الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة المعلوماتية
قسم الذكاء الصنعي

## مشروع مادة معالجة اللغات الطبيعية

## السنة الخامسة - اختصاص ذكاء صنعي

## بعنوان

# Semantic matching

## مطابقة بين النصوص حسب المعنى

إعداد الطلاب:

هراير ديربيدروسيان
ماريان ديب
أحمد أبو محمود
كنان أبو زين الدين
حيدر الصوص

كانون الأول 2023

# مقدمة

هدفنا من هذه الدراسة تقديم مقارنة بين الطرق التي اتبعناها من أجل بناء نظام مطابقة بين نصوص حسب المعنى **sematnitc matching** وبالتالي فإن هدفنا هو حساب **sentence embeddings** لكل جملة وليس مطابقة بين النصوص حسب الكلمات **lexical matching.**

قمنا باختيار مسابقة **Quora Question Pairs** على **kaggle** لعمل تقييم للطرق المتبعة ولم نقم بأي عمليات تنظيف او معالجة للبيانات وذلك لقياس تأثير الطرق المتبعة بشكل معزول قدر الإمكان عن أية عوامل أخرى .

**بيانات التدريب** على **kaggle** تحتوي في كل سطر على سؤالين والمطلوب هو معرفة هل هذين السؤالين متشابهين أم لا. **بيانات الاختبار** لا تحتوي على **labels** وبالتالي فان عملية التقييم على بيانات الاختبار تتم على موقع **kaggle** فقط.

**من أجل كل طريقة من الطرق المتبعة قمنا بحساب الـ sentence embeddings لكل سؤال ثم قمنا بحساب الـ features التالية :**

1. شعاع الفرق بالقيمة المطلقة بين الشعاعين السابقين.
2. cosine similarity
3. المسافة الاقليدية .

بعد حساب الـ **features** السابقة نقوم بتدريب نموذج **logsitic regression** باستخدامها. قمنا باستخدام **log loss** مطبق على الفرق بين احتمال تشابه السؤالين **y_pred** وتشابههما **y_train** كمعيار للتقييم لانه مستخدم في هذه المسابقة على **kaggle.**

# Unweighted Average of Word2vec Embeddings

في هذه الطريقة قمنا بتحميل **pre-trained model** لنموذج **word2vec** وقمنا بحساب **sentence embeddings** عن طريق اخذ المتوسط الحسابي الغير موزون لكلمات هذه الجملة، قمنا بهذه العملية للجملتين في كل سطر، ثم قمنا بحساب الـ **features** المذكورة سابقا.

# Unweighted Average of Siamese CBOW Embeddings

في هذه الطريقة قمنا بتطبيق الورقة البحثية **Kentar et la. 2016** حيث كانت الفكرة الاساسية من هذه الورقة الاتي **Kentar et la. p.2:**

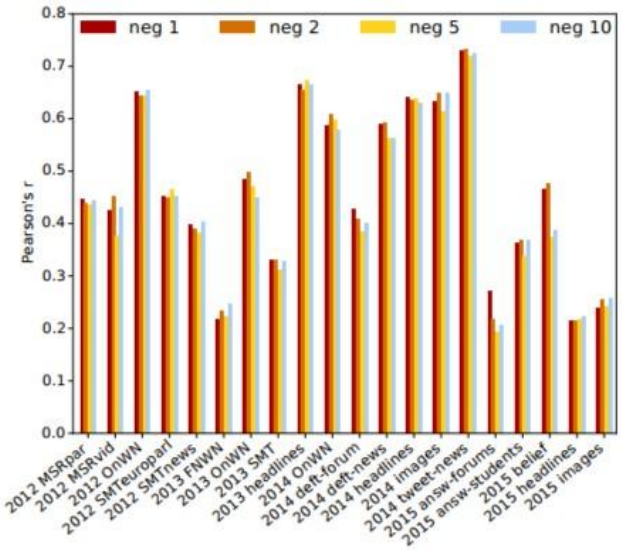
**"The main research question we address is whether directly optimizing word embeddings for the task of being averaged to produce sentence embeddings leads to word embeddings that are better suited for this task than word2vec does"**

هذه الطريقة مستلهمة من طريقة **Mikolov et al. 2013** والتي تنص على توقع الكلمات الاكثر احتمالا بناء على السياق (وهذا سبب التسمية **CBOW**) ولكن هنا نريد تطبيقها على مستوى الجمل حيث نريد للجمل المجاورة لجملة ما ان تكون لها احتمال اكبر.

