Dependency, skip-grams, association tensors and machine learning for sentence completion

Jean Argouarc'h
EHESS / Paris
Neurospin / Saclay
jr.argouarch@gmail.com

Abstract

The aim of this work is to get the best semantic representation of words, using sentence completion as a benchmark for its evaluation. We develop a semantic language model integrating a tensor to represent word co-occurrence statistics, an algorithm to transform dependency and skip-gram word-word relations into a sentence relevance vector and we finally apply machine learning classification to improve the accuracy of the model.
This model clearly highlights the difference of performances between statistical semantic models of the litterature, including those using hyperparameters inspired by neural network models. It shows the complementarity of dependency and linear context relations and gives a precise measurement of their respective contribution.
A fully explicit model yields a performance of 77% of correct word in the sentence, enhanced up to 78.6% by machine learning classification, to be compared with a chance level of 20% (one out of five). Moreover we observe the presence of a bias in a well known sentence completion test set, and we propose a new set of 6000 sentences to the research community for further studies.

1 Introduction

Semantic representation of words based on the statistics of their context is widely used for many linguistic applications. The purpose of this work is to develop an optimized word representation by testing it on a sentence completion task, which has been used as a benchmark to test semantic natural language models for the last years (Zweig et al., 2012; Mikolov et al., 2013; Gubbins and Vlachos, 2013; Mirowski and Vlachos, 2015; Zhang et al., 2016; Woods, 2016).

We create two new integrated models exploiting dependency relations, skip-grams, 3D tensor representation of statistical data, algorithms to compute sentence relevance vectors, and machine learning classifiers. We apply them with several statistical semantic models to compare their efficiency.

After extracting statistics of dependency and skip-gram relations (section 3) from a limited corpus, we apply several models to transform these statistics into association tensors (section 4), we design two sets of relevance features and their associated algorithms to compute a sentence relevance vector (section 6), and we try several machine learning classifiers in this special situation where we have to classify sentences in a local subset of 5 sentences, instead of a global multiclass classification.

In a first approach we develop an algorithm until reaching the state-of-the-art accuracy on the Microsoft Research Sentence Completion Challenge (MR challenge). Suspecting that the results are influenced by the n-grams statistics used to generate the test sentences, we create a new set of sentences. We work on a second, simpler algorithm, and we confirm the bias of the former set.

On this new set of sentences we proceed to a detailed comparison of word-word relations based on linear context, dependency context, and their combination, and of the size of the context window. It shows quite different results concerning the efficiency of statistical semantic models.
Our main contributions are :

- a systematic use of labeled oriented dependency relations,
- a new approach of skip-grams enabling to improve their efficiency, and to measure the influence of their window width,
- two algorithms to compute and compare sentence relevance vectors,

Proceedings of the 14th Conference on Natural Language Processing (KONVENS 2018)
Vienna, Austria – September 19-21, 2018

- the application of machine learning classifiers in this particular classification situation,
- a detailed comparative analysis of performance of several association models, and of skip-gram vs dependency context.

2 Context

When developing natural language computer models, we need benchmarks to measure their efficiency. The most current tests rely on word similarity or word analogy, which have rather loose definitions and metrics. Sentence completion based on real sentences extracted from controlled corpora provide a very different and probably more powerfull semantic benchmark. The most used sentence completion set is the MR challenge (Zweig and Burges, 2012), in which a sentence with a missing word is proposed with five candidate words, the answer and four "impostors".

The semantic models used to solve sentence completion can be divided into two main categories. The first one represents words by vectors computed by neural networks, the second represents word frequency or word association by matrices or tensors. Both gather statistical data from a corpus, the former by machine learning, the latter by counting occurrences. Each word of the corpus is related to its neighbouring words, either through linear proximity or syntactic relation. The linear context may be any type of n-gram, skip-gram or bag-of-words. The syntactic context generally consists in dependency relations extracted by a parser. The words can be inflected words, or lemmas, after part-of-speech tagging, possibly limited to content words.

Using only linear contexts, Zweig et al. (2012) applied skip-grams, RNN, LSA methods and a linear combination of all three, to reach a success ratio of 52% on the MR challenge. Mikolov 2013 obtained 58.4% with a combination of RNN and skip-grams.

Dependency models appeared more recently, with the development of fast dependency parsers. Gubbins and Vlachos (2013) reached 50.0% with labeled dependency, Mirowski and Vlachos (2015) 53.5% with dependency RNN, Zhang et al. (2016) 60.7% with dependency context and a Long Short-Term Memory neural network, and Woods (2016) 61.4 % with a combination of dependency and n-gram contexts.

