استعراض لأهم تقنيات تضمين الكلمات في معالجة اللغة الطبيعية

طالبة الماجستير لما براك

قسم هندسة البرمجيات ونظم المعلومات – كلية الهندسة المعلوماتية – جامعة البعث

إشراف: د. يسر الأتاسي د. سهيل حمود

الملخص

تعد معالجة اللغات الطبيعية مهمة لأنها تساعد في حل الغموض في اللغة وتضيف بنية رقمية مفيدة إلى البيانات للعديد من التطبيقات النهائية، مثل التعرف على الكلام أو تحليلات النص.

العديد من خوارزميات التعلم الآلي وجميع بنيات التعلم العميق تقريبًا غير قادرة على معالجة السلاسل أو النص العادي في شكلها الأولي. إنها تتطلب أرقامًا كمدخلات لأداء أي نوع من الوظائف، سواء كان ذلك التصنيف أو الانحدار وما إلى ذلك بعبارات عامة. ومع الكم الهائل من البيانات الموجودة في تنسيق النص، من الضروري استخراج المعرفة منه وبناء التطبيقات.

في هذا البحث نوضح مراحل معالجة النص وصولاً إلى طرق تضمين النص الذي يمثل طريقة حديثة لتمثيل الكلمات كمتجهات رقمية ليتم معالجتها من قبل خوارزميات التعلم الآلي، حيث توفر مقدمة موجزة عن المجال ونظرة عامة سريعة على بنى وطرق التعلم العميق وتمثيلات الكلمات السياقية العميقة مثل نموذج التضمين من نماذج اللغة.

الكلمات المفتاحية: خوارزميات تعلم عميق –معالجة اللغة الطبيعية –تضمين الكلمات.

A review of the most important Word Embedding Techniques in Natural Language Processing

Abstract

Natural language processing is important because it helps solve ambiguities in the language and adds a useful numerical structure to data for many end-to-end applications.

Many machine learning algorithms and nearly all deep learning architectures are unable to process strings or plain text in their initial form. It requires numbers as input to perform any kind of job, such as classification, regression, etc. in general terms. With the vast amount of data contained in text format, it is imperative to extract knowledge from it and build applications.

In this research, we explain the stages of text processing down to methods of text embedding that represents a modern way of representing words as digital vectors to be processed by machine learning algorithms, as it introduces the field and a quick overview of deep learning structures such as Embeddings from Language Models (ELMO) Model.

Key words: Deep Learning Algorithm-Natural Language Processing - Word Embedding.

1- مقدمة:

معالجة اللغة الطبيعية (Natural language processing) هي فرع من فروع الذكاء الاصطناعي تساعد أجهزة الحاسب على فهم وتفسير ومعالجة اللغة البشرية.

تستمد معالجة اللغات الطبيعية من العديد من التخصصات، بما في ذلك علوم الحاسب واللغويات الحاسوبية، في سعيها لسد الفجوة بين التواصل البشري وفهم الحاسب.

يشمل مجال معالجة اللغة الطبيعية (NLP) مجموعة متنوعة من الموضوعات، والتي تتضمن المعالجة الحسابية وفهم اللغات البشرية. منذ الثمانينيات، اعتمد المجال بشكل متزايد على الحساب المستند إلى البيانات والذي يتضمن الإحصاء والاحتمالات والتعلم الآلي [1]، [2].

على الرغم من أن معالجة اللغة الطبيعية ليست علمًا جديدًا، فإن التكنولوجيا تتقدم بسرعة بفضل الاهتمام المتزايد بالاتصالات بين الإنسان والآلة، بالإضافة إلى توفر البيانات الضخمة والحوسبة القوية والخوارزميات المحسنة، وهذا ما قادنا إلى الاهتمام بتطوير هذا المجال.

كإنسان، يمكنك التحدث والكتابة باللغة الإنجليزية أو العربية أو الإسبانية أو الصينية. لكن اللغة الأصلية للحاسب -المعروفة باسم رمز الآلة أو لغة الآلة -غير مفهومة إلى حد كبير لمعظم الناس. عند أدنى مستويات جهازك، لا يحدث الاتصال بالكلمات ولكن من خلال الملايين من الأصفار والأرقام التي تنتج إجراءات منطقية.

تعتبر معالجة اللغة الطبيعية مهمة بسبب وجود كميات كبيرة من البيانات النصية فهي تساعد أجهزة الحاسب على التواصل مع البشر بلغتهم وتوسّع المهام الأخرى المتعلقة باللغة.

على سبيل المثال، يتيح NLP لأجهزة الحاسب قراءة النص وسماع الكلام وتفسيره وقياس المشاعر وتحديد الأجزاء المهمة.

من خلال استخدام NLP ومكوناتها[2]، يمكن للمرء تنظيم مجموعات ضخمة من البيانات النصية، وتنفيذ العديد من المهام الآلية وحل مجموعة واسعة من المشكلات مثل –التلخيص التلقائي، والترجمة الآلية، والتعرف على الكيانات المسماة، واستخراج العلاقة، وتحليل المشاعر، والتعرف على الكلام، وتجزئة الموضوع إلخ.

2– مشكلة البحث:

يمكن لآلات اليوم أن تحلل البيانات المستندة إلى اللغة أكثر من البشر، دون إجهاد وبطريقة متسقة وغير متحيزة[3]. بالنظر إلى الكم الهائل من البيانات غير المهيكلة التي يتم إنشاؤها كل يوم، من السجلات الطبية إلى وسائل التواصل الاجتماعي، ستكون الأتمتة بالغة الأهمية لتحليل بيانات النص والكلام بكفاءة.

وتساعد NLP في هيكلة اللغات الطبيعية التي تعتبر مصدر بيانات غير منظم وذلك لأن لغة الإنسان معقدة ومتنوعة بشكل مذهل. فنحن نعبر عن أنفسنا بطرق لا نهائية، شفهيًا وكتابيًا.

لا يوجد فقط مئات اللغات واللهجات، ولكن داخل كل لغة مجموعة فريدة من القواعد النحوية والمصطلحات والكلمات العامية.

