GLUE сягає 80.5% (абсолютне поліпшення на 7.7 відсоткового пункту), точність MultiNLI — 86.7% (поліпшення на 4.6%), тестова F1 для відповідей на запитання SQuAD v1.1 — 93.2 (поліпшення на 1.5 пункту), а тестова F1 SQuAD v2.0 — 83.1 (поліпшення на 5.1 пункту).

1 Вступ

Показано, що попереднє навчання мовної моделі ефективно поліпшує розв'язання багатьох завдань оброблення природної мови (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). Серед них — завдання рівня речення, зокрема висновування природною мовою (Bowman et al., 2015; Williams et al., 2018) і перефразування (Dolan and Brockett, 2005), де зв'язки між реченнями передбачають на основі їх цілісного аналізу, а також завдання рівня токена, зокрема розпізнавання іменованих сутностей і відповіді на запитання, де модель має формувати докладний вихід на рівні токенів (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

Існують дві стратегії застосування попередньо навчених мовних подань до цільових завдань: *на основі ознак* і *донавчання*. Підхід на основі ознак, як-от ELMo (Peters et al., 2018a), використовує спеціалізовані архітектури, до яких попередньо навчені подання входять як додаткові ознаки. Підхід донавчання, як-от Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), додає мінімум параметрів під конкретне завдання й навчається на цільових завданнях простим донавчанням *усіх* попередньо навчених параметрів. Обидва підходи мають однакову цільову функцію попереднього навчання: вони використовують однонапрямні мовні моделі для засвоєння загальних мовних подань.

Ми стверджуємо, що сучасні методи обмежують можливості попередньо навчених подань, особливо в підходах донавчання. Головне обмеження полягає в однонапрямності стандартних мовних моделей, що звужує вибір архітектур для попереднього навчання. Наприклад, в OpenAI GPT автори застосовують архітектуру зліва направо, де кожен токен у шарах самоуваги (self-attention) трансформера може звертатися лише до попередніх токенів (Vaswani et al., 2017). Такі обмеження неоптимальні для завдань рівня речення й можуть дуже зашкодити донавчанню для завдань рівня токена, зокрема відповідей на запитання, де вкрай важливо враховувати контекст з обох боків.

У цій статті ми вдосконалюємо підходи донавчання, пропонуючи BERT: **B**idirectional

Encoder **R**epresentations from **T**ransformers. BERT послаблює згадане обмеження однонапрямності завдяки цілі попереднього навчання «маскована мовна модель» (masked language model, MLM), натхненній завданням заповнення пропусків Cloze (Taylor, 1953). Маскована мовна модель випадково маскує деякі вхідні токени й має передбачити початковий словниковий ідентифікатор замаскованого слова лише за його контекстом. На відміну від попереднього навчання мовної моделі зліва направо, ціль MLM дає змогу поєднати в поданні лівий і правий контексти, що дозволяє попередньо навчати глибокий двонапрямний трансформер (bidirectional Transformer). Окрім маскованої мовної моделі, ми застосовуємо завдання «передбачення наступного речення» (next sentence prediction), що спільно навчає подання пар текстів. Внесок нашої роботи такий:

- Ми демонструємо важливість двонапрямного попереднього навчання мовних подань. На відміну від Radford et al. (2018), де попередньо навчають однонапрямні мовні моделі, BERT використовує масковані мовні моделі для отримання глибоких двонапрямних подань. Це відрізняється й від Peters et al. (2018a), де неглибоко конкатенують незалежно навчені моделі зліва направо та справа наліво.

- Ми показуємо, що попередньо навчені подання зменшують потребу в численних складно спроєктованих архітектурах під конкретні завдання. BERT — перша модель подань на основі донавчання, яка досягає найкращої якості на великому наборі завдань рівня речення *та* токена, перевершуючи багато спеціалізованих архітектур.

- BERT встановлює нові найкращі результати в одинадцяти завданнях NLP. Код і попередньо навчені моделі доступні за адресою `https://github.com/google-research/bert`.

2 Пов’язані роботи

Попереднє навчання загальних мовних подань має довгу історію; у цьому розділі ми коротко оглядаємо найпоширеніші підходи.

2.1 Некеровані підходи на основі ознак

Навчання широко застосовних подань слів активно досліджують протягом десятиліть, зокрема ненейронними (Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006) і нейронними (Mikolov et al., 2013; Pennington et al., 2014) методами. Попередньо навчені векторні вкладення слів є невіддільною частиною сучасних систем NLP і суттєво перевершують вкладення, навчені з нуля (Turian et al., 2010). Для попереднього навчання векторів слів застосовували цілі мовного моделювання зліва направо (Mnih and Hinton, 2009), а також розрізнення правильних і неправильних слів у лівому та правому контексті (Mikolov et al., 2013).

Ці підходи узагальнили на більші одиниці, зокрема вкладення речень (Kiros et al., 2015; Logeswaran and Lee, 2018) і абзаців (Le and Mikolov, 2014). Для навчання подань речень попередні роботи ранжували кандидати наступних речень (Jernite et al., 2017; Logeswaran and Lee, 2018), генерували слова наступного речення зліва направо за поданням попереднього (Kiros et al., 2015) або використовували цілі, похідні від знешумлювальних автокодувальників (Hill et al., 2016).

ELMo та її попередник (Peters et al., 2017, 2018a) узагальнюють традиційні дослідження вкладень слів в іншому напрямі. Вони видобувають *контекстно залежні* ознаки з мовних моделей зліва направо й справа наліво. Контекстне подання (contextual representation) кожного токена — це конкатенація подань обох напрямів. Інтегруючи контекстні вкладення слів у наявні спеціалізовані архітектури, ELMo поліпшує найкращі результати на кількох провідних тестах NLP (Peters et al., 2018a), зокрема відповідей на запитання (Rajpurkar et al., 2016), аналізу тональності (Socher et al., 2013) і розпізнавання іменованих сутностей (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016) запропонували навчати контекстні подання через передбачення одного слова за лівим і правим контекстами за допомогою LSTM. Як і ELMo, їхня модель базується на ознаках і не є глибоко двонапрямною. Fedus et al. (2018) показують, що заповнення пропусків може підвищити стійкість моделей генерування тексту.

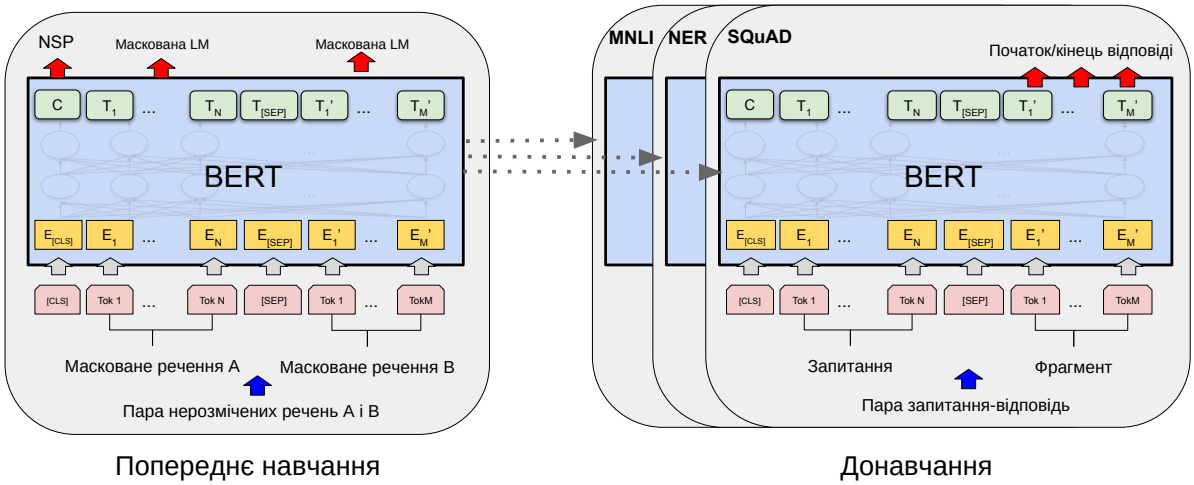


Рисунок 1: Загальні процедури попереднього навчання й донавчання BERT. Окрім вихідних шарів, на обох етапах використовують однакові архітектури. Ті самі попередньо навчені параметри ініціалізують моделі для різних цільових завдань. Під час донавчання налаштовують усі параметри. `[CLS]` — спеціальний символ на початку кожного вхідного прикладу, а `[SEP]` — спеціальний токен-роздільник (наприклад, між запитаннями й відповідями).