As far as we know, no intensive work has been done to compare the accuracy of linear and syntactic contexts and of their combination in the same conditions, and to measure the effect of the width of the window used to link a target word with its context, or the path length in the dependency tree of the sentence.

In the case of counting methods, used since the first LSA applications, the raw statistical data are generally transformed by smoothing or weighting into an association matrix or tensor, in order to solve biases towards rare words (Dunning (1993), Lowe (2001)). Applied to sentence completion they allow to compute and compare the likelihood of the five candidate words in relation with the rest of the sentence. It seems that machine learning has not yet been applied in combination with counting-based methods, contrary to neural networks which are inherently learning algorithms. Levy et al. (2015) have shown some convergence between these conceptually different methods, and particularly between the "hyperparameters" used to optimize neural networks and different types of frequency matrix weighting or smoothing.

As for the test sentences, we first tried the MR challenge set which provides a reference corpus of litterature and a set of 1040 sentences. It suggests to use half of the sentences for development, and the second half for test. Published results on the subject are not very detailed on this point, with the exception of Mirowski and Vlachos (2015), who explicitly followed the proposed protocol, and of Zhang et al. (2016) who mention the use of a complementary set of 4000 sentences as a development set to keep all the 1040 sentences as the test set. Machine learning classification methods generally apply more complex methods such as k-fold validation to secure the results (Hastie et al., 2008).

3 Dependency and skip-gram frequency tensor

To start building the model we gathered word-word dependency and skip-gram relations along the corpus. We represented any word as a pair (lemma, word class), and kept content words (nouns, verbs, adjectives, adverbs) and pronouns [1].

The dependency relations were provided by a dependency parser (Stanford CoreNLP [1], Manning et al. (2014)) which provides 3 types of dependency

[1] because several pronouns are proposed as candidate words in the MR challenge

[1] https://stanfordnlp.github.io/CoreNLP/

relations between content words: subject, object and modifier.

From a semantic point of view we were more interested in agent and patient relations than in subject and object. As the parser indicates passive auxiliaries, we considered the subject of a passive verb as its deep object or *patient* (pat), and its object as its deep subject or *agent* (agt). The *modifier* (mod) relation has a wide scope as it includes all the other relations between the word classes that we have selected, including for instance the indirect object relation. By using the word class of the two related words we are able to differentiate several cases.

We introduced another type of dependency relation, the *coordination* (cor) between two words connected by any coordinating conjunction, which can be extracted easily from the parser's output. So we finally get four dependency relations *agt, pat, mod, cor*.

For linear context we did not base our model on n-grams, as they do not allow to model precisely linear distances in a sentence, but word-word "skip-grams", by skipping over one or several words (Guthrie et al., 2006). We looked for unigrams, more precisely 0-skip-grams to 7-skip-grams (in our model a n-skip-gram is a relation between two words separated by n words in a sentence). We call them *sg0, sg1... sg7*.

In addition to direct dependencies we can also consider more distant dependencies along the dependency tree of the sentence, following for instance Pado and Lapata (2007). Direct dependencies having a path length of one in the tree, we took into account distances of 2 and 3 edges, and called 1-skip-dep (*sd1*) and 2-skip-dep (*sd2*) these indirect relations.

Example of one question of the MR Challenge.

We shall just be in time to have a little breakfast with him.
answer word *aw* = *breakfast*
candidate words *cw's* = {*elegance, garment, breakfast, basket, dog*}
set of direct relations = { *(have,cw,pat), (cw,little,mod), (cw,he,mod), (little,cw,sg0), (cw,he,sg0), (have,cw,sg1), (time,cw,sg2), (be,cw,sg3), (just,cw,sg4), (we,cw, sg5)* }

We finally gathered 14 types of dependency or skip-gram relations r_k between a predecessor word w_i and a successor word w_j which we represent by a triplet (w_i,w_j,r_k) with:

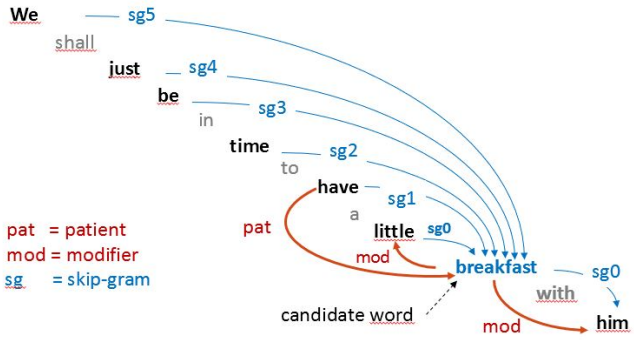

Figure 1: Example of sentence relations

$w_i, w_j \in$ V, V = [list of lemmas]
$r_k \in$ Rel, $k \in \{0...13\}$
Rel = {*agt, pat, mod, cor, sd1, sd2, sg0, ..., sg7*}
And we stored the number of occurrences of all these triplets gathered from the corpus in a frequency tensor F_{ijk}.