عندما نكتب، غالبًا ما نخطئ في تهجئة الكلمات أو اختصارها، أو نحذف علامات الترقيم. بينما يتم الآن استخدام التعلم الخاضع للإشراف وغير الخاضع للإشراف، ونماذج التعلم العميق، على نطاق واسع لنمذجة اللغة البشرية، إلا أن هناك أيضًا حاجة إلى الفهم النحوي والدلالي والخبرة في المجال التي ليست بالضرورة موجودة في مناهج التعلم الآلي هذه. العديد من خوارزميات التعلم الآلي وجميع بنيات التعلم العميق تقريبًا غير قادرة على معالجة السلاسل أو النص العادي في شكلها الأولي. إنها

تتطلب أرقامًا كمدخلات لأداء أي نوع من الوظائف، سواء كان ذلك التصنيف أو الانحدار وما إلى ذلك بعبارات عامة.

3- هدف البحث:

سنقدم في هذه المقالة دراسة لطرق تضمين الكلمات Word Embedding، حيث أن تمثيل الكلمات هي اللبنات الأساسية لأنظمة معالجة اللغة الطبيعية الحديثة.

تمنحنا تمثيلات الكلمات بشكل أساسي طريقة لتحويل جزء من النص إلى أرقام تستطيع أنظمة الحاسب لدينا قراءتها. على مر السنين، تم تطوير العديد من تمثيلات الكلمات التي تهدف إلى التقاط الدلالات والاعتماد النحوي للكلمات. يعد تطوير تمثيلات الكلمات الجيدة التي تلتقط التفاصيل الدقيقة للغة البشرية مهمة صعبة للغاية. يجب أن تضع الكلمات الجيدة الكلمات المتشابهة معًا. اعتبار أساسي آخر هو تعدد المعاني اللغوي (polysemy) هذه هي القدرة على أن تأخذ الكلمة معاني مختلفة في سياقات مختلفة. فمثلا،

An Apple a day keeps the doctor away.
Apple released its latest iPhone model today.

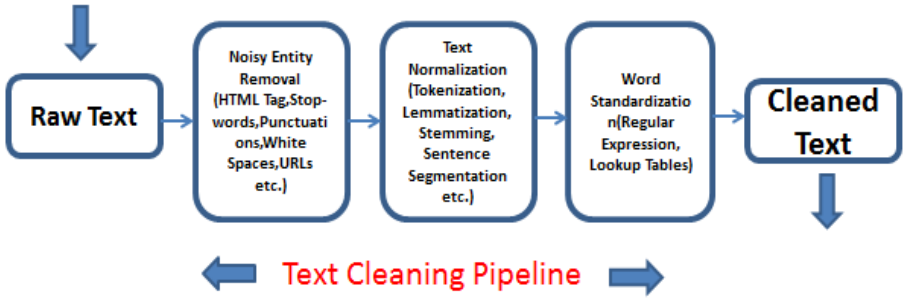
في الجملتين السابقتين، لكلمة "Apple" معاني مختلفة، ويجب أن تأخذ تمثيلات الكلمات ذلك في الاعتبار ولا تحسب نفس التمثيلات للكلمة بشكل مستقل عن السياق.

4- معالجة النص (Text Preprocessing):

نظرًا لأن النص هو الشكل غير المنظم لجميع البيانات المتاحة، فإن أنواعًا مختلفة من الضجيج موجودة فيه ولا يمكن تحليل البيانات بسهولة دون أي معالجة مسبقة. تُعرف العملية الكاملة لتنظيف النص وتوحيده، مما يجعله خاليًا من الضجيج وجاهزًا للتحليل بمعالجة النص.

يتكون في الغالب من ثلاث خطوات:

- إزالة الضجيج Noise Removal
- Lexicon Normalization
- Object Standardization



الشكل –1–مراحل تنظيف النص

4-1 إزالة الضجيج (Noise Removal):

يمكن تحديد أي جزء من النص لا علاقة له بسياق البيانات والمخرجات النهائية على أنه ضجيج.

على سبيل المثال – كلمات التوقف في اللغات المختلفة مثلا (الكلمات المستخدمة بشكل شائع للغة –هي، أنا، من، في، إلخ)، أيضا عناوين URL أو الروابط تعتبر ضجيج، كيانات الوسائط الاجتماعية (مذكورة، علامات التصنيف)، علامات الترقيم. تتناول هذه الخطوة إزالة جميع أنواع كيانات الضجيج الموجودة في النص.

تتمثل الطريقة العامة لإزالة الضجيج في إعداد قاموس للكيانات المزعجة، وتكرار كائن النص بواسطة الرموز (أو بالكلمات)، مما يلغي تلك الرموز المميزة الموجودة في قاموس الضجيج.

```
# Sample code to remove noisy words from a text
```

```
noise_list = ["is", "a", "this", "..."]

def _remove_noise(input_text):

    words = input_text.split()

    noise_free_words = [word for word in words if word not in
noise_list]

    noise_free_text = " ".join(noise_free_words)

    return noise_free_text

_remove_noise("this is a sample text")

>>> "sample text"
```

يوجد طريقة أخرى لإزالة بيانات الضجيج من النص باستخدام التعابير المنتظمة regular expressions

```
# Sample code to remove a regex pattern

import re

def _remove_regex(input_text, regex_pattern):

    urls = re.finditer(regex_pattern, input_text)

    for i in urls:

        input_text = re.sub(i.group().strip(), '', input_text)

    return input_text
```

```
regex_pattern = "#[\w]*"

_remove_regex("remove this #hashtag from analytics ",
regex_pattern)

>>> "remove this  from analytics "
```

4-2 التحليل المعجمي (Lexicon Normalization):

نوع آخر من الضجيج النصي يتعلق بالأشكال المتعددة التي تظهر بكلمة واحدة.

على سبيل المثال play،player، played، plays، playing هي الأشكال المختلفة للكلمة –"play"، على الرغم من أنها تعنى معان مختلفة ولكن جميعها متشابهة من حيث السياق.

تقوم الخطوة الحالية بتحويل جميع التباينات في الكلمة إلى شكلها الطبيعي (المعروف أيضًا باسم lemma). تعد التسوية Normalization خطوة محورية لهندسة الميزات مع النص لأنها تحول الميزات عالية الأبعاد (N ميزات مختلفة) إلى مساحة منخفضة الأبعاد (ميزة واحدة)، وهو طلب مثالي لأي نموذج تعلم آلة ML.

فيما يلي بعض تقنيات بناء الجملة التي يمكن استخدامها:

- **Stemming** هو عملية بدائية قائمة على القواعد لتجريد اللواحق ("ing"، "ly" ، "es" ،" s " إلخ) من الكلمات.