2.2 Некеровані підходи донавчання

Як і підходи на основі ознак, перші роботи цього напряму попередньо навчали лише параметри вкладень слів на нерозміченому тексті (Collobert and Weston, 2008).

Нещодавно кодувальники речень або документів, які створюють контекстні подання токенів, почали попередньо навчати на нерозміченому тексті, а потім донавчати для керованого цільового завдання (Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018). Перевага таких підходів — мала кількість параметрів, які треба навчати з нуля. Принаймні частково завдяки цьому OpenAI GPT (Radford et al., 2018) досягла тодішніх найкращих результатів у багатьох завданнях рівня речення тесту GLUE (Wang et al., 2018a). Для попереднього навчання таких моделей застосовували мовне моделювання зліва направо й цілі автокодувальника (Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015).

2.3 Перенесення навчання з розмічених даних

Інші роботи показали ефективне перенесення з керованих завдань із великими наборами даних, зокрема висновування природною мовою (Conneau et al., 2017) і машинного перекладу (McCann et al., 2017). Дослідження комп'ютерного зору також продемонстрували важливість перенесення навчання (transfer learning) з великих попередньо навчених моделей: ефективний рецепт — донавчати моделі, попередньо навчені на ImageNet (Deng et al., 2009; Yosinski et al., 2014).

3 BERT

У цьому розділі ми представляємо BERT і подробиці її реалізації. Наш підхід має два етапи: *попереднє навчання* та *донавчання*. Під час попереднього навчання модель навчається на нерозмічених даних для різних навчальних завдань. Для донавчання BERT спочатку ініціалізують попередньо навченими параметрами, а потім усі параметри налаштовують на розмічених даних цільових завдань. Кожне цільове завдання має окремі донавчені моделі, хоча їх ініціалізують тими самими параметрами. Приклад відповідей на запитання на рисунку 1 слугуватиме наскрізним прикладом цього розділу.

Відмітна риса BERT — єдина архітектура для різних завдань. Відмінності між попередньо навченою архітектурою й остаточною архітектурою цільового завдання мінімальні.

Архітектура моделі Архітектура BERT — багатошаровий двонапрямний кодувальник Transformer, що спирається на початкову реалізацію з Vaswani et al. (2017), опубліковану в бібліотеці `tensor2tensor`.[1] Оскільки трансформери стали поширеними, а наша реалізація

[1] https://github.com/tensorflow/tensor2tensor

майже ідентична початковій, ми опускаємо вичерпний опис архітектури й відсилаємо читача до Vaswani et al. (2017) та чудових посібників, зокрема «The Annotated Transformer».[2]

У цій роботі кількість шарів (тобто блоків Transformer) позначаємо L, розмір прихованого подання — H, кількість голів самоуваги — A.[3] Переважно наводимо результати для двох розмірів моделей: **BERT$_{\text{BASE}}$** (L=12, H=768, A=12, усього параметрів=110M) і **BERT$_{\text{LARGE}}$** (L=24, H=1024, A=16, усього параметрів=340M).

Для порівняння розмір BERT$_{\text{BASE}}$ обрано таким самим, як в OpenAI GPT. Проте принципово, що трансформер BERT використовує двонапрямну самоувагу, а трансформер GPT — обмежену: кожен токен звертається лише до контексту ліворуч.[4]

Подання входу й виходу Щоб BERT могла розв'язувати різні цільові завдання, наше вхідне подання однозначно представляє як окреме речення, так і пару речень (наприклад, ⟨ запитання, відповідь ⟩) в одній послідовності токенів. У цій роботі «речення» може бути довільним неперервним фрагментом тексту, а не справжнім лінгвістичним реченням. «Послідовність» — це вхідна послідовність токенів BERT, яка може містити одне речення або два, упаковані разом.

Ми використовуємо вкладення WordPiece (Wu et al., 2016) зі словником із 30,000 токенів. Перший токен кожної послідовності — завжди спеціальний класифікаційний токен (`[CLS]`). Його кінцевий прихований стан використовується як сукупне подання послідовності для класифікації. Пари речень упаковують в одну послідовність. Речення розрізняють двома способами. По-перше, їх розділяють спеціальним токеном (`[SEP]`). По-друге, до кожного токена додають навчене вкладення, що вказує його належність до речення `A` чи `B`. Як на рисунку 1, вхідне вкладення позначаємо E, кінцевий прихований вектор спеціального токена `[CLS]` — $C \in \mathbb{R}^H$, а кінцевий прихований вектор i^{th} вхідного токена — $T_i \in \mathbb{R}^H$.

Вхідне подання токена утворюють додаванням відповідних вкладень токена, сегмента й позиції. Цю побудову візуалізовано на рисунку 2.

3.1 Попереднє навчання BERT

На відміну від Peters et al. (2018a) і Radford et al. (2018), ми не застосовуємо традиційні мовні моделі зліва направо чи справа наліво для попереднього навчання BERT. Натомість використовуємо два некеровані завдання, описані далі. Цей етап показано ліворуч на рисунку 1.

Завдання #1: маскована LM Інтуїтивно, глибока двонапрямна модель має бути строго потужнішою за модель зліва направо чи неглибоку конкатенацію моделей обох напрямів. На жаль, стандартні умовні мовні моделі можна навчати лише зліва направо *або* справа наліво: двонапрямне обумовлення дозволило б кожному слову непрямо «бачити себе», і в багатошаровому контексті модель могла б тривіально передбачати цільове слово.

Щоб навчити глибоке двонапрямне подання, ми просто випадково маскуємо певну частку вхідних токенів і передбачаємо замасковані токени. Цю процедуру називаємо «маскованою LM» (MLM), хоча в літературі її часто називають завданням *Cloze* (Taylor, 1953). Кінцеві приховані вектори маскованих токенів подають на вихідну softmax над словником, як у стандартній LM. В усіх експериментах випадково маскуємо 15% токенів WordPiece кожної послідовності. На відміну від знешумлювальних автокодувальників (Vincent et al., 2008), ми передбачаємо лише замасковані слова, а не відновлюємо весь вхід.

Хоча це дає змогу отримати двонапрямну попередньо навчену модель, виникає невідповідність між попереднім навчанням і донавчанням, адже токен `[MASK]` не трапляється під час донавчання. Щоб її послабити, ми не завжди замінюємо «замасковані» слова власне токеном `[MASK]`. Генератор навчальних даних випадково обирає для передбачення 15% позицій токенів. Якщо обрано i-й токен, i-й токен замінюють (1) на `[MASK]` у 80% випадків, (2) на випадковий токен у 10%, (3) залишають i-й токен незмінним у 10%. Далі T_i використовують для передбачення початкового токена з втратою перехресної ентропії. Варіанти процедури порів-

[2] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[3] В усіх випадках розмір прямого поширення/фільтра становить $4H$, тобто 3072 для $H = 768$ і 4096 для $H = 1024$.

