December 2022

Named Entities In Conversation Fingerprinting For Knowledge Dissemination Detection

Sébastien WARICHET
ALE International

Emmanuel HELBERT
ALE International

Nadir DEBBAGH
ALE International

Docket Number	**FR82022001**
Title	**Named Entities In Conversation Fingerprinting For Knowledge Dissemination Detection**
Contributors	**Sébastien WARICHET, Emmanuel HELBERT, Nadir DEBBAGH**
Company	**ALE International**

Description of the technical solution:

1) Context:

The overall context is about how to evaluate the interactions between employees in an organization and their ability to disseminate the knowledge throughout the different departments and teams. This is highly relevant for Human Resources Department in order to identify the skills in the organization, the bottleneck in communication between teams and the optimization of resources in the case of the same skill or knowledge appear in different places in the organization.

This can be done manually through individual interviews or team reports consolidation. Our aim is to make it automatic. And a way to do that is to rely on the communication graphs. A communication graph represents basically the communication links between individuals in a community. Let consider a company and its subcontractors and clients as the community perimeter. Each time someone has an interaction with someone else within this perimeter, this creates a link between these two entities with additional data characterizing the link (type of media, date, duration, content of the communication if possible, persons involved, etc.). The technical problem we face is then twofold:

- How can we leverage the data collected in the graph to evaluate the dissemination of information throughout the community?
- How can we leverage the data collected in the graph to identify if similar knowledge is discussed and disseminate in different place in the community without communication links between them?

2) Current solutions and limitations:

An internal process ensuring knowledge management can do the trick. But it requires resources and is not real time.

Regular individual and team interviews but this requires even more resources, time and is definitely not real time. It is only a snapshot at one moment in the community life.

Artificial Intelligence (AI) can be a good way to analyze the conversations between employees and detect similarities. This is already extensively done in recommendation algorithm on webstores.

One can mention for instance sentence embedding with libraries such as SBERT already used to obtain fingerprint from text and conversation. Our belief is that these existing libraries have strong limitations because they only consider words in the lexicon and not named entities (such as company names or products) and have poor performance in case of long conversations. The core of our proposal is then

to imagine and design a system which overcome these limitations thanks to the use of Named Entity processing.

The experiment below explains why a pure sentence embedding has limitation for our case. It also highlights the pure Named Entity recognition limitations and validate our hypothesis above:

Problems with a pure Sentence Embedding approach:

By considering only an approach based on sentence embeddings, two messages tackling the same problem and having the same subject for two different clients will have a strong similarity. Indeed, we give an example of a list of e-mails where two of them are almost the same and differ only by the named entities that they contain:

Original e-mail :

Dear Jean-Baptiste,
This is Richard DUBOIS, from Transportly. I have met your colleague Sarah in Paris yesterday during the Go For It 2022 event . She gave me your contact.
Sarah mentioned your cloud communication platform "Microsoft Teams" and told me about all the transportation and geo-tracking solutions you provide.
Your services seem to be appropriate for our needs as a growing business in the transportation market. I believe my colleag ue Evan has already sent our plans as well as the resources in a separate e-
mail to the Sales Services as suggested by Sarah. They contain all the details concerning our truck fleet and our needs in ter ms of functionalities for this project.
Is it possible to have the pricing for the deployment and maintenance of this solution by next week? If you have the estimat es worked up for similar projects, could you please email them to me ASAP as well?
Thank you for your time. We're excited about potentially using your product and services.
Best regards,
Richard DUBOIS

Modified e-mail :

Dear Jean-Louis,
This is Nadir DEBBAGH, from You Move. I have met your colleague Solène in Monaco yesterday during the Ready For It 202 2 event. She gave me your contact.
Solène mentioned your cloud communication platform "Rainbow" and told me about all the transportation and geo-tracking solutions you provide.
Your services seem to be appropriate for our needs as a growing business in the transportation market. I believe my colleag ue Evan has already sent our plans as well as the resources in a separate e-
mail to the Sales Services as suggested by Solène. They contain all the details concerning our truck fleet and our needs in ter ms of functionalities for this project.
Is it possible to have the pricing for the deployment and maintenance of this solution by next week? If you have the estimat es worked up for similar projects, could you please email them to me ASAP as well?
Thank you for your time. We're excited about potentially using your product and services.
Best regards,
Nadir DEBBAGH