وهي الأداة التي تسمح بتجريد اي كلمة من جميع الإضافات التي فيها ، والعودة للمصدر الأصلي لها و هي معتمدة على فكرة إرجاع الكلمة إلى أصلها ، وحذف جميع الإضافات عليها سواء في البداية أو النهاية ، فكلمات مثل , playing player , plays , played كلها تعود للكلمة play، و هي مفيدة بشكل كبير في التعرف على معاني الكلمات ، وكذلك في ضم جميع الكلمات ذات الأصل الواحد

إلى نفس الكلمة . فكلمات مثل (كتاب، كتب، مكتبة، كاتب، مكتوب)، كلها من الأصل كتب، ونفس الأمر في أغلب اللغات .

كذلك جملتي:

I was riding in the car.

I was taking a ride in the car .

بنفس المعنى حتى لو اختلفت الكلمات لكن أحيانا تفشل أداة stemming في إيجاد جذر الكلمة، وتقوم بحذف حرف متحرك في النهاية، بينما هو حرف أصل مثل since تصبح sinc.

- **Lemmatization** : هو إجراء منظم وخطوة للحصول على شكل جذر الكلمة ، ويستخدم المفردات (أهمية القاموس للكلمات) والتحليل الصرفي (بنية الكلمات والعلاقات النحوية).

هي مشابهة لـ stemming في الفكرة لكنها أكثر قوة وفعالية، فهي لا تكتفي بإزالة الحروف الزائدة في الكلمات، ولكن بالبحث في معناها وأساسها، فكلمات مثل been، was أصلها هو be وهكذا.

كما أنه يراعي المعنى في الجملة فكلمة meeting قد يكون أصلها meet لو كانت فعل مضارع، وقد يكون أصلها هو نفسه meeting في حالة كانت اسم وليس فعل (بمعنى اجتماع).

```
from nltk.stem.wordnet import WordNetLemmatizer

lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer

stem = PorterStemmer()
```

```
word = "multiplying"

lem.lemmatize(word, "v")

>> "multiply"

stem.stem(word)

>> "multipli"
```

- **الترميزTokenization** :

وهي عملية تقسيم الجملة إلى عدد من الأجزاء أو الكلمات (Token) أو يعني فصل كل كلمة على حدة، للتعامل معها ومعرفة نوعها وما إلى ذلك، وهي قائمة على فصل الكلمات من الجمل، بحيث تكون كل كلمة وحدها، وهناك نوعين منها:

- Word tokenizer: أي فصل الكلمات.
- Sentence tokenizer: أي فصل كل جملة على حدة.

- **تقسيم الجمل Sentence Segmentations:**

وهي الخاصة بتقسيم الكلام كله إلى جمل حسب بدايتها والسياق، وهي عملية مهمة للغاية، وقد تتم بأحد طريقتين إما بعلامة واضحة أنها لنهاية جملة مثل؟ أو! وأحيانا تتم عبر استخدام علامة غير واضحة مثل النقطة. والتي قد يكون لها استخدامات أخرى مثل نهاية اختصار مثل .Dr أو أرقام عشرية وهكذا.

3-4 Object Standardization :

غالبًا ما تحتوي البيانات النصية على كلمات أو عبارات غير موجودة في أي قواميس معجمية قياسية. ولم يتم التعرف على هذه القطع من قبل محركات البحث والنماذج.

بعض الأمثلة هي –الاختصارات، والوسم (hashtag) مع الكلمات المرفقة ، والعامية. بمساعدة التعبيرات العادية وقواميس البيانات المعدة يدويًا، يمكن إصلاح هذا النوع من الضجيج، يستخدم النص البرمجي أدناه طريقة البحث في القاموس لاستبدال اللغة العامية لوسائل التواصل الاجتماعي من النص.

```
lookup_dict = {'rt':'Retweet', 'dm':'direct message', "awsm" :
"awesome", "luv" :"love", "..."}

def _lookup_words(input_text):

    words = input_text.split()

    new_words = []

    for word in words:

        if word.lower() in lookup_dict:

            word = lookup_dict[word.lower()]

        new_words.append(word) new_text = " ".join(new_words)

        return new_text

_lookup_words("RT this is a retweeted tweet by John Doe")

>> "Retweet this is a retweeted tweet by John Doe "
```

5- هندسة الميزات على البيانات النصية (Text to Features):

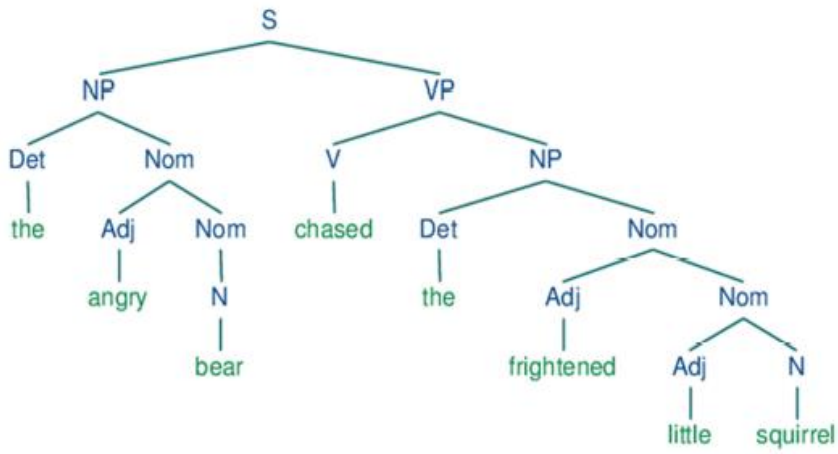
لتحليل البيانات التي تمت معالجتها مسبقًا، يجب تحويلها إلى ميزات. اعتمادًا على الاستخدام، يمكن إنشاء ميزات النص باستخدام تقنيات متنوعة –التحليل النحوي

والكيانات / N–grams / الميزات المستندة إلى الكلمات والميزات الإحصائية وتضمين الكلمات.

5-1 التحليل النحوي (Syntactic Parsing):

يتضمن التحليل النحوي تحليل الكلمات في الجملة من أجل القواعد وترتيبها بطريقة تُظهر العلاقات بين الكلمات. تعد قواعد التبعية وجزء من علامات الكلام من السمات المهمة لتركيب النص.

نتناول هنا بناء هيكلية الكلمات، والتي ترسم العلاقة بين الكلمات في الجملة بناء على قواعد اللغة والأسس النحوية، وتوضح مدى اعتمادية كل منها على الآخر.