## بيانات التدريب

في الورقة البحثية تم استخدام **Toronto Book Corpus** والذي يحوي حوالي **76 مليون جملة** ولكن قمنا باستخدام **Brown Corpus** والذي يحتوي **57 الف جملة** لضيق الوقت ولتواضع الامكانيات.

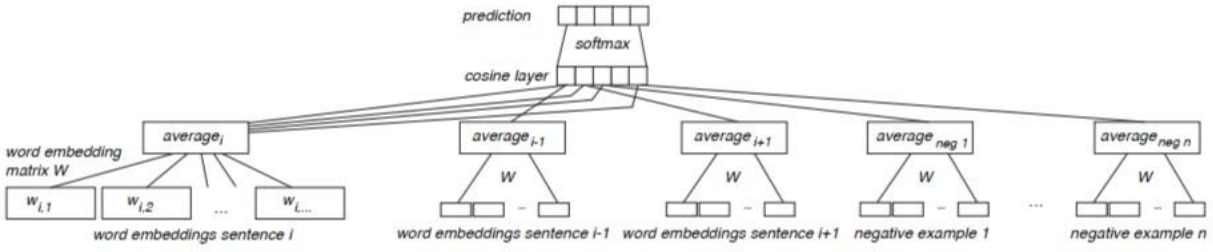
كل عينة في بيانات التدريب تحوي على **خمس جمل**، الجملة الحالية والجملتين السابقة والتالية وجملتين عشوائتين والمخطط التالي يوضح مدى تاثير عدد الجمل العشوائية في كل عينة على اداء النموذج حيث تم التقييم على عدة بيانات تدريب:




وبناء على المخطط السابق تم اعتبار ان جملة او جملتين عشوائيات هو الخيار الافتراضي. لقد قمنا باستخدام جملتين عشوائيات وذلك لصغر حجم بيانات التدريب.

## بنية الشبكة

تم تصميم الشبكة بحيث يكون دخلها عدة جمل احدها هي الجملة الحالية والخرج هو احتمال ظهور كل جملة، والهدف هو تكبير احتمال ظهور الجمل المجاورة. وهذا يخدم الغرض الاساسي لهذه الشبكة المذكور سابقا.




**الطبقة الاولى:** نقوم بحساب الـ **average word embeddings** للكلمات المكونة لجملة ما وذلك من اجل كل جملة في العينة الواحدة.

**الطبقة الثانية:** نقوم بحساب **cosine similarity** بين الجملة الاصلية والجمل الاخرى بما في ذلك الجملة الاصلية.

**الطبقة الثالثة:** نقوم بادخال النتائج لتابع **softmax** وله معادلة من الشكل:

$$p_\theta(s_i, s_j) = \frac{e^{\cos(\mathbf{s_i^\theta}, \mathbf{s_j^\theta})}}{\sum_{s_{/} \in S} e^{\cos(\mathbf{s_i^\theta}, \mathbf{s_{/}^\theta})}},$$

بالحالة المثالية نقوم بحساب المقام على كل الجمل الموجودة بالبيانات ولكن لصعوبة الامر نقوم بحساب المقام على الجمل الموجودة في العينة الواحدة. وهذا ما تم اتباعه في الورقة البحثية.

**الطبقة الرابعة والاخيرة:** نقوم بحساب تابع الخسارة **loss** حيث له معادلة من الشكل:

$$L = -\sum_{s_j \in \{S^+ \cup S^-\}} p(s_i, s_j) \cdot \log(p_\theta(s_i, s_j))$$

حيث:

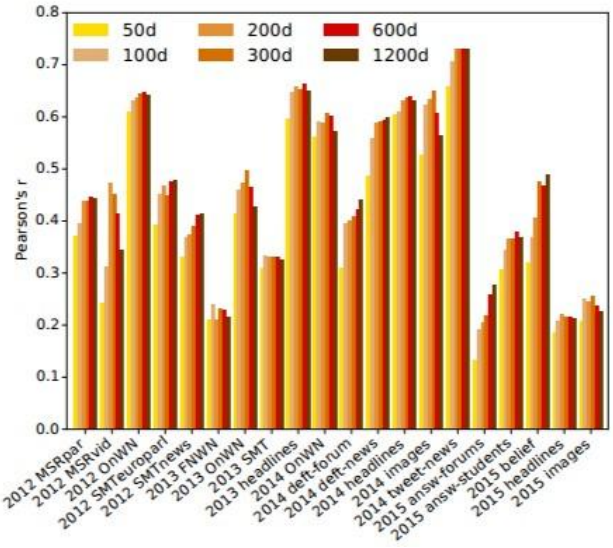
$$p(s_i, s_j) = \begin{cases} \frac{1}{|S^+|}, & \text{if } s_j \in S^+ \\ 0, & \text{if } s_j \in S^-. \end{cases}$$

