Додаток А

Слайди презентації

Предмет дослідження – моделювання ігрових ситуацій.

Задачі дослідження:

- порівняння бібліотек для обробки природної мови;
- обрання найбільш відповідних бібліотек;
- отримання необхідних вхідних даних;
- вибір метрик для порівняння бібліотек;
- порівняння отриманих результатів.

ОСНОВНІ ВИДИ ОБРОБКИ ТЕКСТУ

- Токенізація по реченням
- Токенізація по словам
- Леммитизація та стемминг тексту
- Стоп-слова
- Bag-of-words (мішок слів)

ТОКЕНІЗАЦІЯ ПО РЕЧЕННЯМ

```
Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000
years to archeological discoveries in the Middle East. It is a two player game where each playe
r has fifteen checkers which move between twenty-four points according to the roll of two dice.
```

```
Backgammon is one of the oldest known board games.

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle Ea
st.

It is a two player game where each player has fifteen checkers which move between twenty-four p
oints according to the roll of two dice.
```

ТОКЕНІЗАЦІЯ ПО СЛОВАМ

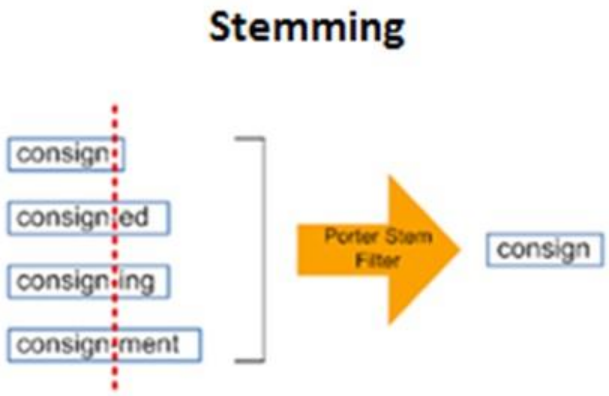
```
Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000
years to archeological discoveries in the Middle East. It is a two player game where each playe
r has fifteen checkers which move between twenty-four points according to the roll of two dice.
```

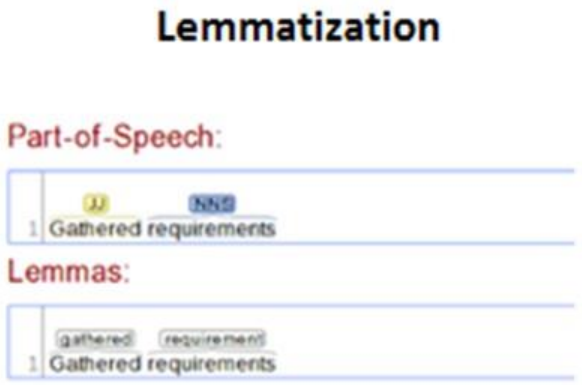
```
['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeologi
cal', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checke
rs', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'o
f', 'two', 'dice', '.']
```

СТЕМІНГ ТА ЛЕММАТИЗАЦІЯ ТЕКСТУ

Стемінг – скорочення слова до його основи

Лематизація – приведення слова до початкової формі





STOP-СЛОВА

Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

BAG-OF-WORDS (МІШОК СЛІВ)

Порівняння бібліотек

	SPACY	NLTK	CORENLP
Моделі нейронної мережі	✓	✗	✓
Інтегровані вектори слова	✓	✗	✗
Багатомовна підтримка	✓	✓	✓
Токенізація	✓	✓	✓
Стемінг та Лемматизація	✓	✓	✓
Аналіз залежності	✓	✗	✓
Суб'єктне об'єднання	✓	✗	✗
Роздільна здатність основної конференції	✗	✗	✓

ДЕТАЛЬНЕ ПОРІВНЯННЯ ШВИДКОСТІ

SYSTEM	TOKENIZE	TAG	PARSE
spaCy	0.2ms	1ms	19ms
CoreNLP	0.18ms	10ms	49ms
NLTK	4ms	443ms	n/a

ВИСНОВКИ

В ході виконання магістерскої атестаційної роботи було порівняно бібліотеки обробки природної мови. Виділені найбільш актуальні метрики порівняння бібліотек, також розглянуто різні засоби і підходи до інтелектуального аналізу тексту. Основну увагу було приділено класифікації тексту, оскільки це дало найкращі рішення для аналізу текстів щодо їх впливу.