وعلينا أن نتعرف على مجموعة من الرموز ومعناها:

Symbol	Means	المعنى
S	Sentence	الجملة
NP	Noun Phrase	جملة إسمية
VP	Verb Phrase	جملة فعلية
Det	Determiner	كلمات الوصل
PP	prepositional phrase	جملة الوصل
ADJP	Adjective Phrase	جملة صفة
ADVP	Adverb Phrase	جملة حال
N	noun	اسم
V	verb	فعل
P	preposition	حرف جر

وهناك نموذجين من هذا الهيكل:

الأول هو: النموذج الدائري constituency

والذي يعتمد على تجزئة الجملة إلى أجزاء صغيرة، ومنها ربط كل كلمة بالكلمات المتعلقة بها، ثم عمل ربط هيكلي بينهم وبين بعضهم فلو كان لدينا جملة

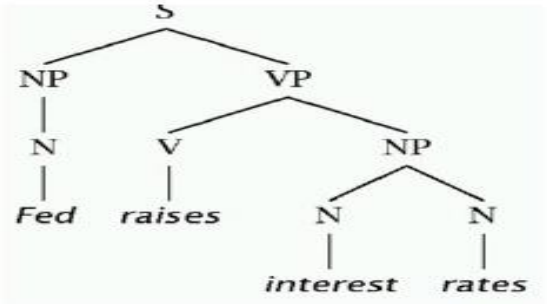
Fed raises interest rate

فكلمة fed وحدها لا تعني شيء، و fed raisesلها معنى لكن ناقص، و raises interestغير مفهومة، بينما interest rate وحدها هي جملة مفيدة، وهي جملة اسمية NP.

ثم يتم ضمها مع الكلمة الأخرى التابعة لها، و هي raises لتكون جملة raises interest rateجملة كاملة، وهي جملة فعلية VP .

أخيراً يتم رفع الأمر مستوى أعلى لإضافة fed والتي هي وحدها NP لتكون كلها هي S.

وإذا كان التكوين صحيح، فسترى أن إعادة ترتيب الأجزاء كل جزء على حدة سيعطي نفس المعنى.



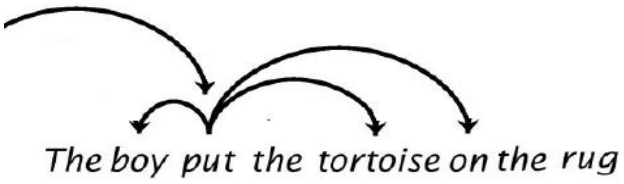
أما النوع الثاني فهو: هيكل الاعتمادية

والذي يقوم على تناول أهم كلمة في الجملة، ثم نبدأ بتحديد ما الذي يعتمد عليه، ونكمل العملية، فجملة:

The boy put the tortoise on the rug

أهم كلمة هنا هي put وبالتالي نقوم بعمل سهم من الخارج إلى هذه الكلمة

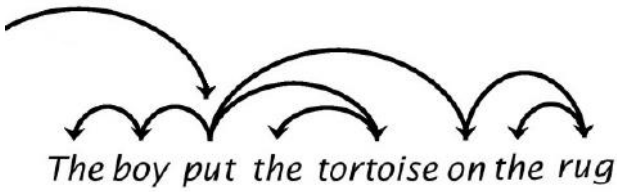
وهذا الفعل، نرى هنا ثلاث كلمات معتمدة عليه، وهي الفاعل، والمفعول به، ومكان الفعل، فيكون هناك أسهم ثلاث تتجه منها هكذا:



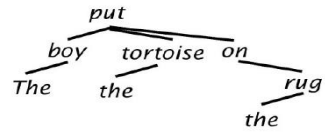
مع التأكيد على أن في جملة أخرى قد لا يتواجد الفاعل أو المفعول، أو قد يتواجد أشياء أخرى مثل زمن الفعل، فجملة:

Train moved yesterday quickly

فهنا نرى أن المرتبط بالفعل move هو الفاعل والزمان والحال ثم نقوم بعمل ربط لكل كلمة من الكلمات التي قمنا بالتعامل معها بالفعل، لنرى هي تعتمد على ماذا هكذا:



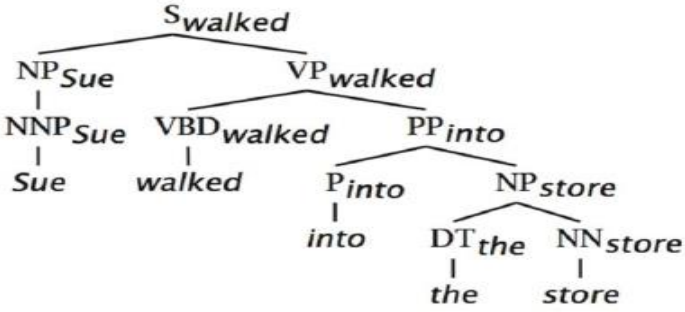
فيكون الشكل النهائي:



مثال آخر:

Sue walked into the store

وهنا نرى العلاقة بين النوعين بشكل واضح، فالتقسيمات معتمدة على علاقة الاعتماد




```
from nltk import word_tokenize, pos_tag

text = "I am learning Natural Language Processing "

tokens = word_tokenize(text)
```

```
print pos_tag(tokens)

>>>  [('I',  'PRP'),  ('am',  'VBP'),  ('learning',  'VBG'),
('Natural', 'NNP'),('Language', 'NNP'),

('Processing', 'NNP']
```

5-2 اكتشاف الكيانات (Entities as features):

نمذجة الموضوع (Topic Modelling) والتعرف على الكيان المحدد (Named-Entity Recognition) هما طريقتان رئيسيتان لاكتشاف الكيان في معالجة اللغة الطبيعية.

A. **نمذجة الموضوع (Topic Modelling):** نمذجة الموضوع هي عملية التعرف تلقائيًا على الموضوعات الموجودة في مجموعة نصية، وهي تستمد الأنماط المخفية بين الكلمات الموجودة في المجموعة بطريقة غير خاضعة للإشراف. يتم تعريف الموضوعات على أنها "نمط متكرر من المصطلحات المتزامنة في مجموعة". ينتج عن نموذج الموضوع الجيد –"الصحة"، "الطبيب"، "المريض"، "المستشفى" لموضوع –الرعاية الصحية، و "المزرعة"، "المحاصيل"، "القمح" لموضوع –"الزراعة".

B. **التعرف على الكيان المحدد (Named-Entity Recognition NER):** هي الخاصة بالتعرف على كلمات هامة و تصنيفها، مثل أسماء الأشخاص، أو المؤسسات أو أسماء الدول أو المدن، الوقت، الأموال، النسب المئوية ، وكذلك يقوم بتحديد أسماء الأعلام من أسماء الاشخاص و الشركات و المدن و العملات ، وهكذا ومن تطبيقات NER :

- يمكن ربط الأسماء المستخرجة بروابط لها من (ويكيبيديا).
- يتم ربط منتجات بشركات معينة.
- يتم ربط أسئلة معينة بإجابات لها في مكان آخر.