4 Association models

4.1 Maximum likelihood estimation

If we apply directly the frequency tensor without any further transformation, we get the maximum likelihood estimation (MLE), which we will keep as a reference to compare more sophisticated models. We evaluate directly the conditional probability $p(cw|w,r)$.

If $cw = w_i$ is a predecessor of $w = w_j$:
$MLE_{ijk} = p(cw|w,r) = F_{ijk} / \sum_i F_{ijk}$
and if it is a successor:
$MLE_{ijk} = p(cw|w,r) = F_{jik} / \sum_i F_{jik}$

4.2 Log-likelihood ratio

Dunning (1993) proposed the log-likelihood model to bypass the non-normality of text statistics, followed by Pado and Lapata (2007). The formulas are:

$q_k = F_{ijk} \qquad l_k = \sum_j F_{ijk} - q_k \qquad m_k = \sum_i F_{ijk} - q_k$
$n_k = \sum_{ij} F_{ijk} - (q_k + l_k + m_k)$
and with: $g(x) = x * log(x)$
$LLR_{ijk} = 2 * (g(q_k) + g(l_k) + g(m_k) + g(n_k)$
$- g(q_k + l_k) - g(q_k + m_k))$
$- g(l_k + n_k) - g(m_k + n_k)$
$+ g(q_k + l_k + m_k + n_k))$

4.3 PPMI

One of the most current models, which gives good results on several semantic tasks is the Pointwise Mutual Information, introduced by Church and Hanks (1990) for matrices, which has inspired a lot

of variations. Applying it to every matrix $F_{::k}$ we compute successively for each k:

$$sr_k = \sum_{i,j} F_{ijk} \quad sr_{i*k} = \frac{\sum_j F_{ijk}}{sr_k} \quad sr_{*jk} = \frac{\sum_i F_{ijk}}{sr_k}$$

$$mr_{ijk} = \frac{F_{ijk}}{sr_k} \qquad PMI_{ijk} = \log\left(\frac{mr_{ijk}}{sr_{i*k} \cdot sr_{*jk}}\right)$$

Generally negative values are replaced by zero to give the Positive PMI (PPMI). From this definition we will compare three PPMI variants which gave the best results in previous works.

4.4 Discounted PPMI

In word-document models, the PPMI is known to introduce a bias for rare words. Following Turney and Pantel (2010), Fyshe et al. (2015) and Woods (2016), we will apply the discounting factor proposed by Pantel and Lin (2002) :

$$fm_{ijk} = min(\sum_p F_{pjk}, \sum_p F_{ipk})$$

$$\delta_{ijk} = \frac{F_{ijk}}{F_{ijk}+1} \cdot \frac{fm_{ijk}}{fm_{ijk}+1}$$

To get the discounted ppmi:

$$PPMID_{ijk} = \delta_{ijk} \cdot PPMI_{ijk}$$

4.5 Shifted PPMI

Levy et al. (2015) proposed to extend the PPMI model to a "shifted" PMI with a parameter a inspired by RNN models, which we will call here PPMIS:

$$PPMIS_{ijk} = max(pmi_{ijk} - log(a), 0)$$

which gave them interesting results on several semantic tests of word similarity and analogy.

4.6 Smoothed PPMI

Levy et al. (2015) tried the context distribution smoothing, inspired by RNN works (Mikolov et al. (2013), Pennington et al. (2014)). They apply an exponent α to the columns j of the context matrix F_{ijk}, replacing the formulas of section 4.3:

$$sr_{*jk} = \frac{\sum_i F_{ijk}}{sr_k} \quad \text{by:} \quad sra_{*jk} = \frac{(\sum_i F_{ijk})^\alpha}{\sum_j (\sum_i F_{ijk})^\alpha}$$

In our case, the rows and the columns of the $F_{::k}$ matrix play a symetric role, so we must apply the same alpha exponentiation to the rows (experimentally confirmed), to get:

$$sra_{i*k} = \frac{(\sum_j F_{ijk})^\alpha}{\sum_i (\sum_j F_{ijk})^\alpha}$$

$$PPMIA_{ijk} = max(\log\left(\frac{mr_{ijk}}{sra_{i*k} \cdot sra_{*jk}}\right), 0)$$

5 Experimental data

5.1 Question sets

The MR challenge provides a set of 1040 questions, one question being a set of one sentence and five candidate words, the answer and four impostor words. For the generation of the set, an n-gram language model has been used to select the impostors, in two steps using bigram statistics, firstly to eliminate the 150 best candidates, secondly to select the 30 best remainders before a final hand-made selection (Zweig and Burges, 2012).