[4] У літературі двонапрямний трансформер часто називають «кодувальником Transformer», а версію лише з лівим контекстом — «декодувальником Transformer», оскільки вона придатна для генерування тексту.

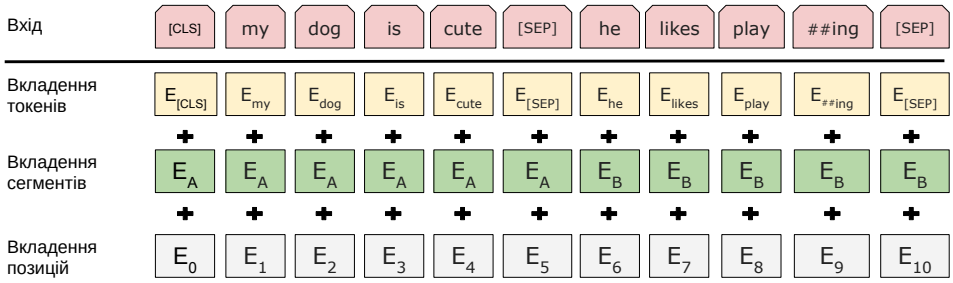


Рисунок 2: Вхідне подання BERT. Вхідні вкладення — сума вкладень токенів, сегментів і позицій.

нюємо в додатку C.2.

Завдання #2: передбачення наступного речення (NSP) Багато важливих цільових завдань, як-от відповіді на запитання (QA) і висновування природною мовою (NLI), ґрунтуються на розумінні *зв'язку* між двома реченнями, чого мовне моделювання безпосередньо не охоплює. Для навчання такого розуміння попередньо навчаємо модель бінарному завданню *передбачення наступного речення*, яке легко утворити з будь-якого одномовного корпусу. Обираючи речення `A` і `B` для прикладу, у 50% випадків беремо за `B` справжнє наступне речення після `A` (мітка `IsNext`), а в інших 50% — випадкове речення корпусу (мітка `NotNext`). Як показано на рисунку 1, для NSP використовують C.[5] Попри простоту, у розділі 5.1 показуємо значну користь такого попереднього навчання і для QA, і для NLI.[6] Завдання NSP близьке до цілей навчання подань у Jernite et al. (2017) і Logeswaran and Lee (2018). Проте в попередніх роботах до цільових завдань переносять лише вкладення речень, тоді як BERT переносить усі параметри для ініціалізації моделі кінцевого завдання. **Дані попереднього навчання**

Процедура загалом відповідає наявним роботам про попереднє навчання мовних моделей. Використовуємо корпуси BooksCorpus (800M слів) (Zhu et al., 2015) і англійську Вікіпедію (2,500M слів). З Вікіпедії видобуваємо лише текстові фрагменти, ігноруючи списки, таблиці й заголовки. Для отримання довгих неперервних послідовностей критично потрібен корпус рівня документів, а не перемішаних речень, як-от Billion Word Benchmark (Chelba et al., 2013).

[5]Остаточна модель досягає точності NSP 97%-98%.

[6]Без донавчання вектор C не є змістовним поданням речення, оскільки його навчали для NSP.

3.2 Донавчання BERT

Донавчання просте: механізм самоуваги Transformer дає змогу BERT моделювати численні цільові завдання, для одного тексту або пар текстів, замінивши відповідні входи й виходи. Для пар текстів часто спочатку незалежно кодують кожен текст, а потім застосовують двонапрямну перехресну увагу, як у Parikh et al. (2016); Seo et al. (2017). BERT натомість об'єднує ці етапи механізмом самоуваги: кодування конкатенованої пари фактично містить *двонапрямну* перехресну увагу між реченнями.

Для кожного завдання ми просто під'єднуємо спеціалізовані входи й виходи до BERT та наскрізно донавчаємо всі параметри. На вході речення `A` і `B` з попереднього навчання відповідають (1) парам речень у перефразуванні, (2) парам гіпотеза-передумова в завданні логічного слідування, (3) парам запитання-фрагмент у відповідях на запитання та (4) виродженій парі текст-$\varnothing$ в класифікації тексту чи розмічуванні послідовності. На виході подання токенів подають у вихідний шар для завдань рівня токена, зокрема розмічування чи відповідей на запитання, а подання `[CLS]` — у шар класифікації, зокрема логічного слідування чи тональності.

Порівняно з попереднім навчанням донавчання відносно дешеве. Усі результати статті можна відтворити щонайбільше за 1 годину на одному Cloud TPU або за кілька годин на GPU, починаючи з тієї самої попередньо навченої моделі.[7] Особливості кожного завдання описано у відповідних підрозділах розділу 4. Додаткові подробиці наведено в додатку A.5.

[7]Наприклад, модель BERT для SQuAD можна навчити приблизно за 30 хвилин на одному Cloud TPU й отримати F1 91.0% на валідаційній вибірці.

[8]Див. (10) на `https://gluebenchmark.com/faq`.

Система	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Середнє** -
SOTA до OpenAI	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Таблиця 1: Тестові результати GLUE за оцінкою сервера (https://gluebenchmark.com/leaderboard). Число під назвою завдання — кількість навчальних прикладів. Стовпець «Середнє» дещо відрізняється від офіційної оцінки GLUE, оскільки виключено проблемний набір WNLI.[8] BERT і OpenAI GPT використовують одну модель для одного завдання. Для QQP і MRPC подано F1, для STS-B — кореляцію Спірмена, для решти завдань — точність. Системи з BERT як компонентом виключено.

4 Експерименти

У цьому розділі подаємо результати донавчання BERT для 11 завдань NLP.

4.1 GLUE

Тест General Language Understanding Evaluation (GLUE) (Wang et al., 2018a) — набір різноманітних завдань розуміння природної мови. Докладний опис його наборів даних наведено в додатку B.1.

Для донавчання на GLUE представляємо вхідну послідовність (одне речення чи пару) як у розділі 3 і використовуємо кінцевий прихований вектор $C \in \mathbb{R}^H$ першого вхідного токена ([CLS]) як сукупне подання. Єдині нові параметри донавчання — ваги класифікаційного шару $W \in \mathbb{R}^{K \times H}$, де K — кількість міток. Обчислюємо стандартну класифікаційну втрату за C і W, тобто $\log(\text{softmax}(CW^T))$.

Для всіх завдань GLUE використовуємо пакети розміру 32 й донавчаємо протягом 3 епох. Для кожного завдання обираємо найкращий темп навчання (серед 5e-5, 4e-5, 3e-5 і 2e-5) на валідаційній вибірці. Крім того, донавчання $\text{BERT}_{\text{LARGE}}$ іноді нестабільне на малих наборах даних, тому робимо кілька випадкових перезапусків і обираємо найкращу модель за валідацією. За перезапусків беремо ту саму попередньо навчену контрольну точку, але інакше перемішуємо дані й ініціалізуємо класифікатор.[9]

Результати наведено в таблиці 1. І $\text{BERT}_{\text{BASE}}$, і $\text{BERT}_{\text{LARGE}}$ суттєво перевершують усі системи в усіх завданнях, поліпшуючи попередню найкращу середню точність відповідно на 4.5% і 7.0%. Зазначимо, що архітектури $\text{BERT}_{\text{BASE}}$ і OpenAI GPT майже ідентичні, окрім маскування уваги. На найбільшому й найчастіше звітованому завданні GLUE, MNLI, BERT дає абсолютне поліпшення точності на 4.6%. В офіційному рейтингу GLUE[10] $\text{BERT}_{\text{LARGE}}$ отримує 80.5 проти 72.8 у OpenAI GPT на момент написання.