C. نموذج N-Grams: هو نموذج يقوم بتعيين الاحتمالات للجمل وتسلسل الكلمات , يمكنك التفكير في N-gram على أنه تسلسل للكلمات N ، و بالتالي لدينا ما يسمى Unigrams لتناول كلمة واحدة ،و نظام bigram الذي يعني التعامل مع كلمتين ، ونظام trigram للتعامل مع ثلاث كلمات ،كمان أن هناك 4-gram و 5-gram وهكذا .

3-5 الميزات الإحصائية Statistical features:

- معامل TF – IDF(term frequency-inverse document frequency)
- سمات التردد/الكثافة Frequency / Density
- سمات قابلية القراءة Readability Features

TF-IDF (مصطلح تردد الوثيقة العكسية): يحدد هذا النهج الإحصائي مدى ملاءمة الكلمة داخل نص في مجموعة من الوثائق ، وغالبًا ما تستخدم لاستخراج الكلمات الأساسية ذات الصلة من النص. تزداد أهمية الكلمة بناءً على عدد المرات التي تظهر فيها في النص (تردد النص)، ولكنها تتناقص بناءً على تكرار ظهورها في مجموعة النصوص (تكرار المستند العكسي).

6- تضمين الكلمات (text vectors) Word Embedding:

تضمين الكلمات هو الطريقة الحديثة لتمثيل الكلمات كمتجهات[5]. الهدف من تضمين الكلمات هو إعادة تعريف سمات الكلمات عالية الأبعاد إلى متجهات سمات منخفضة الأبعاد من خلال الحفاظ على التشابه السياقي في المجموعة. يتم استخدامها على نطاق واسع في نماذج التعلم العميق مثل الشبكات العصبية التلافيفية CNNs والشبكات العصبية المتكررة RNNs.

ويقصد بها، مصفوفة للكلمات والتي تقوم بتمثيل قيم خاصة لكل كلمة، لتحديد معناها، ولمعرفة مدى تقارب او ابتعاد هذه الكلمة عن باقي الكلمات.

فلو كان لدينا خمس كلمات مختلفة هي (الصبر، رجل، تفاحة، كلب، كتاب)

و نريد عمل علاقات رياضية بينهم ، فيمكن أن نقوم بطرح عدد من الأسئلة ، و الإجابة عنها لكل كلمة من الكلمات مثل:

- هل هذا الشيء حي؟ هل هو قادر على التحدث؟ هل هو ذكر؟
- هل هو ملموس؟ هل يمكن أكله؟ هل يمكن بيعه وشراءه؟ هل يتقدم في العمر؟

وهنا يمكن عمل مصفوفة بسيطة هكذا:

المعيار	الصبر	رجل	تفاحة	كلب	كتاب
هل هذا الشئ حي ؟					
هل قادر علي التحدث ؟					
هل هو ذكر أم انثي ؟					
هل هو ملموس ام شئ معنوي ؟					
هل يمكن أكله ؟					
هل يمكن بيعه و شراءه ؟					
هل يتقدم في العمر ؟					

ستكون الإجابات كالتالي:

المعيار	الصبر	رجل	تفاحة	كلب	كتاب
هل هذا الشئ حي ؟	لا	نعم	نعم	نعم	لا
هل قادر علي التحدث ؟	لا	نعم	لا	نعم	لا
هل هو ذكر ؟	نعم	نعم	لا	نعم	نعم
هل هو ملموس ؟	لا	نعم	نعم	نعم	نعم
هل يمكن أكله ؟	لا	لا	نعم	لا	لا
هل يمكن بيعه و شراءه ؟	لا	لا	نعم	نعم	نعم
هل يتقدم في العمر ؟	لا	نعم	نعم	نعم	لا

ماذا عن الأسئلة التي ليست لها قيمة نعم/لا، بل نسبة معينة، مثلا سؤال: هل هو مهم للإنسان، فقيمة الصبر تختلف عن الكتاب عن التفاحة وهكذا.

وتضمين الكلمات، معتمدة على هذا الأساس، ولكن على مستوى أكبر، فمتوسط المكتبات تتيح لنا قيمة 300 رقم يقوم بوصف كل كلمة بشكل دقيق تماما وتستخدم هذه الأرقام للتعرف على المعنى التقريبي للكلمة المتداولة، وأيضا للمقارنة بين الكلمات، ولمعرفة مدى اقتراب كلمة تفاح من كلمة برتقال، ومدى ابتعاد كل منهما عن كلمة الصبر.

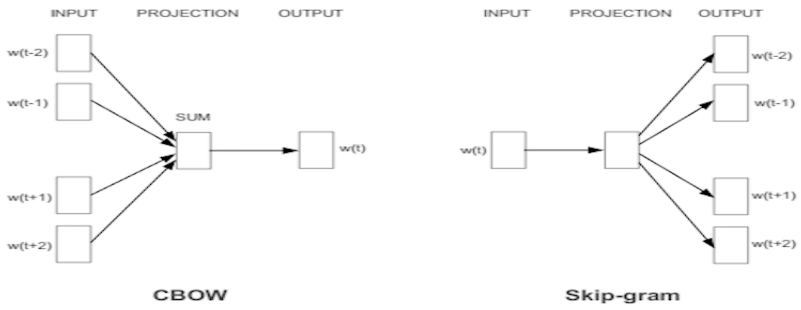
وإنشاء تضمين للكلمات word embedding لا يتم بهذه الطريقة القديمة[8] ولكن يعتمد على تدريب النموذج لمعرفة المعنى التقريبي لكل كلمة بشكل دقيق بناء على بيانات من ملايين الكلمات.

يعد كل من Word2Vec و Glove النموذجين الشائعين لإنشاء تضمين كلمة للنص. تأخذ هذه النماذج مجموعة نصية كمدخلات وتنتج متجهات الكلمات كإخراج.

Glove وهو نموذج أصدره باحثو ستانفورد، على الرغم من أنه يحسن نموذج Word2Vec من خلال مراعاة إحصائيات التكرار الإجمالية للكلمات بالإضافة إلى السياق المحلي ، إلا أنه لا يزال يعاني من نفس المشكلات مثل Word2Vec.