Reference e-mail (completely different than the first two but in the same subject):

Hey Sarah,
Thanks to take this event opportunity to promote Rainbow !
I didn't receive the plans, Richard is talking about, perhaps Evan sent them directly to U ?
I propose you contact Fred for having an R&D view of the feasibility of this project, and also of the potential delay.
Possibly the Verticals have already sell some similar features, Fred should know that.
Please keep me inform to know whether we have to build a project team, and when.
Rgds,

JBaptiste

Reference e-mail #2 (completely different than the others and in a different subject):

Hey guys,
The finalization and the publication of the newsletter is not a matter of one click, it takes time !!
Most probably we will only be able to publish it during next week, so I propose to keep the initial date (June, 10th), that was agreed by everyone... For that target date, I see, today, no problems to do the job.
You both should schedule a organizational meeting: everybody can share their constraints and we would be able to build a better planning template for the next editions of the letter.
Thx to take that point.
Rgds,
Louis

We calculate the cosine similarities between the SBERT embeddings of the pairs of e-mails and obtain the results summarized in the following table:

Mail 1	Mail 2	Cosine Similarity SBERT
m_original	m_modified	0.8701
m_ref	m_original	0.5184
m_ref	m_modified	0.4685
m_ref2	m_original	0.3412
m_ref2	m_modified	0.2980
m_ref2	m_ref	0.2981

We notice that the similarity between “m_original” and “m_modified” is very high despite the fact that they concern two completely different clients “You Move” and “Transportly” and are thus very dissimilar in an enterprise standpoint.

In addition, if we consider the similarities between “m_original” and “m_ref”, as well as between “m_modified” and “m_ref”, we would need a threshold of at least 0.5184 to consider that “m_ref” is linked to “m_original” (which is the case); and thus, this makes “m_original” linked to “m_modified” (because $0.8701 \geq 0.5184$) even though they concern completely different entities.

We also considered similarities between “m_ref2” and the other e-mails to have a reference point for the similarity between e-mails from different subjects.

This shows that sentence embeddings cannot differentiate the information in the enterprise, indeed, simply changing the named entities makes the system wrong.

Another drawback of using Sentence Embedding based techniques is the fact that; if a word is not in the training dictionary, then it will be not considered in the calculation. For example, if we mention the word “Rainbow”, which is a cloud communication platform provided by “ALE International” company,

an embedding method will most probably consider the word as being the natural phenomenon instead of the product.

Problems with a pure Named Entity Recognition approach:

Named Entity Recognition (NER) allows to extract the “objects” or “entities” from a given text such as person names, enterprises and organizations, dates, product names, numbers, locations (e.g. countries, cities, addresses). NER modules analyze the text structure and tag the different entities with their corresponding type.

NER on its own cannot determine if two conversations contain the same information. If we quantify conversation similarity using the number of entities in common between two communications; then mentioning the same entities in both communications would make the system predict a strong similarity even if they treat two different problems (this could be the case when the two communications are between the same persons and for the same client but for two different subjects).

Finally, NER could be used to detect words similarities in two conversations but in that case, the NER algorithms would completely miss correlations between conversations using different words but with the same semantics.

Conclusion:

Thus, it is interesting to consider sentence embeddings as well as named entity recognition jointly in order to quantify conversation similarity because as we could see in the two previous sections, the information brought by the two techniques are complementary and cover different aspects of conversation relatedness.

3) New solution:

The objective of this proposal is to detect similarities between conversations enabling to visualize the propagation of conversation topics as well as the appearance of a similar topic in different places of the organization. Such a feature can then be used to recommend connection between people having similar conversation. Within an enterprise, this can speed up project development, fixing issues, secure business and improve skills.

The new solution provides a feature which is:

- Automatic
- In real time
- Ensuring personal and professional data privacy
- Comprehensive (the whole community is considered)
- With a higher performance

It strongly improves the ability to discriminate conversation based on named entity consideration. The solution does not require training either which ensures data privacy and quick deployment on any site.

The principle of the solution is to characterize the content of a conversation with a fingerprint and use this fingerprint to compare different conversations without knowing their content. Thus, the solution

can guarantee the confidentiality of the data. The solution could also be used to visualize the conversation evolution hop by hop throughout the conversation graph.