To make sure that the sentence generating process does not introduce any bias for the evaluation of the semantic models, we generated a new set of 6000 questions from 7 books of the same period of the 19th century, which are not in the training corpus. We have kept frequency constraints as the MR challenge, by eliminating words with frequency higher than 10^{-3} (one occurrence out of 1000) and with frequency lower than $2 * 10^{-7}$, which correspond roughly to the 100 most frequent and the 5 less frequent words, and we shuffled and divided the set into two subsets A and B to get two statistically equivalent sets.

5.2 Corpus and frequency tensor

After having processed the 2.2 M sentences of the 521 books of the training corpus with the Stanford CoreNLP tools (Manning et al., 2014), including lemmatizing, dependency parsing, coreference resolution, we did the same for all sentences of the question sets, completed with our specific treatments to get agent, patient and coordination dependencies, skip-deps and skip-grams. We get a vocabulary of 211 K words, represented as pairs (lemma, word class). We kept as basis words the wn = 6909 lemmas appearing in the questions (MR + A and B new sets of questions), count all the corpus relations containing two basis words, and get 106 M occurrences to build the tensor F_{ijk} of dimension (wn, wn, rn), with rn = 14.

6 Relevance features and question solving

6.1 Relevance features

For each candidate sentence s of each question q we list all oriented relations or triplets *(w1, w2, r_k)* involving the candidate word *cw* (*cw* being either *w1* or *w2*).

i	features f_i	
0	agt	agent
1	pat	patient
2	mod	modifier
3	cor	coordination
4,5	sd1, sd2	1 and 2-skip-deps
6	sdp	set of direct deps
7	srdp	set of reverse deps
8-15	sg0 to sg7	0 to 7-skip-grams

Table 1: Relevance features of model S

In a first approach we designed a large list of 33 relevance features and a rather complex algorithm to tackle the MR challenge, which we will call model H (for heuristic). When switching to our question sets A and B, we tried to simplify it, and we came down to 16 relevance features (listed in table 1) and a straightforward algorithm, which we call S (for simple). We will here describe first the model S, and further the model H as an extension of the former.

As the occurrences of direct dependencies are rather rare, we try to enlarge their scope. In the model S we have kept two extensions. For each dependency triplet *(w1, w2, r_k), $k \in \{0...3\}$*, of the sentence we consider also the set of the other direct dependencies $sdp = \{(w1, w2, r_i), i \in \{0...3\}, i \neq k\}$, and the set of the reverse dependencies $srdp = \{(w2, w1, r_j), j \in \{0...3\}\}$. For each of these two sets we sum the association values of the 3 or 4 relations of the set.

6.2 Relevance score vector (model S)

For each sentence we have a set of triplets $\{(w1, w2, r_k)\}$. The association value of each triplet is added to the corresponding features f_i according to the algorithm 1 described below to compute the relevance vector R of the sentence.

For an efficient use of classifiers, the input values have to be scaled or normalized. For that we have divided the components of the relevance vector R of each sentence to comply with the following principle : if each relation would have an association value of 1, the sum of the vector components would be 1.

Algorithm 1 to compute the relevance vector R

for $(w1, w2, r_k)$in set of triplets$\{(w1, w2, r_k)\}$ **do**
  $R_k \leftarrow R_k + A(w1, w2, r_k)$
  if $k \in \{0, .., 3\}$ **then**
    for $m \in \{0, .., 3\}$ **do**
      if $m \neq k$ **then**
        $R_6 \leftarrow R_6 + A(w1, w2, r_m)$
      $R_7 \leftarrow R_7 + A(w2, w1, r_m)$

6.3 Relevance score vector (model H)

In this more detailed model, we firstly use more relevance features. We split the 5 previous dependency features into 11 features to take into account the influence of the word-class of the two related words. To do that we analyze the frequency of the triplets (word-class 1, word-class 2, relation), and take the most frequent ones. We also add for each skip-gram (w1, w2, sg_i) the corresponding reverse relation (w2,w1,sg_i). This leads us to a list of 33 relevance features.