6-1نموذج Word2Vec (Word to Vector):

هي عبارة عن شبكة عصبية من طبقتين والتي تقوم بمعالجة النصوص[4]، يكون المدخل لها هو النص text corpus أما الناتج فهي كمية من المصفوفات الخاصة بالميزات للنص.



الشكل -2-نموذج Word2Vec

فهي ببساطة شبكة عصبية، يتم تدريبها علي أساس تضمين الكلمات، وهدفها حساب مدى أهمية وقيمة كل كلمة في الجملة، ومن ثم، نقوم باستنتاج الكلمة الباقية.

والمهمة الأساسية لأداة word2vec هي عمل تجميع للمصفوفات للكلمات المتشابهة والمتماثلة والمرتبطة معا، وهو ما يتم عبر التشابهات الرياضية لكل كلمة.

وهذه التشابهات والتناظرات تشبه (رجل –صبي) == (امرأة – فتاة)

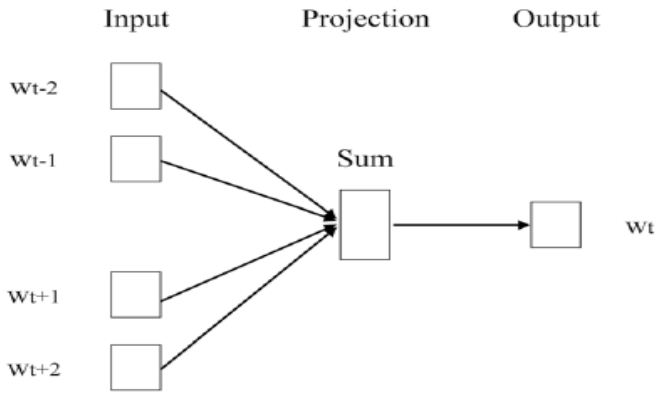
وأيضا أداة word2vec حينما تأخذ كمية كبيرة من البيانات، فلديها القدرة على توقع معاني الكلمات، بناء على موقعها وسياقها.

يتكون نموذج Word2Vec من وحدة معالجة مسبقة، ونموذج شبكة عصبية ضحلة يسمى Continuous Bag of Words ونموذج آخر للشبكة العصبية الضحلة يسمى skip–gram، تستخدم هذه النماذج على نطاق واسع لجميع مشاكل NLP الأخرى.

تم إنشاء هذا النموذج بواسطة Google في عام 2013 وهو نموذج قائم على التعلم العميق التنبؤي لحساب وتوليد تمثيلات متجهة كثيفة عالية الجودة وموزعة ومستمرة للكلمات، والتي تلتقط التشابه السياقي والدلالات. بشكل أساسي، هذه نماذج غير خاضعة للإشراف يمكنها أن تستوعب مجموعة نصية ضخمة، وتخلق مفردات للكلمات المحتملة.

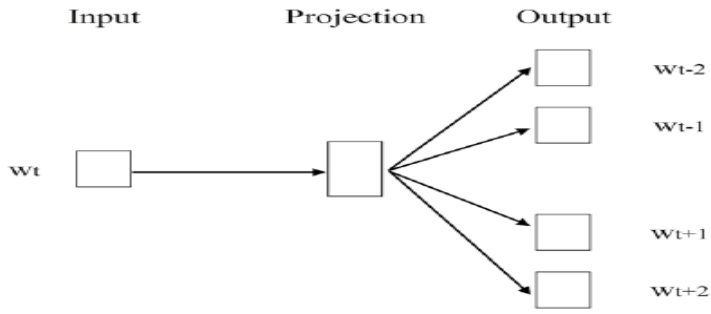
- **بنية CBOW (Continuous Bag of Words) [4]:**

تحاول بنية نموذج CBOW التنبؤ بالكلمة الحالية (الكلمة المركزية) بناء على كلمات السياق (الكلمات المحيطة).



الشكل-3-بنية CBOW

- **بنية Skip Gram [4]:** عادة ما تحاول بنية نموذج Skip-gram تحقيق عكس ما يفعله نموذج CBOW. إنه يحاول التنبؤ بكلمات السياق المصدر (الكلمات المحيطة) بالنظر إلى الكلمة المستهدفة (الكلمة المركزية).



الشكل-4-بنية Skip-gram

لتوليد متجهات الكلمات في Python، فإن المكتبات المطلوبة هي nltk و gensim.

يقوم النص البرمجي التالي بقياس التشابه بين كلمتين من خلال توليد متجهات word vectors باستخدام Word2Vec من خلال نموذجي CBOW و skip-gram باستخدام المكتبة gensim:

```
pip install nltk
pip install gensim
```

```
# Create CBOW model
model1 = gensim.models.Word2Vec(data, min_count = 1,
                                size = 100, window = 5)

# Print results
print("Cosine similarity between 'alice' " +
               "and 'wonderland' - CBOW : ",
    model1.similarity('alice', 'wonderland'))

print("Cosine similarity between 'alice' " +
                 "and 'machines' - CBOW : ",
      model1.similarity('alice', 'machines'))

# Create Skip Gram model
model2 = gensim.models.Word2Vec(data, min_count = 1, size = 100,
                                             window = 5, sg = 1)

# Print results
print("Cosine similarity between 'alice' " +
          "and 'wonderland' - Skip Gram : ",
    model2.similarity('alice', 'wonderland'))

print("Cosine similarity between 'alice' " +
            "and 'machines' - Skip Gram : ",
      model2.similarity('alice', 'machines'))
```

Output :

```
Cosine similarity between 'alice' and 'wonderland' - CBOW :  0.999249298413
Cosine similarity between 'alice' and 'machines' - CBOW :  0.974911910445
Cosine similarity between 'alice' and 'wonderland' - Skip Gram :  0.885471373104
Cosine similarity between 'alice' and 'machines' - Skip Gram :  0.856892599521
```