The main technology used in this solution is the **conversation embedding** or the ability to transform into vectors words in text format. The core of the solution lies in the way we do this transformation. Once conversations are modelized in vectors, we just need to compare all vectors one to one with distance processing to detect similarities in some conversations or not.

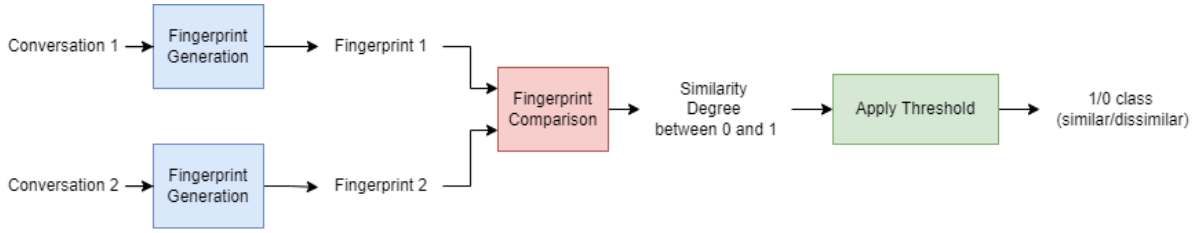
The state of the art already provides some tools to perform embedding on words. Two models are massively used: word2vec and SBERT. SBERT is the best solution as it encodes each word with its context; i.e. the other words close to it. But this has limitation as what we intent to do is to encode conversations, not only words. And SBERT requires more processing. The current usage for conversation embedding with word2vec is to calculate a mean vector based on the vectors of all words used in the conversation. This has strong limitation in the case of long conversations as there is a risk that relevant information in one part of the conversation is diluted by other meaningless parts. In addition, this method does not take into account the place of the words in the conversation.

Hence, the solution relies in a conversation processing method with three main steps:

- fingerprint generation with two parallel fingerprints generation methods
- fingerprint comparison
- conversation management

Processing Method for step 1 and 2:

The first two steps aim at identifying similar conversations within the graph. They allow to first generate fingerprints for conversations, then compare them using a mixed similarity function we proposed. The pipeline is described in the following diagram:



When given two conversations A and B as input, our architecture generates first what we call a “fingerprint” for each of the two messages. Then, both fingerprints are compared, and a threshold is applied to detect if both conversations are similar or dissimilar. The choice of the two conversation is described in step 3.

STEP 1: Fingerprint Generation:

A fingerprint is a numerical representation of the message, and the calculation of the fingerprint is based on the two approaches (sentence embedding and NER); it is composed for a given message of:

- A sentence embedding vector (SENT_EMB)
- A bag of named entities (NER)

To generate a fingerprint, our architecture joins two complementary fingerprints obtained from two different methods:

- **Sentence Embeddings Generation**: We generate a vector numerical representation of the text based on its context and meaning.
- **Named Entities Detection**: We generate the bag of entities which will contain the list of entities mentioned in the conversation given as input.

The sentence embeddings generation is based on state-of-the-art pretrained models and we tested two of them, word2Vec and SBERT. The Named Entities Detection relies on a Named Entity Detection module which represents the core of the solution in addition to the original architecture mixing both sentence embeddings and named entities approaches.

NAMED ENTITY DETECTION MODULE:

In this solution, to perform named entity recognition (NER), we use a pretrained named entity detector provided by the SpaCy library, we rely specifically on the `en_core_web_lg` model for English material, but the solution is adaptable according to the memory and processing time requirements by using a smaller model like `en_core_web_md` or `en_core_web_sm`. It is also possible to exploit this solution on different target languages by using a corresponding NER model (for example, the `fr_core_news_md` model for French)

The Named Entity Detector outputs for a given text, the list of entities it contains in addition to their type. The type could be a person, an organization, a product, a location, a date, a cardinal (number)… SpaCy is able to attach the right type to the named entity detected.

Despite the strength of NER, it still has some limitations like processing time. Indeed, in our solution, NER takes from 65% to 98.84% of the processing time, which is a very considerable fraction of the work. To optimize named entities retrieval, we adopted an approach based on a dictionary of named entities.