$\text{BERT}_{\text{LARGE}}$ значно перевершує $\text{BERT}_{\text{BASE}}$ в усіх завданнях, особливо з дуже малими навчальними вибірками. Вплив розміру моделі докладніше вивчаємо в розділі 5.2.

4.2 SQuAD v1.1

Stanford Question Answering Dataset (SQuAD v1.1) містить 100k пар запитання/відповідь, зібраних краудсорсингом (Rajpurkar et al., 2016). За запитанням і фрагментом Вікіпедії, що містить відповідь, потрібно передбачити текстовий відрізок відповіді у фрагменті.

Як показано на рисунку 1, вхідне запитання й фрагмент представляємо однією упакованою послідовністю: запитання використовує вкладення A, а фрагмент — B. Під час донавчання додаємо лише вектор початку $S \in \mathbb{R}^H$ і вектор кінця $E \in \mathbb{R}^H$. Імовірність, що слово i починає відповідь, обчислюємо скалярним добутком T_i і S із подальшою softmax над усіма словами абзацу: $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$. Аналогічна формула дає кінець відповіді. Оцінка кандидата від позиції i до j дорівнює $S \cdot T_i + E \cdot T_j$; передбаченням є відрізок із найбільшою оцінкою за умови $j \geq i$. Ціль навчання — сума логарифмів правдоподібності правильних позицій початку й кінця. Донавчаємо протягом 3 епох із темпом 5e-5 та пакетами розміру 32.

[9] GLUE не поширює тестових міток; ми подали лише по одному результату $\text{BERT}_{\text{BASE}}$ і $\text{BERT}_{\text{LARGE}}$ на сервер оцінювання GLUE.

[10] https://gluebenchmark.com/leaderboard

Система	Валід. EM	Валід. F1	Тест EM	Тест F1
Лідери рейтингу (10 грудня 2018)				
Людина	-	-	82.3	91.2
#1 Ансамбль - nlnet	-	-	86.0	91.7
#2 Ансамбль - QANet	-	-	84.5	90.5
Опубліковані				
BiDAF+ELMo (Окрема)	-	85.6	-	85.8
R.M. Reader (Ансамбль)	81.2	87.9	82.3	88.5
Наші				
$\text{BERT}_{\text{BASE}}$ (Окрема)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Окрема)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ансамбль)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Окр.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Анс.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Таблиця 2: Результати SQuAD 1.1. Ансамбль BERT містить 7 систем із різними контрольними точками попереднього навчання та початковими станами генератора випадкових чисел для донавчання.

Система	Валід. EM	Валід. F1	Тест EM	Тест F1
Лідери рейтингу (10 грудня 2018)				
Людина	86.3	89.0	86.9	89.5
#1 Окрема - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Окрема - nlnet	-	-	74.2	77.1
Опубліковані				
unet (Ансамбль)	-	-	71.4	74.9
SLQA+ (Окрема)	-		71.4	74.4
Наші				
$\text{BERT}_{\text{LARGE}}$ (Окрема)	78.7	81.9	80.0	83.1

Таблиця 3: Результати SQuAD 2.0. Системи з BERT як компонентом виключено.

Таблиця 2 показує провідні записи рейтингу та найкращі опубліковані системи (Seo et al., 2017; Clark and Gardner, 2018; Peters et al., 2018a; Hu et al., 2018). Найкращі системи рейтингу SQuAD не мають актуальних публічних описів[11] і можуть навчатися на будь-яких відкритих даних. Тому застосовуємо помірне розширення даних: спочатку донавчаємо систему на TriviaQA (Joshi et al., 2017), а потім — на SQuAD.

Наша найкраща система перевершує лідера рейтингу на +1.5 F1 в ансамблі та +1.3 F1 як окрема модель. Фактично одна модель BERT перевершує найкращий ансамбль за F1. Без донавчання на TriviaQA втрачаємо лише 0.1-0.4 F1 і все одно значно випереджаємо всі наявні системи.[12]

4.3 SQuAD v2.0

Завдання SQuAD 2.0 розширює SQuAD 1.1: у наданому абзаці може не бути короткої відповіді, що робить постановку реалістичнішою.

Ми просто розширюємо BERT для SQuAD v1.1. Запитання без відповіді трактуємо як такі, де відрізок відповіді починається й закінчується на токені `[CLS]`. Простір імовірностей позицій початку й кінця розширюємо позицією `[CLS]`. Під час передбачення порівнюємо оцінку відсутності відповіді $s_{\texttt{null}} = S{\cdot}C + E{\cdot}C$ з оцінкою найкращого непорожнього відрізка $\hat{s_{i,j}} = \max_{j \geq i} S{\cdot}T_i + E{\cdot}T_j$. Передбачаємо непорожню відповідь, коли $\hat{s_{i,j}} > s_{\texttt{null}} + \tau$, де поріг τ обираємо на валідаційній вибірці для максимізації F1. Для цієї моделі не використовували TriviaQA. Донавчали протягом 2 епох із темпом 5e-5 та пакетами розміру 48.

Порівняння з попередніми лідерами рейтингу й найкращими опублікованими роботами (Sun et al., 2018; Wang et al., 2018b) подано в таблиці 3; системи з BERT як компонентом виключено. Отримано поліпшення +5.1 F1 відносно попередньої найкращої системи.

4.4 SWAG

Набір Situations With Adversarial Generations (SWAG) містить 113k прикладів доповнення пар речень, що перевіряють висновування на основі здорового глузду (Zellers et al., 2018). За заданим реченням потрібно обрати найправдоподібніше продовження з чотирьох варіантів.

Донавчаючись на SWAG, будуємо чотири вхідні послідовності, кожна з яких є конкатенацією заданого речення (A) й можливого продовження (B). Єдиний доданий спеціалізований параметр — вектор, скалярний добуток якого з поданням C токена `[CLS]` дає оцінку кожного варіанта; оцінки нормалізують шаром softmax.

Донавчаємо модель протягом 3 епох із темпом 2e-5 і пакетами розміру 16. Результати наведено в таблиці 4. $\text{BERT}_{\text{LARGE}}$ перевершує базову авторську систему ESIM+ELMo на +27.1%, а OpenAI GPT — на 8.3%.

[11] QANet описано в Yu et al. (2018), але після публікації систему суттєво вдосконалили.

[12] Ми використали абзаци TriviaQA-Wiki з перших 400 токенів документів, які містять принаймні одну з поданих можливих відповідей.

Система	Валід.	Тест
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Людина (експерт)†	-	85.0
Людина (5 анотацій)†	-	88.0

Таблиця 4: Точність SWAG на валідації й тесті. †Якість людини виміряно на 100 прикладах, як зазначено в статті про SWAG.

Завдання	Валідація MNLI-m (точн.)	QNLI (точн.)	MRPC (точн.)	SST-2 (точн.)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
Без NSP	83.9	84.9	86.5	92.6	87.9
LTR & без NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Таблиця 5: Абляція завдань попереднього навчання з архітектурою $\text{BERT}_{\text{BASE}}$. «Без NSP» навчається без передбачення наступного речення. «LTR & без NSP» — як LM зліва направо без NSP, подібно до OpenAI GPT. «+ BiLSTM» додає випадково ініціалізовану BiLSTM над «LTR + без NSP» під час донавчання.

5 Абляційні дослідження

У цьому розділі вилучаємо окремі компоненти BERT, щоб краще зрозуміти їхню відносну важливість. Додаткові абляційні дослідження подано в додатку C.

