

Търсене и извличане на информация.  
Приложение на дълбоко машинно обучение

Зимен семестър 2024/2025

## Задание за курсов проект Невронен машинен превод чрез генеративен езиков модел

15 януари 2025 г.

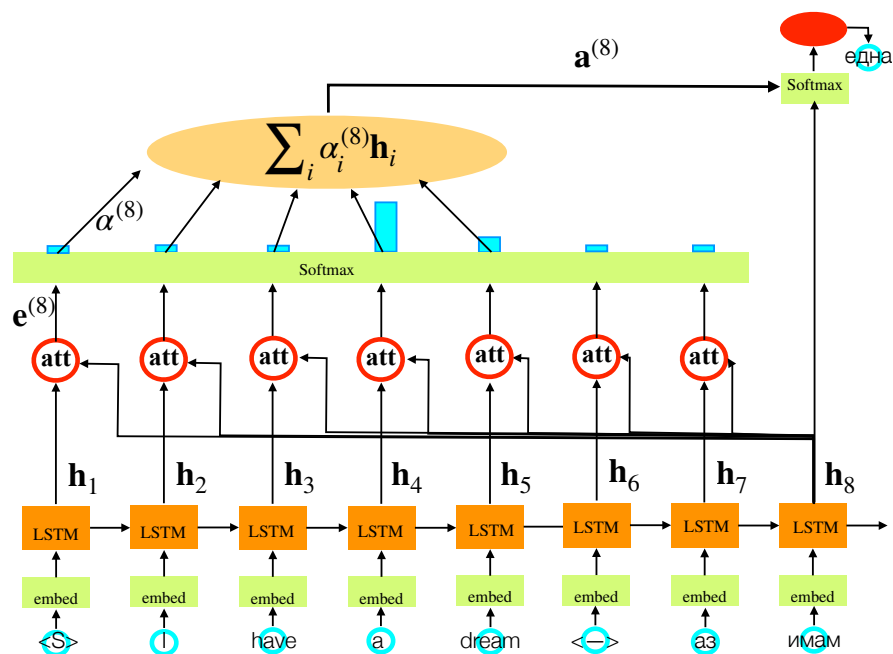
### Общ преглед

За курсовия проект ще трябва да реализирате генеративен езиков модел с техники за дълбоко машинно обучение. Проектът е планиран така, че да ви даде възможност бързо да се задълбочите в експерименти с дълбоко машинно обучение. В рамките на проекта ще имате възможност да имплементирате съвременни техники и да експериментирате със собствени нови архитектури.

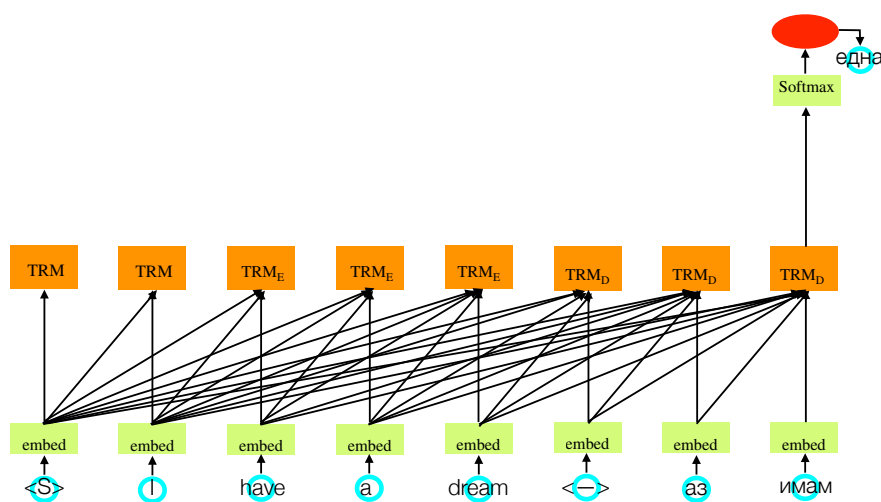
### Задача: Невронен машинен превод чрез генеративен езиков модел от английски към български език

В машинния превод целта е да преведем изречение от входния език (английски) в целевия език (български). За да реализираме тази функционалност ние ще имплементираме генеративен езиков модел. Езиковият модел ще обучим с последователности, състоящи се от изречение на входен език следвано от специален разделител и съответното изречение – превод на целевия език. По този начин, при задаване на изречението на входния език, следвано от специалния разделител, ние ще сме в състояние да генерираме съответен превод на целевия език.