In addition, there is a case where named entities can be common words (e.g. Rainbow, Cirrus, Panda). For this case, we will detect the use of these common words as named entities by comparing the frequency of use of these words with the standard frequencies. These standard frequencies are associated to a specific language and quite easily available. We can assume that the solution uses existing technology to detect the language of the conversation and thus select the right statistics. Regarding the implementation, the solution will manage a dictionary per language with frequency statistics for each language. The detection will simply be done by comparing the ratio of the word frequency calculated in a corpus of internal conversation divided by the general word frequency for the detected language to a threshold.

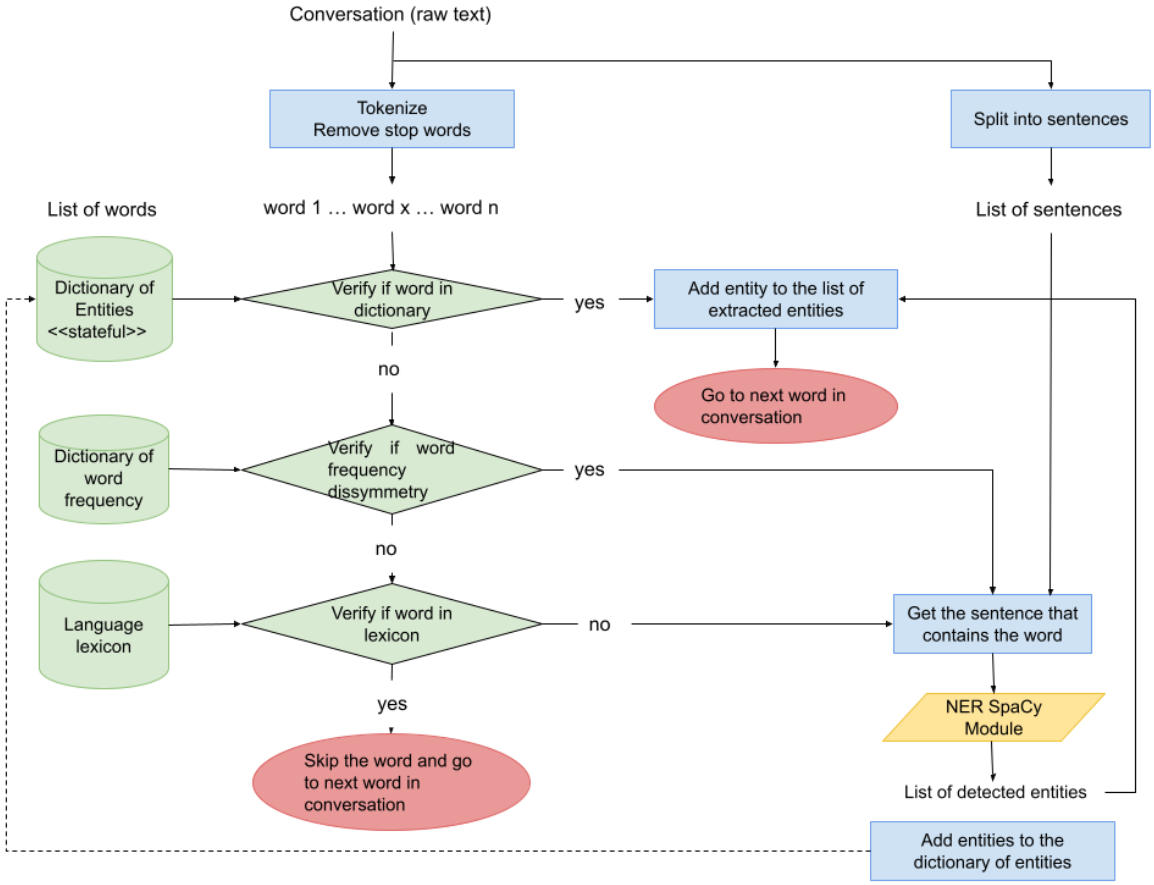
The whole algorithm is described below.

When given a text, named entity extraction will be performed by first tokenizing the text. We then check a dictionary of named entities to verify if the entity has already been identified and as so, recorded in the dictionary. If it is the case, we add it in the list of extracted entities for this conversation. If it is not already in the dictionary, we then check if there is a dissymmetry in the word frequency by comparing the ratio of the word frequency in the conversation corpus divided by the usual word frequency in the detected language to a threshold. If the ratio is higher than this threshold, the word is considered as a named entity and a NER is performed on the phrase that contains the unknown word to extract the named entities it contains, then add them to the list of extracted entities as well as the dictionary. If not, we check that the word is in the lexicon and if not, we perform NER again the same

way as above. Note that we blacklist (ignore) all the words that are not in the target language lexicon and are not detected by NER to avoid unnecessary calls to the SpaCy NER module.