In a second phase we try to compact these data without impacting the accuracy of the model, driven by the experimental results. So we got down to 20 features and 8 relation types, by forgetting the skip-grams sg 5 to sg7, by simply adding the frequencies of sg3 and sg4 to sg2, and the same for the reverse skip-grams. So we keep only 3 skip-gram relations sg0 to sg2, and the reverse relations rsg0 to rsg2. Finally the best accuracy was obtained by replacing every skip-gram relevance score A(w1,w2,sg_i) in algorithm 1 by the sum: $\sum_{i \in \{0,1,2\}} A(w1, w2, sg_i)$.

6.4 Question solving

Having computed a relevance vector for each sentence of a question, a **direct solution** consists in simply adding the components for each sentence and compare the results of the five sentences.

We can also apply supervised **machine learning classifiers** to these vectors. To do so, for each question, from the five sentence vectors of fn components (16 for model S and 20 for model H), we first sort the 5 sentences by decreasing order of direct score (sum of the components of the vector). We select the n best sentences (n between 2 and 5), concatenate their n vectors to get a unique "flat" vector of n x fn components to represent the question. We have to choose the value of the "flattening" parameter n to obtain the best tradeoff between underfitting, if the number of features is too small compared to the size of the development set, and overfitting if we introduce too many features. In our case, both for the 1040 MR questions and 3000 A or B questions, our best value was n = 3.

During the training or development phase of the learning process, we know the rank of the answer score (right sentence). The question can be used for learning only if this rank is inferior or equal to n. During the testing phase, from the *flat* or concatenated vector of the question the classifier

predicts the rank of the answer.

We proceed to a 5-fold cross-validation which consists in dividing the question set into 5 equal subsets, and to run the classifiers 5 times with 4/5 of the questions for the development and the last 1/5 for testing. So that any question is used four times for development and one for testing.

We have tested 10 different classifiers provided by scikit-learn.org [1]. The best one for this application in most cases is the multilayer perceptron classifier (MLPC), using backpropagation to minimize a cross-entropy function, with an alpha regularization parameter which can be adjusted. The support vector machine classifier using a radial basis function (SVM RBF) gives also good results, generally about 0.5% lower that the former. Both are able to process multiclass data as in our application which has n classes (our *flattening* parameter).

7 Experimental results

7.1 Success ratio per relevance feature

With the relevance vector of each sentence of a question, we can already compute the success ratio of any relevance feature f_i, i.e. the number of questions for which this feature attributes the best score to the good answer. We do it in two steps. First we compute the *contribution ratio* of the feature, i.e. the percentage of sentences where the corresponding relation exists for the candidate word with an association value different from zero. Then for these sentences where the feature is contributing we compute its success ratio which we call its *intrinsic efficiency*. And we define its *global efficiency* as the product of the contribution ratio by the intrinsic efficiency. This allows us to compare the efficiency of every feature for every association model.

Figure 2 shows the results for 3 association models on A+B or MR questions. The intrinsic efficiency of the four basic dependency relations stays between 70 and 80%, but as their contribution ratio is much lower than for the skip-grams, their global efficiency is lower.

7.2 Direct summation

By just adding the components of the feature scores of the sentences, we already get with model S on A+B questions scores between 76.9% for PPMIA and 44.2% for MLE as detailed in table 2.

	model H		model S	
	direct	clf	direct	clf
PPMIA	75.4	**77.0**	76.9	**78.6**
PPMID	70.5	71.7	66.7	70.0
LLR	67.5	69.6	68.7	69.5
PPMIS	60.8	63.0	63.2	66.5
MLE	47.0	53.9	44.2	54.5

Table 2: % accuracy on A+B questions

	model H		model S	
	direct	clf	direct	clf
PPMID	58.1	**60.0**	53.9	55.4
PPMIA	54.2	54.6	52.3	54.1
PPMIS	56.6	58.4	52.3	53.6
LLR	36.4	37.7	39.3	40.2
MLE	37.5	37.6	35.5	36.1

Table 3: % accuracy on MR questions

As the MR questions have been intentionnally generated to be difficult to solve, it is not surprising to observe lower scores from 53.9% for PPMID to 35.5% for MLE (table 3).

With model H we get comparable results on A+B sets, and better results on MR set, up to 58.1% for PPMID.