5.1 Вплив завдань попереднього навчання

Важливість глибокої двонапрямності BERT демонструємо, оцінюючи дві цілі попереднього навчання з тими самими даними, схемою донавчання й гіперпараметрами, що й $\text{BERT}_{\text{BASE}}$:

Без NSP: двонапрямна модель, навчена як «маскована LM» (MLM), але без «передбачення наступного речення» (NSP).
LTR & без NSP: модель лише з лівим контекстом, навчена як стандартна LM зліва направо (LTR), а не MLM. Обмеження лівим контекстом зберігаємо й під час донавчання: його зняття створювало невідповідність між етапами, яка погіршувала цільові результати. Модель також попередньо навчали без NSP. Вона безпосередньо порівнянна з OpenAI GPT, але використовує наш більший набір даних, наше вхідне подання й схему донавчання.

Спочатку досліджуємо вплив NSP. У таблиці 5 показуємо, що вилучення NSP суттєво погіршує QNLI, MNLI й SQuAD 1.1. Далі оцінюємо навчання двонапрямних подань, порівнюючи «Без NSP» з «LTR & без NSP». Модель LTR поступається MLM в усіх завданнях, особливо на MRPC і SQuAD.

Для SQuAD інтуїтивно зрозуміло, що LTR погано передбачатиме токени: їхні приховані стани не мають правого контексту. Щоб сумлінно спробувати посилити LTR, ми додали зверху випадково ініціалізовану BiLSTM. Це суттєво поліпшує SQuAD, проте результати все одно набагато гірші, ніж у попередньо навчених двонапрямних моделей. Для завдань GLUE BiLSTM шкодить.

Можна також навчити окремі моделі LTR і RTL та представляти токен конкатенацією їхніх подань, як ELMo. Проте: (a) це вдвічі дорожче за одну двонапрямну модель; (b) це неприродно для QA, адже RTL не може обумовити відповідь запитанням; (c) це строго менш потужно, ніж глибока двонапрямна модель, яка може враховувати лівий і правий контексти на кожному шарі.

5.2 Вплив розміру моделі

Досліджуємо вплив розміру моделі на точність після донавчання. Ми навчили низку моделей BERT із різними кількостями шарів, прихованих одиниць і голів уваги; решта гіперпараметрів і процедура навчання відповідають описаним раніше.

Результати для вибраних завдань GLUE наведено в таблиці 6. Подаємо середню точність валідації за 5 випадковими перезапусками донавчання. Більші моделі послідовно поліпшують точність на всіх чотирьох наборах, навіть на MRPC із лише 3,600 розміченими прикладами, що істотно відрізняється від завдань попереднього навчання. Дещо несподівано, що такі великі поліпшення можливі для моделей, які вже досить великі порівняно з літературою. Наприклад, найбільший Transformer у Vaswani et al. (2017) має (L=6, H=1024, A=16) і 100M параметрів кодувальника, а найбільший знайдений у літературі — (L=64, H=512, A=2) і 235M параметрів (Al-Rfou et al., 2018). Натомість $\text{BERT}_{\text{BASE}}$ має 110M параметрів, а $\text{BERT}_{\text{LARGE}}$ — 340M.

Давно відомо, що збільшення моделі стабільно поліпшує великомасштабні завдання, як-от машинний переклад і мовне моделювання; це показує й перплексія LM на відкладених навчальних даних у таблиці 6. Проте, на нашу думку, це перша робота, яка переконливо демонструє: масштабування до надзвичайно великих моделей суттєво поліпшує й дуже малі завдання, якщо модель достатньо попередньо навчена. Peters et al. (2018b) отримали неоднозначні результати впливу збільшення попередньо навченої bi-LM з двох до чотирьох шарів; Melamud et al. (2016) побіжно зазначили, що збільшення прихованої розмірності з 200 до 600 допомогло, а до 1,000 — уже ні. Обидві роботи використовували підхід на основі ознак. Ми припускаємо, що пряме донавчання на цільових завданнях із дуже малою кількістю додаткових випадково ініціалізованих параметрів дає змогу спеціалізованим моделям скористатися більшими й виразнішими поданнями навіть за дуже малих цільових даних.

5.3 Підхід на основі ознак із BERT

Усі наведені досі результати BERT використовували донавчання: до попередньо навченої моделі додають простий класифікатор і спільно донавчають усі параметри на цільовому завданні. Однак підхід на основі ознак, де з моделі видобувають незмінні ознаки, має переваги. По-перше, не всі завдання легко представити кодувальником Transformer, тому потрібна додаткова спеціалізована архітектура. По-друге, обчислювально вигідно один раз попередньо обчислити дороге подання навчальних даних, а потім проводити багато експериментів із дешевшими моделями поверх нього.

Ми порівнюємо ці підходи на завданні розпізнавання іменованих сутностей (NER) CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003). На вході BERT використовуємо WordPiece зі збереженням регістру й максимальний доступний контекст документа. За стандартною практикою формулюємо це як розмічування, але без вихідного шару CRF. Вхід класифікатора токенів за набором міток NER — подання першого підтокена.

Для абляції донавчання застосовуємо підхід на основі ознак: видобуваємо активації одного чи кількох шарів *без* налаштування параметрів BERT. Ці контекстні вкладення подаємо на випадково ініціалізовану двошарову 768-вимірну BiLSTM перед класифікатором.

Результати наведено в таблиці 7. $\text{BERT}_{\text{LARGE}}$ конкурентна з найкращими методами. Найкращий варіант конкатенує подання токенів із чотирьох верхніх прихованих шарів попередньо навченого Transformer й відстає від донавчання всієї моделі лише на 0.3 F1. Це демонструє ефективність BERT і для донавчання, і для підходів на основі ознак.

Гіперпарам.				Точність валідації		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Таблиця 6: Абляція розміру BERT. #L — кількість шарів; #H — прихована розмірність; #A — кількість голів уваги. «LM (ppl)» — перплексія маскованої LM на відкладених навчальних даних.

Система	Валід. F1	Тест F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Донавчання		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
На основі ознак ($\text{BERT}_{\text{BASE}}$)		
Вкладення	91.0	-
Передостанній прихований	95.6	-
Останній прихований	94.9	-
Зважена сума останніх чотирьох шарів	95.9	-
Конкатенація останніх чотирьох шарів	96.1	-
Зважена сума всіх 12 шарів	95.5	-

Таблиця 7: Результати розпізнавання іменованих сутностей CoNLL-2003. Гіперпараметри обрано на валідації. Оцінки валідації й тесту усереднено за 5 випадковими перезапусками з цими гіперпараметрами.

6 Висновок

Нещодавні емпіричні поліпшення завдяки перенесенню навчання з мовних моделей показали, що насичене некероване попереднє навчання — невіддільна частина багатьох систем розуміння мови. Зокрема, навіть завдання з малими ресурсами можуть скористатися глибокими однонапрямними архітектурами. Наш головний внесок — узагальнення цих висновків

на глибокі *двонапрямні* архітектури, завдяки чому одна попередньо навчена модель успішно розв'язує широке коло завдань NLP.

Література

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Контекстні рядкові вкладення для розмічування послідовностей. У *Proceedings of the 27th International Conference on Computational Linguistics*, с. 1638–1649.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Мовне моделювання на рівні символів із глибшою самоувагою. *Препринт arXiv arXiv:1808.04444*.

Rie Kubota Ando and Tong Zhang. 2005. Підхід до навчання передбачувальних структур на кількох завданнях і нерозмічених даних. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. П'яте змагання PASCAL із розпізнавання текстового логічного слідування. У *TAC*. NIST.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Адаптація до предметної області через навчання структурних відповідностей. У *Proceedings of the 2006 conference on empirical methods in natural language processing*, с. 120–128. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. Великий розмічений корпус для навчання висновування природною мовою. У *EMNLP*. Association for Computational Linguistics.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Класові n-грамні моделі природної мови. *Computational linguistics*, 18(4):467–479.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017, завдання 1: семантична текстова подібність — багатомовне й міжмовне оцінювання. У *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, с. 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. Тест на мільярді слів для вимірювання прогресу статистичного мовного моделювання. *Препринт arXiv arXiv:1312.3005*.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Пари запитань Quora.