Додаток Б

Програмний код

```
@dataclass
class ModelArguments:
    """
    Arguments pertaining to which model/config/tokenizer we are going to fine-tune, or train from scratch.
    """
   model_name_or_path: Optional[str] = field(
        default=None,
        metadata={
            "help": "The model checkpoint for weights initialization. Leave None if you want to train a model from scratch."
        },
    )
    model_type: Optional[str] = field(
        default=None,
        metadata={"help": "If training from scratch, pass a model type from the list: " + ", ".join(MODEL_TYPES)},
    )
    config_name: Optional[str] = field(
        default=None, metadata={"help": "Pretrained config name or path if not the same as model_name"}
    )
    tokenizer_name: Optional[str] = field(
        default=None, metadata={"help": "Pretrained tokenizer name or path if not the same as model_name"}
    )
```

```
    cache_dir: Optional[str] = field(
        default=None, metadata={"help": "Where do you want to store the pretrained models downloaded from s3"}
    )
@dataclass
class DataTrainingArguments:
    """
    Arguments pertaining to what data we are going to input our model for training and eval.
   train_data_file: Optional[str] = field(
        default=None, metadata={"help": "The input training data file (a text file)."}
    )
    eval_data_file: Optional[str] = field(
        default=None,
        metadata={"help": "An optional input evaluation data file to evaluate the perplexity on (a text file)."},
    )
    line_by_line: bool = field(
        default=False,
        metadata={"help": "Whether distinct lines of text in the dataset are to be handled as distinct sequences."},
    )
    mlm: bool = field(
        default=False, metadata={"help": "Train with masked-language modeling loss instead of language modeling."}
    )
    mlm_probability: float = field(
        default=0.15, metadata={"help": "Ratio of tokens to mask for masked language modeling loss"}
    )
```

```
    block_size: int = field(
        default=-1,
        metadata={
            "help": "Optional input sequence length after tokenization."
            "The training dataset will be truncated in block of this size for training."
            "Default to the model max input length for single sentence inputs (take into account special tokens)."
        },
    )
    overwrite_cache: bool = field(
        default=False, metadata={"help": "Overwrite the cached training and evaluation sets"}
    )
def get_dataset(args: DataTrainingArguments, tokenizer: PreTrainedTokenizer, evaluate=False, local_rank=-1):
    file_path = args.eval_data_file if evaluate else args.train_data_file
    if args.line_by_line:
        return LineByLineTextDataset(
            tokenizer=tokenizer, file_path=file_path, block_size=args.block_size, local_rank=local_rank
        )
    else:
        return TextDataset(
            tokenizer=tokenizer, file_path=file_path, block_size=args.block_size, local_rank=local_rank,
        )
def main():
    parser = HfArgumentParser((ModelArguments, DataTrainingArguments, TrainingArguments))
```

```
    model_args, data_args, training_args = parser.parse_args_into_dataclasses()

    if data_args.eval_data_file is None and training_args.do_eval:
        raise ValueError(
            "Cannot do evaluation without an evaluation data file. Either supply a file to --
eval_data_file "
            "or remove the --do_eval argument."
        )
    if (
        os.path.exists(training_args.output_dir)
        and os.listdir(training_args.output_dir)
        and training_args.do_train
        and not training_args.overwrite_output_dir
    ):
        raise ValueError(
            f"Output directory ({training_args.output_dir}) already exists and is not empty.
Use --overwrite_output_dir to overcome."
        )
    # Setup logging
    logging.basicConfig(
        format="%(asctime)s - %(levelname)s - %(name)s -   %(message)s",
        datefmt="%m/%d/%Y %H:%M:%S",
        level=logging.INFO if training_args.local_rank in [-1, 0] else logging.WARN,
    )
    logger.warning(
        "Process rank: %s, device: %s, n_gpu: %s, distributed training: %s, 16-bits
training: %s",
        training_args.local_rank,
        training_args.device,