It may happen that the NER module classifies an entity with different type. The module records all classification with the occurrence number of each type. It is then possible to automatically choose the type with the highest number of occurrences. The module also gives the possibility to the conversation initiator, through a simple GUI (Graphical User Interface), to manually confirm the right type to consider for this entity.

This process is described in the following diagram:



By using this approach, we obtain a significant speedup on NER computation despite a small error. The following table shows the gain obtained from using NER conditioned on a dictionary (our approach), instead of NER computation. The execution times were calculated on a list of distinct e-mails sent within the same enterprise "Enron"; the dataset is publicly available on Kaggle `wcukierski/enron-email-dataset`.

Number of E-mails	Using NER Only	NER with a dictionary	Speedup
20000	22 ***min*** 51 ***sec***	6 ***min*** 58 ***sec***	**3.27**
40000	1 ***h*** 18 ***min*** 52 ***sec***	16 ***min*** 35 ***sec***	**4.75**

SENTENCE EMBEDDING GENERATION MODULE :

In our experiments we used two approaches to generate sentence embeddings, but the module can use any sentence embedding library:

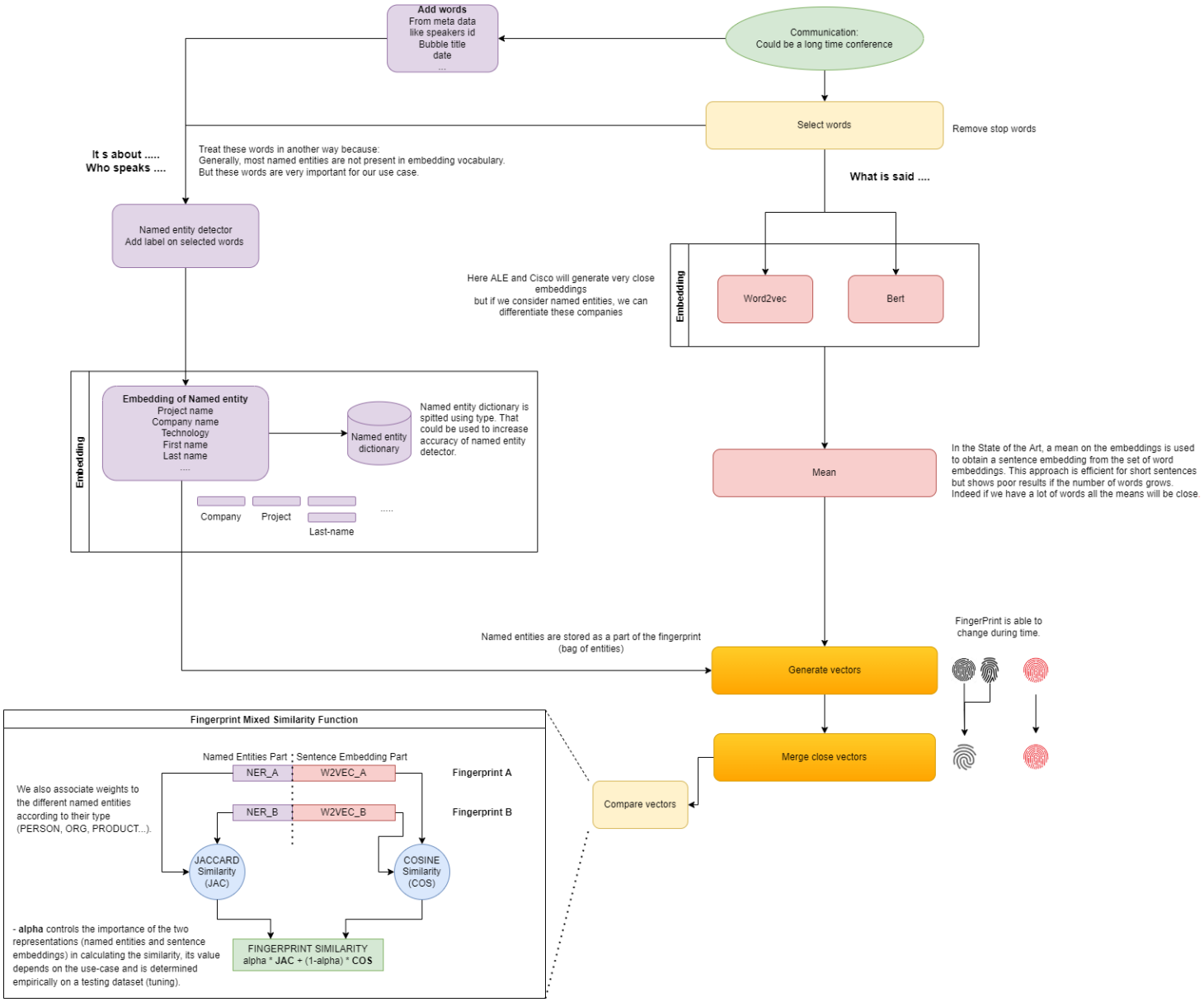
- Word2Vec: We generate embeddings for each word of the conversation (if they exist in the word2vec dictionary), then we calculate the mean of the word vectors to obtain a sentence embedding in 300 dimensions.
- SBERT: We use a pretrained Sentence Transformers (Siamese SBERT) to generate a sentence embedding in 384 dimensions (for the small SBERT model) or 768 dimensions (for the large one).