وهو تابع يقوم بحساب احتمال ظهور جملة، حيث الجمل العشوائية لها احتمال ظهور يساوي **الصفر** والجملتين السابقة والتالية لها احتمال ظهور **0.5** بناء على المعادلة السابقة، لم يتم ذكر احتمال ظهور الجملة الحالية في الورقة البحثية ولذلك قمنا باعتباره **صفر** لاننا نريد ان نتوقع الجملتين السابقة والتالية وليس الحالية.

## التدريب

البرامترات الوحيدة القابلة للتدريب في هذا النموذج هي الـ **word embeddings** ولها **300** بعد **dimensions** حيث قمنا بتهيئتها باستخدام تابع التوزيع الطبيعي **normal distribution** حيث قمنا باختيار قيمة متوسطة **mean = 0.0** وايضا انحراف معياري **standarard deviation = 0.01** وتم استخدام خوارزمية **stochastic gradient descent** حيث معدل التعلم **learning rate = 0.0001** وحجم كل دفعة تدريب **batch size = 100** وتم التدريب في حقبة واحدة فقط **epoch = 1.** تم اتباع طريقة الورقة البحثية بالبرامترات السابقة.

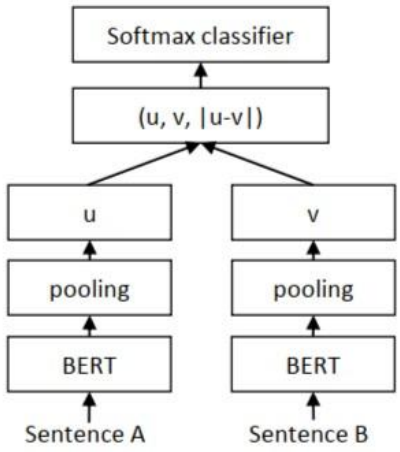
المخطط الاتي يوضح تاثير عدد ابعاد الـ **word embeddings** على اداء النموذج حيث تم الاختبار على عدة بيانات:




بعد انتهاء التدريب نقوم بحساب المتوسط الحسابي الغير موزون للاوزان الناتجة عن التدريب **unweighted average of siamese CBOW embeddings** وذلك من اجل كل جملة لدينا ونقوم بحساب الـ **features** المذكورة سابقا.

# Sentence-BERT Embeddings

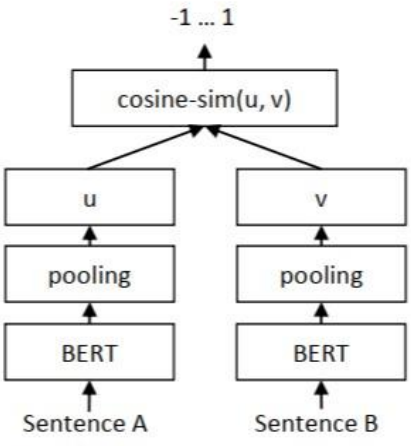
المشكلة الاساسية التي حاول **Reimers et la. 2019** حلها هي ان **BERT** يتطلب كثيرا من الوقت لايجاد اقرب جملتين متشابهتين من بين عدة جمل، وذلك لان **BERT** يقوم بالصاق **concatinate** للجملتين ببعضهما وبعدها يقوم بجساب الـ **consine similarity** وهذه العملية مكلفة حيث يجب أن تقوم بها من اجل كل جملتين في البيانات. لذا تم اقتراح البنية الاتية:




نقوم بادخال جملتين لـ **BERT** ومن اجل كل جملة نقوم بتطبيق **pooling** ثم حساب شعاع يمثلها، وثم نقوم بحساب الفرق بين الشعاعين بالقيمة المطلقة ثم يتم تطبيق **softmax** حسب المعادلة:

$$o = \text{softmax}(W_t(u, v, |u - v|))$$

حيث **W** هو وزن يتم تدريبه، والمراد هو تصنيف جملتين اذا كانتا متناقضتين او متلازمتين **entailment** او حياديتين، بعد التدريب نحص على الشبكة الاتية، حيث يكون الدخل جملتين والخرج هو **cosine similarity** او مقدار التشابه بينهما:



لم نقم بالتمعن بتفاصيل هذه الورقة نظرا لاننا قمنا باستخدام مكتبة **FastSentenceTransformer** الغير سريعة :) وبالطبع قمنا بحساب الـ **features** المذكورة سابقا من اجل بيانات التدريب والاختبار لـ **kaggle.**

## SIF Weighted Average of Glove Embeddings

في هذه الطريقة قمنا بتطبيق الورقة البحثية **Arora et la. 2016**

حيث سبب اختيار SIF :

**"smooth inverse frequency (SIF) This method achieves significantly better performance than the unweighted average on a variety of textual similarity tasks, and on most of these tasks even beats some sophisticated supervised methods tested in (Wieting et al., 2016)"**

هي شبيه ل TF-IDF :

**"this SIF reweighting is highly reminiscent of TF-IDF reweighting from information retrieval (Sparck Jones, 1972; Robertson, 2004)"**

حيث الورقة البحثية تركز على **العلاقة** التالية كثيراً لأنها محور وجود **SIF** : ( إعادة الأوزان للمتجهات )

**discourse vector** $$\tilde{c}_s = \sum_{w \in s} \frac{a}{p(w) + a} v_w$$

## بيانات التدريب

في الورقة البحثية قاموا بالتجارب على ثلاث أنواع من النماذج **( PSL - GLOVE - SN )** واستخدام أربع أنواع من dataset من أجل التواتر لكل كلمة **(enwiki- poliblogs - commoncrawl - text8 )** .

حيث قمنا بتحميل **pre-trained model** لنموذج **Glove** يحتوي على الكلمات مع المتجهات , كل كلمة **300** بعد , و تحميل ملف **enwiki** يحتوي على الكلمات مع الأوزان لكل كلمة من أجل تطبيقه في عملنا .

## بنية الشبكة

**Algorithm 1** Sentence Embedding

**Input:** Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter $a$ and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
**Output:** Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
1: **for all** sentence $s$ in $\mathcal{S}$ **do**
2: $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix $X$ whose columns are $\{v_s : s \in \mathcal{S}\}$, and let $u$ be its first singular vector
5: **for all** sentence $s$ in $\mathcal{S}$ **do**
6: $v_s \leftarrow v_s - uu^\top v_s$
7: **end for**

حيث يقوم **بأربع مراحل** :

**المرحلة الأولى** : إعطاء الأوزان لكل كلمات في **GLOVE** ثم تعديل الأوزان بناءا على بارامتر ال a .

**المرحلة الثانية** : إعطاء الأوزان للكلمات في الجملة المدخلة .

**المرحلة الثالثة** :حساب متوسط الأوزان للمتجهات الكلمة في الجملة المدخلة

**المرحلة الرابعة** : إزالة **common component removal** .

الجمل المدخلة في عملنا هي بيانات التدريب ( الأسئلة ) .

وأخيراً من أجل كل جملة مدخلة , يكون لدينا **SIF Sentence Embeddings** ومنها نحسب **الـ features .**

## النتائج :

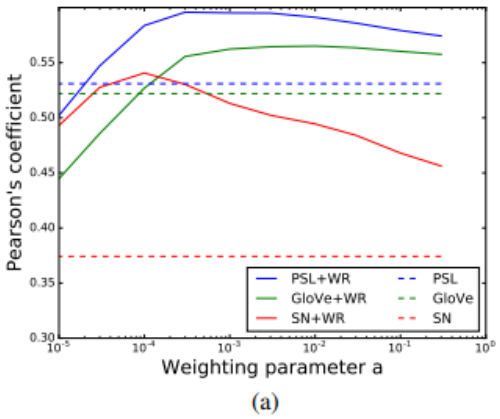
في الورقة البحثية :

وجدت أن قيمة البارامتر **a = 10^-3** يعطي أفضل اداءاً من **a = 10^-4** على الكلمات **Glove** بإستخدام **enwiki**

حيث نستخدم a من أجل تعديل قيمة ال **w**

**a/(a + p(w))**

تم اختيار **a = 10^-3** في عملنا



(a)

# Sent2vec Embeddings

يتم تدريب Sent2vec على الجمل بإستخدام متجهات word n-grams .