Christopher Clark and Matt Gardner. 2018. Просте й ефективне розуміння прочитаного за кількома абзацами. У *ACL*.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Напівкероване моделювання послідовностей із навчанням за перехресними поданнями. У *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, с. 1914–1925.

Ronan Collobert and Jason Weston. 2008. Єдина архітектура оброблення природної мови: глибокі нейронні мережі з багатозадачним навчанням. У *Proceedings of the 25th international conference on Machine learning*, с. 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Кероване навчання універсальних подань речень на даних висновування природною мовою. У *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, с. 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Напівкероване навчання послідовностей. У *Advances in neural information processing systems*, с. 3079–3087.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: великомасштабна ієрархічна база зображень. У *CVPR09*.

William B Dolan and Chris Brockett. 2005. Автоматична побудова корпусу перефразованих речень. У *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. MaskGAN: краще генерування тексту через заповнення пропусків. *Препринт arXiv arXiv:1801.07736*.

Dan Hendrycks and Kevin Gimpel. 2016. Поєднання нелінійностей і стохастичних регуляризаторів за допомогою гауссових лінійних одиниць похибки. *CoRR*, abs/1606.08415.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Навчання розподілених подань речень на нерозмічених даних. У *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Універсальне донавчання мовної моделі для класифікації тексту. У *ACL*. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Мнемонічний читач із підкріпленням для машинного розуміння прочитаного. У *IJCAI*.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Дискурсивні цілі для швидкого некерованого навчання подань речень. *CoRR*, abs/1705.00557.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: масштабний тестовий набір даних із віддаленим наглядом для розуміння прочитаного. У *ACL*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Вектори Skip-thought. У *Advances in neural information processing systems*, с. 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Розподілені подання речень і документів. У *International Conference on Machine Learning*, с. 1188–1196.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. Змагання схем Винограда. У *Aaai spring symposium: Logical formalizations of commonsense reasoning*, том 46, с. 47.

Lajanugen Logeswaran and Honglak Lee. 2018. Ефективний підхід до навчання подань речень. У *International Conference on Learning Representations*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Засвоєне в перекладі: контекстуалізовані вектори слів. У *NIPS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: навчання загального контекстного вкладення за допомогою двонапрямної LSTM. У *CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Розподілені подання слів і фраз та їхня композиційність. У *Advances in Neural Information Processing Systems 26*, с. 3111–3119. Curran Associates, Inc.

Andriy Mnih and Geoffrey E Hinton. 2009. Масштабована ієрархічна розподілена мовна модель. У D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, ред., *Advances in Neural Information Processing Systems 21*, с. 1081–1088. Curran Associates, Inc.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. Розкладна модель уваги для висновування природною мовою. У *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: глобальні вектори для подання слів. У *Empirical Methods in Natural Language Processing (EMNLP)*, с. 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Напівкероване розмічування послідовностей із двонапрямними мовними моделями. У *ACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Глибокі контекстуалізовані подання слів. У *NAACL*.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Аналіз контекстних вкладень слів: архітектура й подання. У *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, с. 1499–1509.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Поліпшення розуміння мови завдяки некерованому навчанню. Технічний звіт, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: понад 100,000 запитань для машинного розуміння тексту. У *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, с. 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Потік двонапрямної уваги для машинного розуміння. У *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Рекурсивні глибокі моделі семантичної композиційності на деревному банку тональності. У *Proceedings of the 2013 conference on empirical methods in natural language processing*, с. 1631–1642.

Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. 2018. U-Net: машинне розуміння прочитаного із запитаннями без відповіді. *Препринт arXiv arXiv:1810.06638*.

Wilson L Taylor. 1953. «Процедура Cloze»: новий інструмент вимірювання читабельності. *Journalism Bulletin*, 30(4):415–433.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Вступ до спільного завдання CoNLL-2003: мовонезалежне розпізнавання іменованих сутностей. У *CoNLL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Подання слів: простий і загальний метод напівкерованого навчання. У *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, с. 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Увага — це все, що вам потрібно. У *Advances in Neural Information Processing Systems*, с. 6000–6010.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Видобування й комбінування стійких ознак за допомогою знешумлювальних автокодувальників. У *Proceedings of the 25th international conference on Machine learning*, с. 1096–1103. ACM.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. GLUE: багатозадачний тест і платформа аналізу розуміння природної мови. У *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, с. 353–355.

Wei Wang, Ming Yan, and Chen Wu. 2018b. Мережі злиття ієрархічної уваги різної деталізації для розуміння прочитаного й відповідей на запитання. У *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Оцінки прийнятності за допомогою нейронної мережі. *Препринт arXiv arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. Тестовий корпус широкого охоплення для розуміння речень через висновування. У *NAACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Нейронна система машинного перекладу Google: подолання розриву між людським і машинним перекладом. *Препринт arXiv arXiv:1609.08144*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. Наскільки переносні ознаки глибоких нейронних мереж? У *Advances in neural information processing systems*, с. 3320–3328.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: поєднання локальної згортки з глобальною самоувагою для розуміння прочитаного. У *ICLR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: масштабний змагальний набір даних для висновування на основі здорового глузду. У *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Вирівнювання книжок і фільмів: до візуальних пояснень у формі історій через перегляд фільмів і читання книжок. У *Proceedings of the IEEE international conference on computer vision*, с. 19–27.

Додаток до «BERT: попереднє навчання глибоких двонапрямних трансформерів для розуміння мови»

Додаток складається з трьох розділів:

- Додаткові подробиці реалізації BERT подано в додатку A;
- Додаткові подробиці експериментів — у додатку B;
- Додаткові абляційні дослідження — у додатку C.

  Ці дослідження BERT охоплюють:

  – Вплив кількості кроків навчання;
  – Абляцію різних процедур маскування.

A Додаткові відомості про BERT

A.1 Ілюстрація завдань попереднього навчання

Нижче наводимо приклади завдань попереднього навчання.

Маскована LM і процедура маскування Нехай нерозмічене речення — `my dog is hairy`, а випадкове маскування обрало 4-й токен (`hairy`). Тоді процедура має такий вигляд:

- У 80% випадків: замінити слово на `[MASK]`, наприклад, `my dog is hairy` → `my dog is [MASK]`.
- У 10% випадків: замінити слово випадковим, наприклад, `my dog is hairy` → `my dog is apple`.
- У 10% випадків: залишити слово незмінним, наприклад, `my dog is hairy` → `my dog is hairy`. Це зміщує подання до справді спостережуваного слова.

Перевага процедури — кодувальник Transformer не знає, які слова треба буде передбачити чи які замінено випадковими, тому мусить підтримувати розподільне контекстне подання *кожного* вхідного токена. До того ж випадкова заміна зачіпає лише 1.5% усіх токенів (10% від 15%), тож, схоже, не шкодить розумінню мови. У розділі C.2 оцінюємо вплив цієї процедури.

На відміну від стандартного навчання мовної моделі, маскована LM передбачає лише 15% токенів пакета, тож для збіжності може знадобитися більше кроків попереднього навчання.