7.3 Machine learning classification

With the multilayer perceptron classifier MLPC we gain between 1 and 10 % accuracy over the direct summation. Figure 4 shows the detailed results of PPMIA, LLR and PPMID association models in the 5-fold cross-validation, with the dispersion over the 5 folds, and the resulting means.

Figure 5 and table 3 show that the model H improves substantially the efficiency of the PPMID association model, approaching the state-of-the-art accuracy on MR questions at 60% despite the constraint of the k-fold cross validation averaging the test results on the whole set, but with an important dispersion : the maximum value on the highest k-fold reaches 64.9% illustrating the risk not to use k-fold validation.

Levy and Goldberg (2014) highlighted the correspondance between hyperparameters used by neural network models and association count-based models. The PPMIA model with the hyperparameter alpha = 0.55 gives the best result instead of 0.75 recommended by several authors. The PPMIS

[1] From : `http://scikit-learn.org/`
The other classifiers tested are: AdaBoost, K Nearest Neighbors, Linear SVC, Gaussian Process, Decision Tree, Random Forest, Gaussian Naive Bayes and Quadratic Discriminant Analysis.

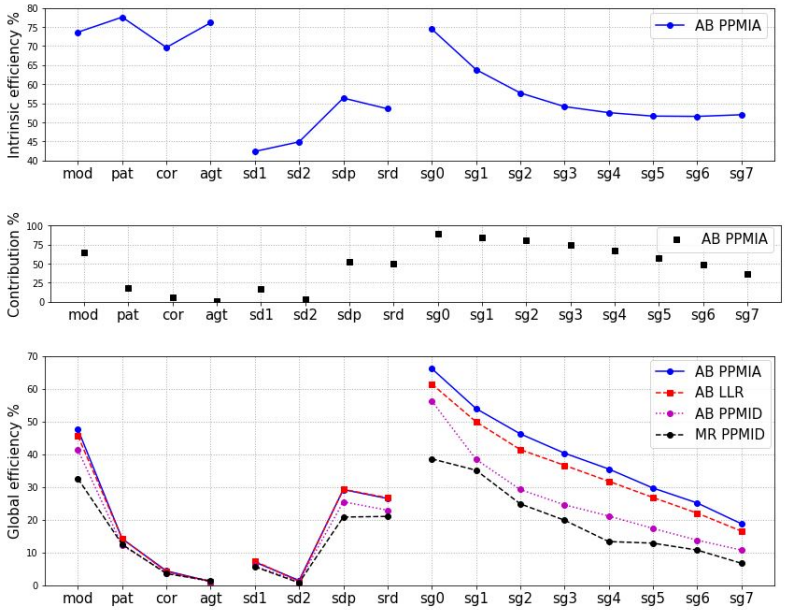

Figure 2: Feature success and contribution ratios

Top figure shows the intrinsic efficiency of every feature (i.e when contributing) for PPMIA (alpha = 0.55)
Middle : contribution ratio of every feature on AB sentences and PPMIA (alpha = 0.55)
Bottom : overall efficiency of every feature in four cases (product of intrinsic efficiency by contribution ratio). Lower curve shows an abnormally low sg0 value for MR sentences

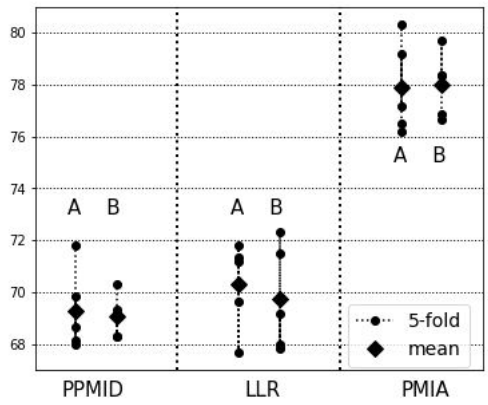

Figure 4: k-fold results for model S on A & B sets

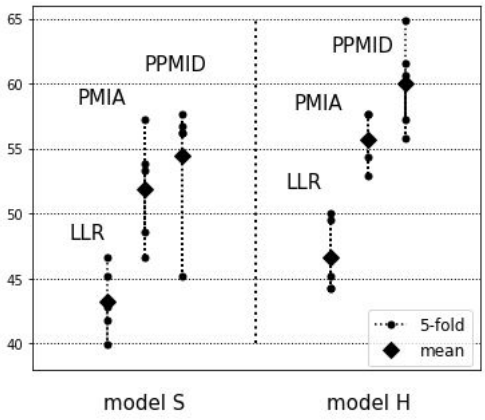

Figure 5: k-fold results on MR set

model in our case has its optimum at k = 1.5, but with a much lower performance than the former.