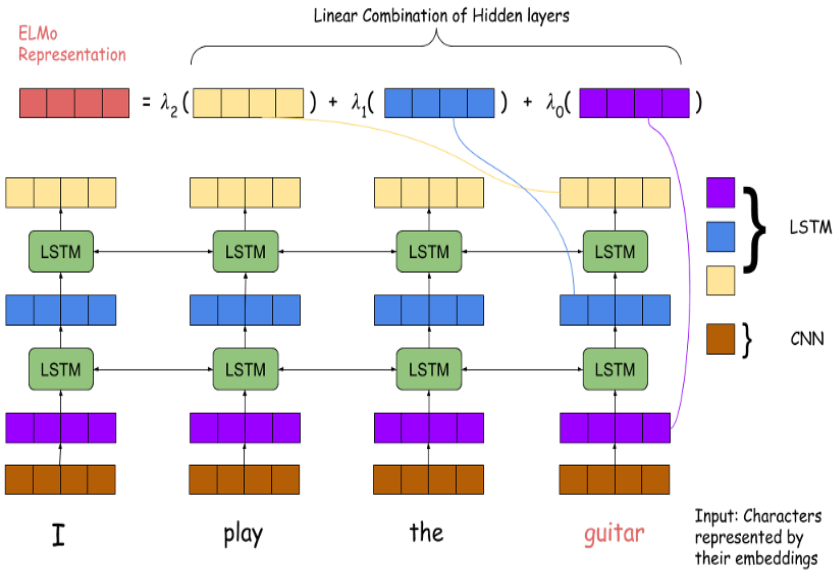
تظهر النتائج أن CBOW أعلى دقة من Skip-gram وذلك لأنها تحاول التنبؤ بالكلمة الحالية بناء على كلمات السياق (الكلمات المحيطة)، مما يعزز من قدرتها على إيجاد أقرب تشابه للكلمة مع المحيط، أو باختصار لأن نموذج CBOW يعتبر الكلمة التي نبحث عنها كلمة مركزية للبحث، وهذا على عكس Skip-gram التي تحاول التنبؤ بكلمات السياق المصدر بمعنى أنها تبحث عن الكلمات المحيطة

وتتعامل معها ككلمات مصدر أو مركزية، ثم تحاول معرفة موضع الكلمة التي نبحث عنها أو الكلمة الهدف.

6-2 نموذج ELMO (Embeddings from Language Models):

هي طريقة جديدة لتمثيل الكلمات كمتجهات أو التضمين(Embeddings) [6] ، يتم أخذ الرموز المميزة على مستوى الأحرف كمدخلات إلى LSTM ثنائي الاتجاه والذي ينتج Embeddings على مستوى الكلمة على عكس كلمة (Embeddings) التي أنتجتها مناهج وأساليب قديمة مثل "Bag of Words"، وأساليب المتجهات السابقة مثل Word2Vec وGloVe ، فإن تضمين Elmo حساس جداً للسياق ، وينتج تمثيلات مختلفة للكلمات التي تشترك في نفس التهجئة ولكن لها معاني مختلفة (المرادفات) مثل "عين "و "عين الانسان" و"عين الماء" الخ، وبعد تضمين الأحرف الموجودة في الكلمات أو الجمل بواسطة LSTM يتم إدخالها إلى نموذج CNN.

يتكون ELMO من ثلاث وحدات بشكل أساسي[6]،[7] -(1) الشبكات العصبية التلافيفية (CNN)، (2) الذاكرة طويلة المدى ثنائية الاتجاه (LSTM)، و (3) التضمينEmbeddings .



الشكل-5-نموذج ELMO

✓ **دخل النموذج:**

يعتمد الإدخال في ELMO على الأحرف فقط. نقوم بتحويل كل حرف إلى متجه باستخدام عمليات دمج الأحرف، والتي يتم تمريرها بعد ذلك إلى طبقات CNN. يمكننا تضمين الأحرف (Character Embeddings) من تجاوز عيوب النماذج السابقة. أولاً، قد يلتقط تضمين الأحرف أدق تفاصيل اللغة التي قد يفتقدها تضمين الكلمات (Word Embeddings). ثانيًا، يمكننا تمثيل الكلمات خارج المفردات بشكل فعال، وهذه قفزة هائلة إلى الأمام.

✓ **دور طبقة CNN:**

تأخذ طبقات CNN تضمين الأحرف للكلمة التي تقوم بمعالجتها كمدخلات وتحسب تمثيلات أكثر قوة من خلال التقاط ميزات n-gram. تحسب طبقات CNN بشكل أساسي الميزات من مجموعة الرموز المقدمة إليها، وبالتالي تستخرج بعض المعلومات

الإحصائية التي تمثل عدد تكرار الكلمة داخل السياق , والزمانية والمكانية ومقصود بهم ، معرفة موضع ذكر الكلمة , لان ELMO تعمل على فهم الجملة من السياق فهي تنظر إلى الجملة بأكملها قبل تعيين تضمين كل كلمة فيها، بمعنى كلمة عين كما وضحنا أعلاه يكون لها أكثر من متجه بحسب مكانها أو موقعها من السياق التي جاءت فيه والسبب في ذلك كما ذكرنا أن ELMO تعمل على فهم السياق أو معاني الكلمات بحسب ورودها ، وليس مجرد إنشاء متجه للكلمة مهما كان موضعها ، كما هو الحال مع word2vec ، ثم يتم تمرير هذا الإخراج إلى طبقات LSTM ثنائية الاتجاه. بعد تضمين الحروف الموجودة في الكلمات أو الجمل بواسطة LSTM يتم ادخالها إلى نموذجCNN.

- ✓ **دور طبقة الذاكرة ثنائية الاتجاه طويلة المدى LSTM:**

نحن نعلم أن LSTM، في جوهرها، تحافظ على المعلومات حول المدخلات التي شاهدتها بالفعل. هذا يجعله مفيدًا جدًا لمهام نمذجة التسلسل حيث يكون تذكر السياق السابق أمرًا حيويًا. بشكل عام، كانت LSTM أحادية الاتجاه، أي أنهم يأخذون بعين الاعتبار المعلومات من الماضي فقط للتنبؤ بالنتيجة.

من ناحية أخرى، تنظر LSTM ثنائية الاتجاه إلى المدخلات من كلا الطرفين، أحدهما من الماضي والآخر من المستقبل لعمل توقع. يمكنها القيام بذلك من خلال التفكير في اثنين من LSTM، أحدهما يمتد من الماضي إلى المستقبل والآخر يمتد من المستقبل إلى الماضي. من خلال ربط الحالات المخفية من كلا LSTMs، يمكنه الاحتفاظ بمعلومات حول التسلسل بأكمله في أي وقت.

إذن، على سبيل المثال، إليك ما ستعالجه LSTM للأمام والخلف:

Forward LSTM: She went to play ...
Backward LSTM: ... with her bat and ball.

كما ترى، فإن LSTM ثنائي الاتجاه سوف يتفوق في التنبؤ بالكلمة المفقودة من خلال فهم الماضي والمستقبل.