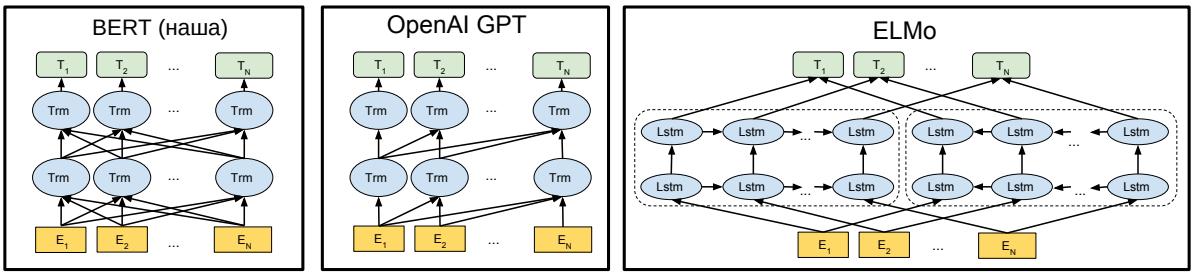


Рисунок 3: Відмінності архітектур попереднього навчання. BERT використовує двонапрямний Transformer. OpenAI GPT — Transformer зліва направо. ELMo конкатенує незалежно навчені LSTM зліва направо й справа наліво, утворюючи ознаки для цільових завдань. Із трьох моделей лише подання BERT спільно обумовлені лівим і правим контекстами на всіх шарах. Крім архітектурних відмінностей, BERT і OpenAI GPT використовують донавчання, а ELMo — підхід на основі ознак.

У розділі C.1 показуємо, що MLM справді збігається дещо повільніше за модель зліва направо, яка передбачає кожен токен, проте емпіричний виграш MLM набагато переважає додаткові витрати навчання.

Передбачення наступного речення Завдання передбачення наступного речення ілюструють такі приклади.

Вхід = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Мітка = `IsNext`

Вхід = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Мітка = `NotNext`

A.2 Процедура попереднього навчання

Щоб утворити вхідну навчальну послідовність, вибираємо з корпусу два фрагменти тексту, які називаємо «реченнями», хоча зазвичай вони значно довші за одне речення (але можуть бути й коротшими). Перший отримує вкладення A, другий — B. У 50% випадків B справді йде після A, а у 50% є випадковим реченням — для завдання передбачення наступного речення. Сумарна довжина становить ≤ 512 токенів. Маскування LM застосовуємо після токенізації WordPiece з однаковою часткою маскування 15%, не виділяючи частини слів окремо.

Навчаємо з пакетами по 256 послідовностей (256 послідовностей * 512 токенів = 128,000 токенів/пакет) протягом 1,000,000 кроків, що становить приблизно 40 епох над корпусом із 3.3 мільярда слів. Використовуємо Adam із темпом 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L2-згасанням ваг 0.01, розігрівом темпу протягом перших 10,000 кроків і його лінійним спаданням. Імовірність проріджування на всіх шарах — 0.1. Услід за OpenAI GPT використовуємо активацію `gelu` (Hendrycks and Gimpel, 2016) замість стандартної `relu`. Втрата — сума середньої правдоподібності маскованої LM і середньої правдоподібності передбачення наступного речення.

$\text{BERT}_{\text{BASE}}$ навчали на 4 Cloud TPU в конфігурації Pod (усього 16 чипів TPU).[13] $\text{BERT}_{\text{LARGE}}$ навчали на 16 Cloud TPU (усього 64 чипи). Кожне попереднє навчання тривало 4 дні.

Довші послідовності непропорційно дорогі, адже вартість уваги квадратична за довжиною. Для прискорення в експериментах попередньо навчаємо модель на довжині 128 протягом 90% кроків. Решту 10% навчаємо на довжині 512, щоб засвоїти позиційні вкладення.

A.3 Процедура донавчання

Більшість гіперпараметрів донавчання такі самі, як у попередньому навчанні, крім розміру пакета, темпу й кількості епох. Імовірність проріджування завжди 0.1. Оптимальні значення залежать від завдання, але такі діапазони добре працюють для всіх завдань:

- **Розмір пакета**: 16, 32
- **Темп навчання (Adam)**: 5e-5, 3e-5, 2e-5
- **Кількість епох**: 2, 3, 4

Великі набори (наприклад, 100k+ розмічених прикладів) значно менш чутливі до вибору гіперпараметрів, ніж малі. Донавчання за-

[13] https://cloudplatform.googleblog.com/2018/06/Cloud-TPU-now-offers-preemptible-pricing-and-global-availability.html

звичай дуже швидке, тому доцільно повністю перебрати наведені значення й обрати найкращу модель за валідацією.

A.4 Порівняння BERT, ELMo та OpenAI GPT

Тут досліджуємо відмінності нещодавніх популярних моделей навчання подань: ELMo, OpenAI GPT і BERT. Їхні архітектури наочно порівняно на рисунку 3. Зазначимо, що, крім відмінностей архітектури, BERT і OpenAI GPT використовують донавчання, а ELMo — підхід на основі ознак.

Найближчий до BERT наявний метод попереднього навчання — OpenAI GPT, що навчає LM Transformer зліва направо на великому текстовому корпусі. Чимало рішень BERT навмисно зроблено максимально близькими до GPT, щоб порівняння мало мінімум сторонніх відмінностей. Головна теза роботи: двонапрямність і два завдання попереднього навчання з розділу 3.1 пояснюють більшість емпіричних поліпшень. Водночас є й інші відмінності навчання BERT і GPT:

- GPT навчали на BooksCorpus (800M слів); BERT — на BooksCorpus (800M) і Вікіпедії (2,500M).
- GPT вводить роздільник речень (`[SEP]`) і класифікаційний токен (`[CLS]`) лише під час донавчання; BERT навчає `[SEP]`, `[CLS]` і вкладення речень `A/B` під час попереднього навчання.
- GPT навчали 1M кроків із пакетами по 32,000 слів; BERT — 1M кроків із пакетами по 128,000 слів.
- GPT використовувала темп 5e-5 для всіх експериментів донавчання; BERT обирає для завдання найкращий темп за валідаційною вибіркою.

Щоб відокремити вплив цих відмінностей, проводимо абляційні експерименти в розділі 5.1. Вони показують, що більшість поліпшень справді походить від двох завдань попереднього навчання й двонапрямності, яку вони забезпечують.

A.5 Ілюстрації донавчання для різних завдань

Донавчання BERT для різних завдань показано на рисунку 4. Спеціалізовані моделі утворюємо з BERT та одного додаткового вихідного шару, тож з нуля треба навчити мінімум параметрів. Завдання (a) і (b) мають рівень послідовності, а (c) і (d) — токена. На рисунку E — вхідне вкладення, T_i — контекстне подання токена i, [CLS] — спеціальний символ виходу класифікації, а [SEP] — спеціальний роздільник несуміжних послідовностей токенів.

B Докладна постановка експериментів

B.1 Докладний опис експериментів GLUE

Наші результати GLUE в таблиці 1 взято з `https://gluebenchmark.com/leaderboard` і `https://blog.openai.com/language-unsupervised`. GLUE охоплює наведені нижче набори даних; їхній опис спочатку узагальнено в Wang et al. (2018a).