Фигура 1



(а) Схема на примерна рекурентна невронна мрежа с механизъм за внимание за генеративен машинен превод.



(б) Схема на примерна невронна мрежа с Transformer архитектура за генеративен машинен превод.

За разлика от специализираните невронни архитектури за машинен превод, генеративният езиков модел позволява и допълнителни функционалности. Например бихме могли да зададем освен изречението на входния език, също и начало на превода на целевия език и да генерираме края на превода. Също така може да зададем само начало на изречението на входния език и да генерираме края на изречението на входния език, следвано от превода на цялото изречение на целевия език.

В това задание се изисква да се имплементира невронна архитектура за езиков модел. Имате пълната свобода да използвате рекурентна, конволюционна или трансформер архитектура, както и техни комбинации. Препоръчително е архитектурата да включва и някаква форма на механизъм за “внимание” (Attention). На Фигура 1. (а) е представена схема на примерна LSTM базирана архитектура за осъществяване на невронен машинен превод. Схема на невронен машинен превод, базиран на Transformer архитектура е дадена на Фигура 1. (б).

## Изискване за съдържание на курсовата работа

Курсовата работа трябва да съдържа:

1. Всички програмни модули и параметри, с които е имплементирана вашата невронна мрежа.
2. Програмите за подготовка на данни, обучение и тестване на решението, ако са различни от съответните помощни програми, включени в пакета.
3. Обучен модел на вашата реализация, който достига докладваното от вас качество на превода.
4. Програма, позволяваща превод на произволен корпус от изречения на входния език и записването на резултата в нов файл, ако е различна от съответната помощна програма, включена в пакета.
5. Програма, позволяваща генерирането на продължение на зададена начална последователност, ако е различна от съответната помощна програма, включена в пакета.
6. Кратко описание / доклад – в рамките на 2-3 страници – на вашето решение. Описанието следва да съдържа:

(а) Вашите имена и факултетен номер.

- (б) Достатъчно пълно описание на архитектурата, която сте реализирали. Описанието на архитектурата следва да съдържа и параметрите, които сте използвали, така че описанието да е достатъчно за репродуцирането ѝ.
- (в) Цитирания и референции към всички чужди програми и източниците на информация, които сте използвали.
- (г) Описание за начина на обучение на модела и проведените експерименти за настройване на параметрите за обучение.
- (д) Резултат от оценяване на модела върху тестовия корпус – перплексия и BLEU резултат.

## Ограничения и препоръки за архитектурата на модела

Цел на курсовата работа е от една страна да даде възможно най-голяма свобода и креативност за реализирането на модела. От друга страна се цели да постави някакви рамки по отношение на платформата и методологията, за да се поставят студентите при близки условия.

## Ограничения и изисквания

- Моделът трябва да е имплементиран с използване на платформата Pytorch.
- Моделът трябва да използва архитектура за генеративен езиков модел, като езиковия модел може да бъде реализиран с архитектура по ваш избор – рекурентна, конволюционна, трансформер, комбинация от архитектури или друга.
- За обучението на модела не се разрешава използването на други корпуси, извън приложения в пакета (вижте раздела Корпус).
- Не се разрешава използването на модели, които са предварително обучени върху други корпуси.
- Предадената имплементация трябва да реализира възможност за превод на корпус на изречения. Това може да стане като се използва функционалността `translate` на приложената програма `run.py` (вижте раздела Помощни програми). Но е допустима и друга имплементация, която трябва да е добре описана.

- Моделът трябва да имплементира метод за генерация при зададена произволна начална последователност за реализиране на разширената функционалност – задаване и на начало на превода или задаване само на началото на изречението на входния език. Това може да стане като се използва функционалността **generate** на приложената програма **run.py**, но е допустима и друга имплементация.

## Препоръки

Дадените по-долу препоръки са само за ориентация. Няма изискване за реализация на коя да е от тях. Също така, имате пълната свобода да реализирате други елементи към вашата архитектура, които не са описани по-долу, стига да не противоречат на описаните в предишния раздел ограничения и изисквания.