أي بالمراحل التالية :

تقسيم الجمل الى الكلمات , إنشاء متجهات n-grams لكلمات , حساب متوسط المتجهات , ثم تدريب النموذج .

في هذه الطريقة قمنا بالاستعانة بنموذج مدرب مسبقاً ل **sent2vec** حيث يحتوي على مليون وخمسين الف كلمة ولكل كلمة يتمثل لها 600 متجه ,ثم قمنا بحساب **sentence embeddings** لكلجمل وبعد ذلك بحساب الـ **features** بين كل جملتين المذكورة سابقا.

في الورقة البحثية **Pagliardini et la. 2018** تم اتباع نهج **Mikolov et la. 2013** ولكن على مستوى الجمل

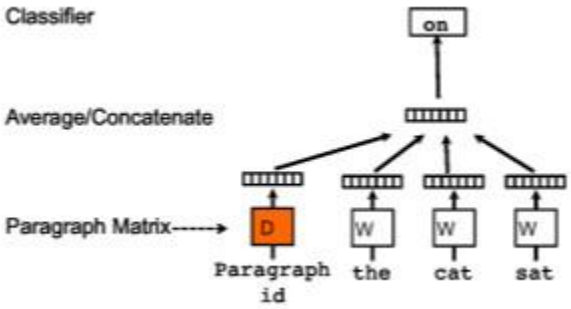
# Doc2vec Embeddings

في هذه الطريقة قمنا بتحميل مكتبة Doc2vec ,وتدريبها باستخدام كلمات من Brown Corpus وهو مجموعة قليلة من النصوص مقارنة مع الداتا , حيث يحتوي Brown Corpus على 57 ألف كلمة غير مكررة , وقد تم اختياره لضيق الوقت وتواضع الإمكانيات .

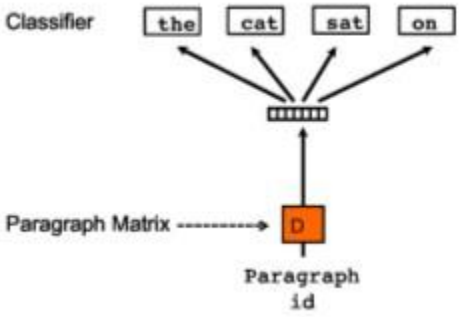
بعد التدريب , يتم تخزين نموذج Doc2vec كلمات ومتجهاتها .

ثم قمنا بحساب **sentence embeddings** لكل ( جمل بشكل **Tokens** ) وبعد ذلك بحساب الـ **features** بين كل جملتين المذكورة سابقا.

هذه الفكرة ماخوذة من **Quoc et la. 2014** حيث قاموا باتباع نهج **Mikolov et la. 2013**، حيث قاموا بتصميم شبكة تشبه شبكة **CBOW** وقاموا ولكل باضافة **paragraph id** والذي يعبر عن الـ **paragraph** ككل:




وقاموا باتباع طريقة **skip-gram** ولكن الخرج المراد هو **paragraph id**:



باتباع هاتين الطريقتين ينتج لدينا **word embeddings** تحمل معلومات السياق الموجودة فيه وبالتالي **sentence embeddings** افضل

# جدول المقارنة

| Model Name | Train Log Loss | Test Log Loss |
|---|---|---|
| Unweighted Average of Word2vec Embeddings | 12.32 | 7.55 |
| Unweighted Average of Siamese CBOW Embeddings | 13.45 | 8.07 |
| Sentence-BERT Embeddings | 8.49 | 7.1 |
| SIF Weighted Average of Glove Embeddings | 11.84 | 8.03 |
| Sent2vec Embeddings | 15.02 | 11.03 |
| Doc2vec Embeddings | 13.53 | 9.47 |

بالنسبة للحقول الفارغة فسيتم الإعلان عنها يوم المقابلة ان شاء الله نظراً لاننا لم نستطع القيام بالاختبار قبل تسليم التقرير .

# المراجع

**[Arora et la. 2016] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.**

**[Kentar et la. 2016] Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. arXiv e-prints, 1606.04640.**

**[Mikolov et la. 2013] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector spaces. arXiv e-prints, 1301.3781.**

**[Pagliardini et la. 2018] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. arXiv e-prints, 1703.02507.**

**[Quoc et la. 2014] Quoc Le, and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. arXiv e-prints, 1405.4053.**

**[Reimers et la. 2019] Nils Reimers, and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv e-prints, 1908.10084.**