        training_args.n_gpu,
```

```
        bool(training_args.local_rank != -1),
        training_args.fp16,
    )
    logger.info("Training/evaluation parameters %s", training_args)
    if model_args.config_name:
        config = AutoConfig.from_pretrained(model_args.config_name,
cache_dir=model_args.cache_dir)
    elif model_args.model_name_or_path:
        config = AutoConfig.from_pretrained(model_args.model_name_or_path,
cache_dir=model_args.cache_dir)
    else:
        config = CONFIG_MAPPING[model_args.model_type]()
        logger.warning("You are instantiating a new config instance from scratch.")

    if model_args.tokenizer_name:
        tokenizer = AutoTokenizer.from_pretrained(model_args.tokenizer_name,
cache_dir=model_args.cache_dir)
    elif model_args.model_name_or_path:
        tokenizer = AutoTokenizer.from_pretrained(model_args.model_name_or_path,
cache_dir=model_args.cache_dir)
    else:
        raise ValueError(
            "You are instantiating a new tokenizer from scratch. This is not supported, but
you can do it from another script, save it,"
            "and load it from here, using --tokenizer_name"
        )

    if model_args.model_name_or_path:
        model = AutoModelWithLMHead.from_pretrained(
            model_args.model_name_or_path,
```

```
            from_tf=bool(".ckpt" in model_args.model_name_or_path),
            config=config,
            cache_dir=model_args.cache_dir,
        )
    else:
        logger.info("Training new model from scratch")
        model = AutoModelWithLMHead.from_config(config)

    model.resize_token_embeddings(len(tokenizer))

    if config.model_type in ["bert", "roberta", "distilbert", "camembert"] and not
data_args.mlm:
        raise ValueError(
            "BERT and RoBERTa-like models do not have LM heads but masked LM
heads. They must be run using the --mlm "
            "flag (masked language modeling)."
        )

    if data_args.block_size <= 0:
        data_args.block_size = tokenizer.max_len
        # Our input block size will be the max possible for the model
    else:
        data_args.block_size = min(data_args.block_size, tokenizer.max_len)

    # Get datasets
    train_dataset = (
        get_dataset(data_args, tokenizer=tokenizer, local_rank=training_args.local_rank)
        if training_args.do_train
        else None
    )
```

```
    eval_dataset = (
        get_dataset(data_args, tokenizer=tokenizer, local_rank=training_args.local_rank,
evaluate=True)
        if training_args.do_eval
        else None
    )
    data_collator = DataCollatorForLanguageModeling(
        tokenizer=tokenizer, mlm=data_args.mlm,
mlm_probability=data_args.mlm_probability
    )

    # Initialize our Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        data_collator=data_collator,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        prediction_loss_only=True,
    )

    # Training
    if training_args.do_train:
        model_path = (
            model_args.model_name_or_path
            if model_args.model_name_or_path is not None and
os.path.isdir(model_args.model_name_or_path)
            else None
        )
        trainer.train(model_path=model_path)
```

```
        trainer.save_model()
        # For convenience, we also re-save the tokenizer to the same directory,
        # so that you can share your model easily on huggingface.co/models =)
        if trainer.is_world_master():
            tokenizer.save_pretrained(training_args.output_dir)

    # Evaluation
    results = {}
    if training_args.do_eval and training_args.local_rank in [-1, 0]:
        logger.info("*** Evaluate ***")

        eval_output = trainer.evaluate()

        perplexity = math.exp(eval_output["loss"])
        result = {"perplexity": perplexity}

        output_eval_file = os.path.join(training_args.output_dir, "eval_results_lm.txt")
        with open(output_eval_file, "w") as writer:
            logger.info("***** Eval results *****")
            for key in sorted(result.keys()):
                logger.info("  %s = %s", key, str(result[key]))
                writer.write("%s = %s\n" % (key, str(result[key])))

        results.update(result)
    return results
def _mp_fn(index):
    # For xla_spawn (TPUs)
    main()
if __name__ == "__main__":
```