- Препоръчително е да се започне с по-проста архитектура, която евентуално да се усложнява, ако не дава желаните резултати.
- Реализацията на търсене по лъча не е задължително. Обикновено алчното търсене дава само с 1-2 точки по-нисък BLEU резултат.
- Влагането на думите може да бъде реализирано както по обичайния начин със слой за влагане, така и чрез конволюция на символно ниво, както беше показано на упражнение или чрез използване на кодиране до поддуми. По-сложните методи може да подобрят малко резултата, но изискват повече работа.
- Реализирането на архитектура с “внимание” не е задължително, но е силно препоръчително. Без механизъм за “внимание” качеството на превода ще бъде значително по-ниско. Специализирана рекурентна архитектура за машинен превод е описана в [1] и [2].
- В статията [2] е изследвано влиянието на различните параметри при рекурентна невронна архитектура върху качеството на превода. За да спестите време за обхватни експерименти може да се запознаете с тази работа.
- Реализирането на архитектура, използваща Transformer блокове и многоглаво внимание, както е описано в [3], може да повиши качеството на превода.
- Обучението на Transformer архитектура обикновено изисква повече време и внимание при настройването на параметрите.

- В статията [4] е изследвано влиянието на различните параметри на Transformer архитектурата върху качеството на превода. За да спестите време за обхватни експерименти може да се запознаете с тази работа.
- За да може де се предвиждат целеви думи, които не са в речника на целевия език може да се добави допълнителен рекурентен слой на ниво символи, както е показано в статията [5] или да използвате кодиране до поддуми, както е показано в статията [6].

## Корпус

В пакета на заданието в директорията `en_bg_data` е предоставен двуезичен английско-български подравнен корпус. Корпусът се състои от:

- 180000 двойки изречения за обучение във файловете `train.en` и `train.bg`.
- 1000 двойки изречения за валидация във файловете `dev.en` и `dev.bg`.
- 6000 двойки изречения за тестване във файловете `test.en` и `test.bg`.

## Помощни програми

В пакета са включени помощни програми, които свободно може да използвате във вашата курсова работа. Използването на тези програми не е задължително. Вие може да ги променяте свободно или да ги заменяте с други по ваше усмотрение.

### `model.py`

Ако искате да ползвате пълната функционалност на приложените помощни програми е необходимо във файла `model.py` да имплементирате модел за машинен превод в обект `LanguageModel`, който да имплементира следните методи:

- `__init__(self, ...)` – конструктор на обекта,
- `forward(self, source)` – метода трябва по партида от входни последователности `source` да върне съответната крос-ентропия,
- `generate(self, prefix)` – метода трябва да извършва генерира продължение на дадената начална последователност `prefix`.

## `utils.py`

Във файла `utils.py` са имплементирани функциите за подготовка на тренировачни данни. В този файл са имплементирани следните функции и обекти:

- Обект `progressBar` – обект за визуализиране на прогрес.
- Функция `readCorpus(fileName)` – функцията чете текстов файл от изречения разделени с нов ред и връща списък от изречения, като всяко изречение е списък от думи.
- Функция `getDictionary(corpus, startToken, endToken, unkToken, padToken, transToken, wordCountThreshold = 2)` – от даден корпус извлича всички думи и връща речник на думите с повече от зададения брой срещания във вид на хеш, който връща индекса на съответната дума. Към речника се добавят специални думи за начало, край, непозната дума, попълване и начало на превод.
- Функция `prepareData(sourceFileName, targetFileName, sourceDevFileName, targetDevFileName, startToken, endToken, unkToken, padToken, transToken)` – подготвя данните необходими за трениране.

## `run.py`

Във файла `run.py` са имплементирани функционалности за трениране, прилагане и тестване на модел. Очаква се във файла `model.py` да създадете своя имплементация на невронен езиков модел за машинен превод. Програмата `run.py` използва файла `parameters.py`, в който се прочитат параметрите, необходими за изпълнение на съответните функционалности. В `run.py` са имплементирани следните команди:

- `python run.py prepare` – подготвя данните като изчита съответните корпуси и записва на диска необходимите python обекти.
- `python run.py train` – извършва първоначален процес на обучение на модел. Предполага се, че в `model.py` е имплементиран модел `LanguageModel`, който реализира невронен машинен превод и неговият `forward` метод по партида от последователности връща съответната крос-ентропия. По време на обучението, през `test_every` брой стъпки се измерва крос-ентропията спрямо корпуса за верификация. Ако стойността е по-ниска, то модела се запазва на диска. Може да ви се наложи да промените скоростта на спускане

`learning_rate` по време на обучението. След `maxEpochs` брой епохи обучението завършва.