The output of this module for a given conversation is its vector representation (sentence embedding).

The drawing below depicts how these two parallel processes are organized within the solution.

The right leg is dedicated to the analysis of general conversation through sentence embedding. The left leg is dedicated to the analysis of metadata associated to the communication graph (date, title of a meeting, words in the invitation to the meeting) and named entities detected by the dedicated block described above. These metadata and names entities are collected and embedded in the bag of entities mentioned above. Thus, a conversation fingerprint is the association of these two partial fingerprints: Sentence Embedding (Vector) and Named Entities Embedding (bag of entities).

Both paths process words extracted from the real-time conversation. If it is a text conversation (chat), that is simple. If this is a voice or video conversation, an ASR (Automatic Speech Recognition) engine is first applied in order to get textual words.



The second part of the solution consists in an estimation block that will compare the fingerprints of different conversation to detect if these conversations deal with the same subject or different subjects. This is done in step 2 thanks to a fingerprint comparison module.

STEP 2: Fingerprint Comparison:

Once the fingerprints generated for the two conversations A and B, we obtain two representations composed each of an NER part and a SENT_EMB part:

$$Fingerprint\ A\ = (NER_A, SENT\_EMB_A)$$

$$Fingerprint\ B\ = (NER_B, SENT\_EMB_B)$$

In this step, we associate a degree of similarity between the two fingerprints. For this task, many techniques and different similarity metrics can be used, we propose a mixed similarity which combines:

- A **Weighted Jaccard** similarity calculated between the two bags of entities NER_A and NER_B.

- A **Cosine** similarity calculated between the two sentence embedding vectors (SENT_EMB_A and SENT_EMB_B).

Note that both similarities can be replaced by others that are more suited depending on the way named entities have to be considered (for the first similarity metric), but also the embedding generation process (for the second one).

Weighted Jaccard Similarity:

When given two bags of entities NER_A and NER_B, we first associate weights to the named entities according to their type. (Recall that NER_A contains the list of named entities that appear in conversation A, and the same is for NER_B).

For example, if we consider, for a given use-case, that the entity types "ORG" and "PERSON" are more important than all the others (entities of interest), we associate a weight β to these, and a weight $1 - \beta$ to all the entities of other types (with $0.5 \leq \beta \leq 1$ to ensure that the entities ORG and PERSON have bigger weight than the others).

This β parameter is chosen empirically on a testing dataset, and the "entities of interest" are determined according to the use-case.

We chose to use a single parameter β with two groups of entities (entities of interest/less important entities) in order to minimize the number of hyperparameters to tune. Indeed, if we associate a different weight to each entity type, there is a combinatorial explosion of the number of possibilities, making thus a grid search over the parameters space nearly impossible in a reasonable amount of time.