8 Discussion

8.1 Question sets

Several previous works on sentence completion have tackled the MR challenge. The first result approaching 60% was by Mikolov et al, 2013 with a combination of three neural network models at 58.4%, without syntactic context. Zhang et al. (2016) applied neural network LSTM to model the dependency tree of the sentence and reached an accuracy of 60.7%. Their model includes implicitly right and left linear word proximity so it is actually a combination of dependency and linear contexts. They use a set of 4000 sentences of the training corpus as a validation set to train the model before applying it to the 1040 sentences of the test. Woods, 2016 combined dependency context and n-grams and claims a performance of 61.4%. Our approach is comparable to her's, and we agree on the optimal efficiency of PPMID on the MR set.

If we rely on the results on the MR question set we should conclude that the best association model is the PPMID as it overpasses others both with models H and S, and that model H is more efficient than model S. But the bias introduced in it gives

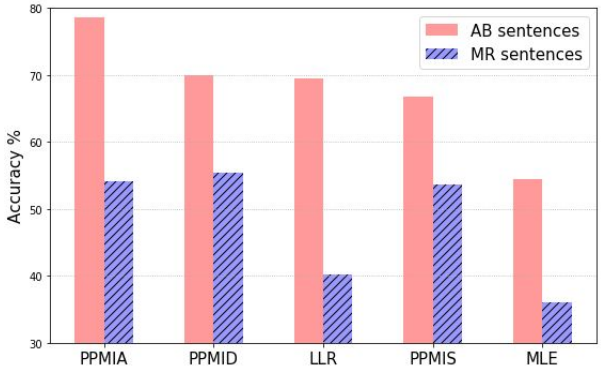

Figure 6: Efficiency of association models on MR and AB sentences (Model S)

an effect which is visible on figure 2, on which we see that the performance of skip-gram sg0, which is the relation used to increase the difficulty of the questions, is abnormally low compared to its values on questions sets A+B. As the questions sets A and B are generated without any influence of the relevance features, and as they give results clearly different from the MR question set, we must follow the former to evaluate our semantic models. For that purpose we provide open access to these question sets A and B for the use of the research community [2], with a detailed description of the generating process.

8.2 Relevance features and association models

On figure 6 we compare the efficiency of the association models on MR and AB sentences. It is not surprising to observe a higher efficiency on AB than on MR sentences as the latter have been generated with constraints on bigram frequencies (corresponding to our sg0) to make the challenge harder. But the relative efficiency of the models being very different, we must rely on AB sentence to evaluate them.

The PPMIA model gives the best result, with the parameter $\alpha = 0.55$, rather different from the value 0.75 generally used. The PPMIS model gives its best result with the parameter $a = 1.5$ on sentence completion, where Levy et al. (2015) found values from 1 to 5 depending on the type of task, word similarity or analogy.

The model H was developed and finetuned for the MR sentences and it shows a better efficiency on the MR set. Its efficiency on A+B questions is not far from model S. It has the advantage of

[2]http://www.unicog.org/biblio/Category/misc.html

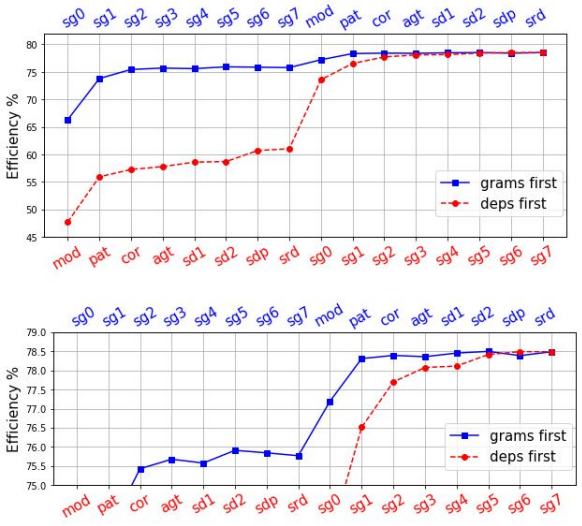

Figure 7: Efficiency with increasing number of features (bottom figure = zoomed view of top figure). Bottom curves : beginning by dependency context. Top curves : beginning by skip-grams.

needing much smaller frequency and association tensors (8 relations instead of 14). But its relevance computing is the rather opaque result of a long heuristic path. Taking into account the reverse skip-grams for instance seems to increase the efficiency, but it is difficult to understand why. On the contrary the model S is very straigthforward. And by using sparse matrix tools, the memory size of the association tensors which store all the information necessary for solving all the questions (MR+A+B) remains reasonable (about 50 Mo in our case for the association tensors).