- `python run.py extratrain` – извършва допълнителен цикъл на трениране върху последно записания модел. Тази команда позволява да се продължи обучението след прекъсване на обучението.
- `python run.py perplexity <sourceCorpus> <targetCorpus>` – измерва перплексията на вече записан модел върху тестов корпус с входни изречения дадени във файла `<sourceCorpus>` и целеви изречения дадени във файла `<targetCorpus>`.
- `python run.py translate <sourceCorpus> <resultCorpus>` – превежда тестов корпус с входни изречения дадени във файла `<sourceCorpus>` в целеви изречения. Целевите изречения се записват във файла `<resultCorpus>`. За да работи тази команда в модела `LanguageModel` трябва да бъде имплементиран метод `generate(self, prefix)`.
- `python run.py generate <prefix>` – генерира последователност с начало низа `<prefix>`. За тази команда отново трябва да се използва метода `generate(self, prefix)` на класа `LanguageModel`.  
Пример:  
`python run.py generate "<S> I have a"`
- `python run.py bleu <targetCorpus> <resultCorpus>` – измерва BLEU точките между корпус от целеви изречения преведени от референтен преводач дадени във файла `<targetCorpus>` и корпус от целеви изречения получени от машинния превод дадени във файла `<resultCorpus>`.

## Използване на чужди програми извън Pytorch

1. Вие имате право да използвате съществуващи програми и библиотеки извън стандартния Pytorch пакет. Трябва обаче **ясно да цитирате** своите източници и да посочите кои части от проекта не са ваша работа. Ако използвате или заемате код от която и да е външна библиотека, опишете как използвате външния код и предоставете връзка към източника. Също така, по време на защитата трябва да сте в състояние да обясните и да отговорите на всички въпроси, свързани с вашата реализация, включително и детайли по имплементацията на използваните от вас чужди програми и библиотеки. Вашата курсова работа ще бъде оценена според вашите приноси и доколко разбирате представената реализация.



2. Вие можете свободно да обсъждате идеи и подробности за курсовата работа с други студенти. При никакви обстоятелства обаче не е разрешено да разглеждате кода на другите или да включвате техния код във вашия проект.
3. Вие нямате право да споделяте кода си публично (например в GitHub), преди курсът да е приключил.

## **Забележки**

1. За курсовата работа не се предоставя код за тестване. Препоръчва се вие сами да си направите тестови скриптове, с които да се уверите в коректността на програмите ви.
2. Очаква се най-много време да ви отнеме експериментирането с настройка на параметрите на вашия модел. Един пълен цикъл на обучение отнема няколко часа. Поради това е необходимо да си планирате добре времето, така че да успеете да се справите навреме с работата.
3. За обучението на модела ще ви бъде необходимо значително машинно време. Ако не разполагате с мощен компютър с графична карта, то може да се възползвате от услугата Google Colab <https://colab.research.google.com>, където безплатно се предоставя изчислителна среда с инсталирани Python и Pytorch, която може да се конфигурира да използва графична карта (GPU).

## **Критерии за оценяване**

Курсовият проект ще бъде оценен цялостно. Това означава, че ще бъдат разгледани различни фактори при определяне на оценката: креативността, качеството на превода на вашия модел, сложността и техническата коректност на вашата реализация, вашия конкретен принос, усилията, които сте приложили, и качеството на вашето описание.

За да се оцени достигнатото качество на превод ще бъде използвана програмата за измерване на BLEU, която е приложена в пакета на заданието. Измерването ще бъде извършено върху тестов корпус от текстове от европейския парламент, който не е приложен към материалите. Отличните реализации се очаква да достигнат BLEU резултат около и над 35 точки.

## Инструкция за предаване на курсовата работа

Изисква се в Moodle да бъде предаден архив FNXXX.zip (където XXX е вашият факултетен номер), в който са пакетирани всички файлове от изисканото съдържание.

Пожелавам ви успех!

## Литература

- [1] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1409.0473>
- [2] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, “Massive exploration of neural machine translation architectures,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1442–1451. [Online]. Available: <https://www.aclweb.org/anthology/D17-1151>
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: <https://arxiv.org/abs/1706.03762>
- [4] A. Araabi and C. Monz, “Optimizing transformer for low-resource neural machine translation,” 2020. [Online]. Available: <https://arxiv.org/abs/2011.02266>
- [5] M.-T. Luong and C. D. Manning, “Achieving open vocabulary neural machine translation with hybrid word-character models,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1054–1063. [Online]. Available: <https://www.aclweb.org/anthology/P16-1100>
- [6] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” 2016. [Online]. Available: <https://arxiv.org/abs/1508.07909>