Додаток В

Апробація результатів роботи

42 • Die wichtigsten Vektoren für die Entwicklung der Wissenschaft im Jahr 2020 • Band 1

ОБРОБКА ПРИРОДНОЇ МОВИ ЯК ЧАСТИНА ШТУЧНОГО ІНТЕЛЕКТУ

Мачула Олена Володимирівна
здобувач магістерського ступеня факультету комп'ютерних наук
Харківський національний університет радіоелектроніки
Науковий керівник: Четвериков Г.Г.
професор кафедри програмної інженерії
Харківський національний університет радіоелектроніки
Україна

Обробка природної мови – загальний напрямок штучного інтелекту і математичної лінгвістики. Вивчає проблеми комп'ютерного аналізу і синтезу природних мов.

Через велику універсальність та величезну кількість різних застосувань у NLP, в цій главі буде зосереджено лише тематика, яка є важливою для цієї роботи. Розділ розпочинається з пояснення виявлення знань та міжгалузевого стандартного процесу для видобутку даних. Він продовжується розслідуванням NLP загалом і обговорює технології та методи, які мають вирішальне значення для виконання цієї дипломної роботи. Внаслідок сучасні тенденції в NLP, таких як нейронна мережа прямого поширення та рекурентна нейронна мереж.

Для розуміння більшої картини роботи я коротко введу терміни процес пошуку корисних знань в базах даних та глибинний аналіз даних. У літературі видобуток даних та KD у базах даних часто трактуються як синоніми, але насправді, майнінг даних – це під задача KD-процесу, схожа на описану CRISP-DM у наступному розділі. DM може розглядатися як фаза моделювання, яка витягує створення та зразки підготовлених даних з різними підходами.

Міжгалузевий стандартний процес для дослідження даними був розроблений в 2000 р. і застосовується до багатьох різних проблем з видобутком даних. В деталях, процес складається з шести основних етапів – розуміння бізнесу, розуміння даних, підготовка даних, моделювання, оцінка та розробка. Крім того, весь процес є ітераційним, тобто кожен крок буде оброблятися кілька разів.

Розуміння бізнесу – головне завдання на початку процесу обміну даними. З точки зору бізнесу, весь проект слід розуміти та аналізувати. Крім того, ця фаза намагається перетворити всю проблему в перспективу обміну даними, яка зосереджена на похідних цілях бізнесу.

Рисунок 1 Міжгалузевий стандартний процес для дослідження даними

Збір даних – це початковий крок у розумінні даних. Цей крок служить для забезпечення початкові уявлення про дані, формують ранні гіпотези та виявляють проблеми в даних. Таким чином можна переробити всю проблематику і повернутися до справирозуміння. Згодом акцентується увага на підготовці даних, де різні методи застосовуються для отримання начального набору даних, які можуть бути використані як вхідні дані при моделюванні фази. Відповідні завдання для підготовки даних в NLP описані в наступному розділі. Крім того, попередньо обробляючи дані зазвичай розбиваються на менші підмножини перед моделюванням. Один називається навчальним набором, на якому тренується модель, другий – тестовим набором і він використовується для оцінки. Іноді третій набір, який називається набором перевірки, надає дані для навчання і параметри DM-моделі.

Моделювання – це етап вибору та застосування різних моделей до підготовлених даних. Низка методів для кожної окремої задачі вибору даних. Часто необхідно повернутися до етапу підготовки даних, оскільки деякі моделі вимагають різного рівня форми введення. Перед розробкою, де модель передається у виробництво. Необхідно оцінити застосовані моделі за допомогою показників, орієнтованих на бізнес-цілі. Тому крок оцінки вимірює ефективність усіх попередніх етапів за допомогою дотримання цілей бізнесу.