- ✓ **طبقة الخرج (Embedding/Output):**

هنا ينفصل نموذج ELMO عن البنى السابقة. في حين أن البنى السابقة استخدمت للتو الحالة المخفية النهائية لنموذج LSTM كتمثيل للكلمة، فإن ELMO تعتبر مزيجًا خطيًا من جميع الحالات المخفية لـ LSTM. تعتبر أوزان حالات LSTM المخفية خاصة بالمهمة. هذا يعني أنه بناءً على الغرض من التمثيلات، سيعطي نموذج ELMO أوزانًا مختلفة لكل طبقة. هذا مفيد للغاية لأنه، تلتقط الطبقات المختلفة خصائص مختلفة للغة.

في تقييماتهم الجوهرية، أظهروا أن حالات LSTM ذات المستوى الأعلى تلتقط الجوانب المعتمدة على السياق لمعنى الكلمة، في حين أن الحالات ذات المستوى الأدنى نموذج لجوانب بناء الجملة.

يتيح تعريض كل هذه الطبقات في نفس الوقت للنموذج اختيار التمثيلات التي يجدها أكثر فائدة للمهمة قيد البحث. على سبيل المثال، قد تكون الطبقات ذات المستوى الأعلى أكثر فائدة لمهام توضيح معنى الكلمة. في المقابل، قد تكون الطبقات ذات المستوى الأدنى أكثر فائدة لمهام مثل وضع علامات على جزء من الكلام.

المثال التالي يبين دقة فهم ELMO للمعاني حيث نقوم بمقارنة لإيجاد التشابه بين جملتين:

```
[ ]  1 result_embed_A = elmo_embeddings(["How are you my friend"])
```

```
[ ]  1 result_embed_B = elmo_embeddings(["This reference is good"])
```

```
[ ]  1 result_embed_C = elmo_embeddings(["how is your work"])
```

```
[ ]  1 result_embed_D = elmo_embeddings(["good reference"])
```

```
[ ]  1 result_embed_F = elmo_embeddings(["These books are helpful"])
```

```
[ ]  1 simsA_B = cosine_similarity(result_embed_A, result_embed_B)
     2 simsA_C = cosine_similarity(result_embed_A, result_embed_C)
     3 simsB_C = cosine_similarity(result_embed_B, result_embed_C)
     4 simsB_D = cosine_similarity(result_embed_B, result_embed_D)
     5 simsB_F = cosine_similarity(result_embed_B, result_embed_F)
```

```
1 print(["How are you my friend","vs" ,"This reference is good  "] ,simsA_B)
2 print(["How are you my friend","vs" ,"how is your work        "] ,simsA_C)
3 print(["This reference is good","vs" ,"how is your work       "] ,simsB_C)
4 print(["This reference is good","vs" ,"good reference         "] ,simsB_D)
5 print(["This reference is good","vs" ,"These books are helpful"] ,simsB_F)
```

```
['How are you my friend', 'vs', 'This reference is good  '] [[0.4756319]]
['How are you my friend', 'vs', 'how is your work        '] [[0.7792548]]
['This reference is good', 'vs', 'how is your work       '] [[0.50702286]]
['This reference is good', 'vs', 'good reference         '] [[0.64696056]]
['This reference is good', 'vs', 'These books are helpful'] [[0.6244303]]
```

نجد من خلال المقارنات أن ELMO تختلف عن النماذج السابقة مثل Word2vec التي كانت تستوجب حذف بعض الكلمات أثناء المعالجة الأولية للجمل، فبالنسبة لـ ELMO كل الحروف والكلمات ذات أهمية.

نلاحظ في آخر مقارنة رغم عدم وجود كلمات مشتركة لكن المعنى واحد فإن ELMO أعطت درجة تشابه 0.62 وكذلك مقارنة الجملتين 'How are you my friend' و 'how is your work' كانت درجة التشابه 0.77، وذلك يوضح طريقة ELMO في التعامل مع الجمل رغم اختلاف السياق أو طريقة السرد.

تظهر النتائج التجريبية قوة نموذج ELMO [6]. يمكن لـ ELMO أن تتحسن بشكل كبير مع أنواع مختلفة من البيانات وتساهم في حل العديد من مشكلات معالجة اللغة الطبيعية NLP، تظهر ELMO في شكل بنى أو نماذج مستقلة عن بعضها البعض،

بسبب اختلاف نوعية المشكلات أو البيانات التي تدربت عليها مثل نماذج تحليل المشاعر ونماذج توليد الكلام والإجابة عن الأسئلة وأمثلة أخرى، وتحقق هذه النماذج أداء متطوراً عن الطرق السابقة في التحديات المختلفة.

7- خاتمة:

استعرضنا في هذا البحث بعض تقنيات تضمين الكلمات لتمثيل الكلمات كمتجهات رقمية يمكن لخوارزميات تعلم الآلي معالجتها.

على مدى السنوات العديدة الماضية، تم دفع مجال معالجة اللغة الطبيعية إلى الأمام من خلال الانفجار في استخدام نماذج التعلم العميق. قدمنا في هذه المقالة دراسة لطرق تضمين الكلمات ونظرة عامة سريعة على نموذج ELMO إحدى بنى التعلم العميق.

8-المراجع:

[1] K. S. Jones, "Natural language processing: A historical review," in Current Issues in Computational Linguistics: In Honour of Don Walker. Dordrecht, the Netherlands: Springer, 1994, pp. 3–16.

[2] E. D. Liddy, "Natural language processing," in Encyclopedia of Library and Information Science, 2nd ed. New York, NY, USA: Marcel Decker, Inc., 2001.

[3] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537 ,2011.

[4] Wang, S., Zhou, W., & Jiang, C. A survey of word Embeddings based on deep learning. Computing, 2019.

[5] Turian J, Ratinov L, Bengio Y Word representations: a simple and general method for semi supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics, 2010, pp 384–394

[6]. Matt G., Joel G., Mark N., AllenNLP: A Deep Semantic Natural Language Processing Platform, 2018.

[7]. Peters M. E., Neumann M., Uyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L. Deep contextualized word representations, 2018.

[8].Chen X, Lei X, Liu Z, Sun M, Luan H Joint learning of character and word Embeddings. In: International conference on artificial intelligence, 2015.

[9]. Devlin J, Chang MW, Lee K, Toutanova K Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

[10]. Bian J, Gao B, Liu TY Knowledge-powered deep learning for word embedding. In: Joint European conference on machine learning and knowledge discovery in databases, Springer, 2014, pp 132-148.

[11]. Qi L., Matt J., Phil B., A Survey on Contextual Embeddings, 2020.