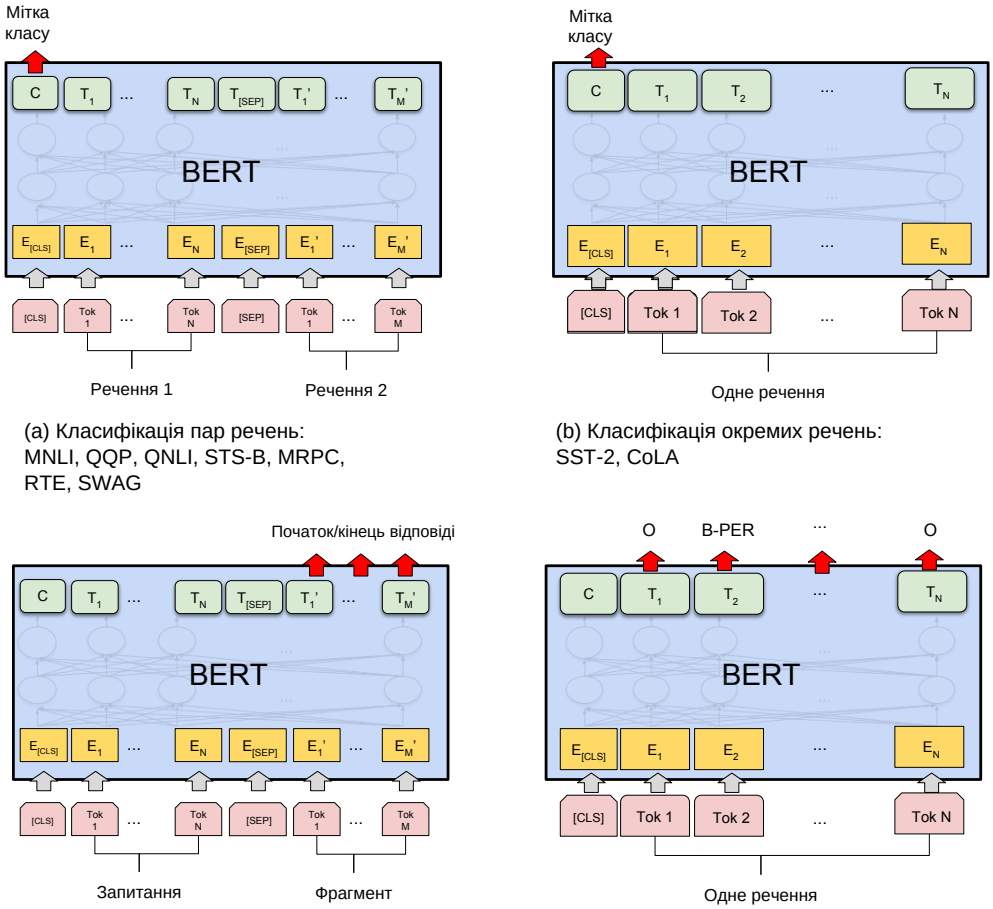
MNLI Multi-Genre Natural Language Inference — масштабне краудсорсингове завдання класифікації логічного слідування (Williams et al., 2018). Для пари речень потрібно передбачити, чи друге *випливає* з першого, *суперечить* йому або є *нейтральним* щодо нього.

QQP Quora Question Pairs — бінарна класифікація: визначити, чи є два запитання на Quora семантично еквівалентними (Chen et al., 2018).

QNLI Question Natural Language Inference — версія Stanford Question Answering Dataset (Rajpurkar et al., 2016), перетворена на бінарну класифікацію (Wang et al., 2018a). Позитивні приклади — пари (запитання, речення), що містять правильну відповідь; негативні — пари з того самого абзацу, які відповіді не містять.

SST-2 Stanford Sentiment Treebank — бінарна класифікація окремих речень із кінорецензій, тональність яких розмічена людьми (Socher et al., 2013).

CoLA Corpus of Linguistic Acceptability — бінарна класифікація окремих речень: передбачити, чи є англійське речення лінгвістично «прийнятним» (Warstadt et al., 2018).



(a) Класифікація пар речень: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Класифікація окремих речень: SST-2, CoLA

(c) Відповіді на запитання: SQuAD v1.1

(d) Розмічування окремих речень: CoNLL-2003 NER

Рисунок 4: Ілюстрації донавчання BERT для різних завдань.

STS-B Semantic Textual Similarity Benchmark — набір пар речень із новинних заголовків та інших джерел (Cer et al., 2017). Їм надано оцінки від 1 до 5, що позначають семантичну подібність двох речень.

MRPC Microsoft Research Paraphrase Corpus містить автоматично видобуті з інтернет-новин пари речень із людською розміткою їхньої семантичної еквівалентності (Dolan and Brockett, 2005).

RTE Recognizing Textual Entailment — бінарне завдання логічного слідування, подібне до MNLI, але зі значно меншими навчальними даними (Bentivogli et al., 2009).[14]

WNLI Winograd NLI — малий набір для висновування природною мовою (Levesque et al., 2011). На сайті GLUE зазначено проблеми його побудови,[15] і кожна навчена система, подана до GLUE, показала точність нижчу за базові 65.1, що їх дає передбачення класу більшості. Тому для справедливого порівняння з OpenAI GPT ми виключаємо цей набір. У нашій заявці GLUE завжди передбачали клас більшості.

C Додаткові абляційні дослідження

C.1 Вплив кількості кроків навчання

Рисунок 5 показує точність MNLI на валідації після донавчання контрольної точки, попередньо навченої протягом k кроків. Це дає змогу відповісти на такі запитання:

1. Запитання: чи BERT справді потребує такого обсягу попереднього навчання

[14] У цій статті наводимо лише результати донавчання на одному завданні. Багатозадачне донавчання потенційно може поліпшити їх ще більше. Наприклад, спільне навчання з MNLI істотно поліпшувало наші результати RTE.

[15] https://gluebenchmark.com/faq

(128,000 слів/пакет * 1,000,000 кроків) для високої точності донавчання?
Відповідь: так, $\text{BERT}_{\text{BASE}}$ отримує майже 1.0% додаткової точності MNLI після 1M кроків проти 500k.

2. Запитання: чи попереднє навчання MLM збігається повільніше за LTR, адже передбачає лише 15% слів пакета, а не кожне?
Відповідь: MLM справді збігається трохи повільніше за LTR. Проте за абсолютною точністю MLM починає перевершувати LTR майже відразу.

C.2 Абляція різних процедур маскування

У розділі 3.1 зазначено, що під час попереднього навчання за ціллю MLM BERT застосовує змішану стратегію маскування цільових токенів. Нижче досліджуємо вплив різних стратегій.

Мета цих стратегій — зменшити невідповідність між попереднім навчанням і донавчанням, адже символ `[MASK]` ніколи не трапляється на другому етапі. Наводимо валідаційні результати MNLI й NER. Для NER подаємо і донавчання, і підхід на основі ознак: очікуємо, що в останньому невідповідність посилиться, бо модель не зможе скоригувати подання.

Результати наведено в таблиці 8. Позначення Маска означає заміну цільового токена символом `[MASK]` для MLM; Той самий — збереження токена; Випадк. — заміну іншим випадковим токеном.

Числа ліворуч у таблиці — імовірності конкретних стратегій попереднього навчання MLM (BERT використовує 80%, 10%, 10%).

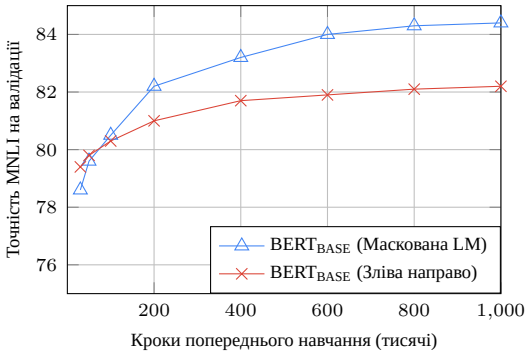


Рисунок 5: Абляція кількості кроків навчання. Показано точність MNLI після донавчання параметрів моделі, попередньо навчених протягом k кроків. Вісь x — значення k.

Частки маскування			Валідаційні результати		
Маска	Той самий	Випадк.	MNLI Донавч.	NER Донавч.	NER Ознаки
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Таблиця 8: Абляція різних стратегій маскування.

Праворуч подано валідаційні результати. Для підходу на основі ознак конкатенуємо останні 4 шари BERT як ознаки: це найкращий підхід у розділі 5.3.

Донавчання виявилося напрочуд стійким до різних стратегій маскування. Проте, як і очікувалося, лише стратегія Маска створила проблеми в NER на основі ознак. Цікаво, що лише стратегія Випадк. також значно поступається нашій.