Обробка природних мов є основною дослідницькою сферою цієї дипломної роботи. Це пов'язано з іншим сферами, таким як штучний інтелект або машинне навчання.

Різні методи попередньої обробки функцій, наприклад, видалення стоп слова та векторна модель. У цьому розділі також розглядаються показники оцінки та обговорюється класична машина модель навчання.

В розділі розміщено теми – штучний інтелект, обробка природних мов, машинне навчання та глибоке навчання в одному контекст. ШІ – це дуже широкий термін і є способом описати системи, які здатні «думати». У літературі існує багато різних пояснень, як визначити цю тему. Інтерпретація буде прийняти, чому включає NLP і ставить його по відношенню до ШІ. ШІ складається з чотирьох основних частини, які є машинним навчанням, аргументація, плануванням та NLP. Обґрунтування дозволяє машині надавати пропозиції на основі даних, тоді як планування дає змогу системам самостійно діяти при інтерпретації даних.

У ньому є багато різних застосувань, які відносяться до неструктурованої природної мови. Наприклад, сферами його застосування є машинний переклад, розпізнавання мови, діалогові системи, розпізнавання іменованих об'єктів, пошук інформації та класифікація тексту. Таким чином, домен NLP охоплює всі взаємодії між комп'ютером і людиною, шляхом використання писемної чи розмовної природної мови.

Ця дослідницька та прикладна робота, що стосується маніпулювання та розуміння природних мов. Обробка людської мови заснована на розумінні наміченого значення повідомлення, яке є важким навіть для людей, наприклад, коли використовується іронія. Усі компоненти природної мови, таких як фонетика, фонологія, морфологія, синтаксис, семантика та прагматики, повинні враховуватися, щоб отримати повне розуміння повідомлення.

Фонетика – це про акустичні властивості звуку, що видається голосовим трактом людини. Вивчає, як звуки фізично побудовані, наприклад, з мовою або губами. Звук конкретної людської мови вивчається фонологією. Наприклад, англійська мова має 45 відмінних звуків, званих фонемами. Фонетика та фонологія особливо важливими аспектами розпізнавання мовлення при перетворенні звуків у реальні слова, які можна обробити комп'ютером.

Морфологія стосується значення та архітектури слів. Слово-будування і лематизація, які описані нижче, базуються на цьому компоненті шляхом перетворення слова. Впорядкованість слів та побудова граматично правильних речень досліджено синтаксис. Навпаки, семантика вивчає значення побудованих речень шляхом використання синтаксичних та морфологічних словоформ.

Для отримання загального значення повідомлення, прагматика використовує контекст ситуації. Наприклад, припустимо, хтось запитує: «Чи могли б ви передати сіль?». Дано, у контексті ситуації питання фактично є проханням передати сіль, а не запитання, чи хтось може це зробити. Тому комп'ютер потрібно брати всі частини природної мови врахувати, щоб нею користуватися.

Одне відоме визначення машинного навчання засноване на ідеї досвіду та ілюструє навчальну частину в ML: «Комп'ютерна програма, як кажуть, вивчає досвід щодо деяких клас завдань і міра ефективності, якщо її виконання при завданнях в, які вимірюється, покращується з досвідом.».

Висновки.

ШІ та розбивається на чотири підкатегорії: контрольоване навчання, непідконтрольне навчання, глибоке навчання та навчання з підкріпленням. Контрольоване та непідконтрольне навчання – це два основних типи проблем пошуку даних. Різниця між цими завданнями полягає в структурі даних про навчання.

Список використаних джерел:

1. Gobinda G. Chowdhury. "Natural Language Processing". In: Annual Review of Information Science and Technology, 2003. – 51-89.

2. Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: 2014.

3. Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu. Deep Learning Based Text Classification: A Comprehensive Review // 2005.