8.3 Dependency vs linear context

To study the contribution of every feature to the global result, we have computed the efficiency of the model when cumulating the features one after the other, first by starting with the skip-grams then by the dependency relations, giving the results of figure 7.

We have applied our model S with PPMIA ($\alpha = 0.55$) to the question set AB (6000 questions) by adding successively the features and measuring the evolution of the success ratio.The 4 dependency relations together reach a level of 58%, and 61% with sd1,sd2, sdp and srdp. The efficiency jumps up to 73% with sg0 and then reaches almost its asymptote at sg5 at 78.5%, close to the best result 78.6 % with all the features . Skip-gram context alone reaches a maximum of 76.5% at sg5.

Finally the dependency context brings an im-

provement of about 2% over the linear context, which is not negligible, as it is higher than the improvement of the classifier over direct computing (about 1.5%).

The best compromise would be to keep the 6 first skip-grams and the 4 first dependency features giving a result of 78.5%. The features sdp and srd (sum of dependencies and sum of reverse dependencies) seem to be redundant with skip-grams. Moreover, as sd1 and specially sd2 (skip-deps 1 and 2) contribute very little to the final result, they can be given up to reduce the volume of computing and of memory without significant decrease of accuracy.

8.4 Content words vs function words

We explored also the contribution of function words in addition to content words. When taking into account all the function words we get about the same performance. As it multiplies by about 2.4 the number of relations in the corpus there is no real interest in doing so.

9 Conclusion

We have confirmed that sentence completion is a good benchmark for testing and comparing semantic features and models.

A straightforward algorithm with explicit computings, consisting in just adding association values of the relations in the sentences, already provides interesting performances of 77%, compared with chance level of 20%. With the contribution of a machine learning classifier the performance reaches 78.6% in this particular multiclass situation. We have been able to rank the association models of the litterature. The PPMIA model reusing the $alpha$ parameter of most neural network models, re-adjusted to a value of 0.55 instead of 0.75, proved to be the best. In this application with a small corpus of 20 M words, the optimal context to take into account to gather statistics about words includes skip-grams in a window of 6 content words before and after the target word and the 4 syntactic relations: modifier, patient, coordination, agent.

This new model combining multi-relation semantic tensor representation and relevance computing could be used for several other semantic tasks such as word sense desambiguation, translation, dependency parsing, question answering, or information retrieval.

Acknowledgments

This work is part of the NCM-NL (Neurocomputational models of Natural Language) French-US project and was done at Neurospin (INSERM/CEA) at Saclay. I thank Christophe Pallier for his support and valuable comments, and anonymous reviewers for their positive suggestions.

References

[Church and Hanks1990] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

[Dunning1993] Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

[Fyshe et al.2015] Alona Fyshe, Leila Wehbe, Partha Pratim Talukdar, Brian Murphy, and Tom M. Mitchell. 2015. A Compositional and Interpretable Semantic Space. In *HLT-NAACL*, pages 32–41.

[Gubbins and Vlachos2013] Joseph Gubbins and Andreas Vlachos. 2013. Dependency Language Models for Sentence Completion. In *EMNLP*, pages 1405–1410. Citeseer.

[Guthrie et al.2006] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4. sn.

[Hastie et al.2008] Trevor Hastie, Robert Tibshirani, and Friedman. 2008. *The elements of statistical learning*, volume 1. Springer series in statistics New York.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

[Levy et al.2015] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

[Lowe2001] Will Lowe. 2001. Towards a theory of semantic space. In *Proceedings of the Cognitive Science Society*, volume 23.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pages 55–60.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mirowski and Vlachos2015] Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. *arXiv preprint arXiv:1507.01193*.

[Pado and Lapata2007] Sebastian Pado and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

[Pantel and Lin2002] Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

[Turney and Pantel2010] Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *arXiv:1003.1141 [cs]*, March. arXiv: 1003.1141.

[Woods2016] Aubrie M. Woods. 2016. Exploiting Linguistic Features for Sentence Completion. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 438.

[Zhang et al.2016] Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down Tree Long Short-Term Memory Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320, San Diego, California, June. Association for Computational Linguistics.

[Zweig and Burges2012] Geoffrey Zweig and Chris JC Burges. 2012. The Microsoft Research Sentence Completion Challenge. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 29–36. Association for Computational Linguistics.

[Zweig et al.2012] Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher JC Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 601–610. Association for Computational Linguistics.