Once the β value fixed, we calculate the weighted Jaccard similarity by following these steps:

1. Calculate the sum of weights of the intersection of the two bags of entities NER_A and NER_B ($INTER$).
2. Calculate the sum of weights of the union of the two bags of entities NER_A and NER_B ($UNION$).
3. Calculate the weighted Jaccard similarity: $JACCARD = \frac{INTER}{UNION}$

Example:

Let suppose that:

$$NER\_A = \{David, Alcatel, Sophie, 13, 13/01/2020, Strasbourg\}$$

$$NER\_B = \{Arnaud, Alcatel, Sophie\}$$

We suppose in this example that $\beta = 0.75 \Rightarrow$ Thus $1 - \beta = 0.25$. We also suppose that the entities of interest are PERSON and ORG.

For our two bags of entities, we have the following types:

$$TYPE\_A = \{PERSON, ORG, PERSON, CARDINAL, DATE, LOCATION\}$$

$$TYPE\_B = \{PERSON, ORG, PERSON\}$$

And thus, the associated weights are:

$$WEIGHTS\_A = \{0.75, 0.75, 0.75, 0.25, 0.25, 0.25\}$$

$$WEIGHTS\_B = \{0.75, 0.75, 0.75\}$$

We can then calculate the intersection and the union as well as their associated weights:

$$NER\_INTER = \{Alcatel, Sophie\}$$

$$WEIGHTS\_INTER = \{0.75, 0.75\}$$

$$NER\_UNION = \{David, Alcatel, Sophie, 13, 13/01/2020, Strasbourg, Arnaud\}$$

$$WEIGHTS\_UNION = \{0.75, 0.75, 0.75, 0.25, 0.25, 0.25, 0.75\}$$

Now, by following the previous description of the 3 Jaccard Similarity calculation steps, we calculate the sums of weights of each set of entities ($INTER$ and $UNION$), then compute the Jaccard similarity ($JACCARD$).

$$INTER = 1.5$$

$$UNION = 3.75$$

$$\boxed{JACCARD = \frac{INTER}{UNION} = 0.4}$$

Cosine Similarity:

We use the cosine similarity which takes as input two numerical vectors and calculates their proximity in terms of the angle they form in the representation space. This measure allows to quantify the context similarity between the sentence embeddings of two conversations A and B. It is calculated following this formula:

$$COSINE(SENT\_EMB_A, SENT\_EMB_B) = \frac{SENT\_EMB_A . SENT\_EMB_B}{\left|\left|SENT\_EMB_A\right|\right|_2 . \left|\left|SENT\_EMB_B\right|\right|_2 + \epsilon}$$

Where:

- $||\,.\,||_2$ is the L_2 norm.
- ϵ is a small constant that ensures numerical stability (set to 10^{-5} in our case).
- The . operation denotes the dot product in the numerator and the product between two scalars in the denominator.

Total Similarity and Threshold Class:

The total similarity will then be calculated by weighting the two similarities with an α value which lays between 0 and 1.

$$Similarity(A,B) = \alpha.\boldsymbol{JACCARD} + (1-\alpha).\boldsymbol{COSINE}$$

Notice that an α value of 1 corresponds to exclusively using NER, and $\alpha = 0$ corresponds to a pure sentence embedding based solution.

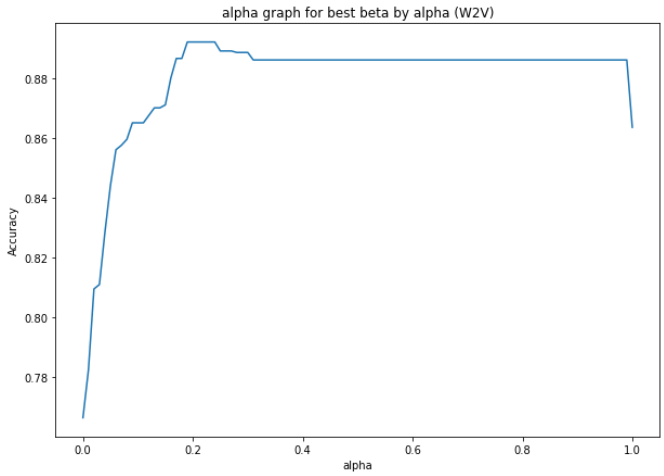
To decide whether the two conversations A and B are similar or not, we apply a threshold on the calculated similarity.

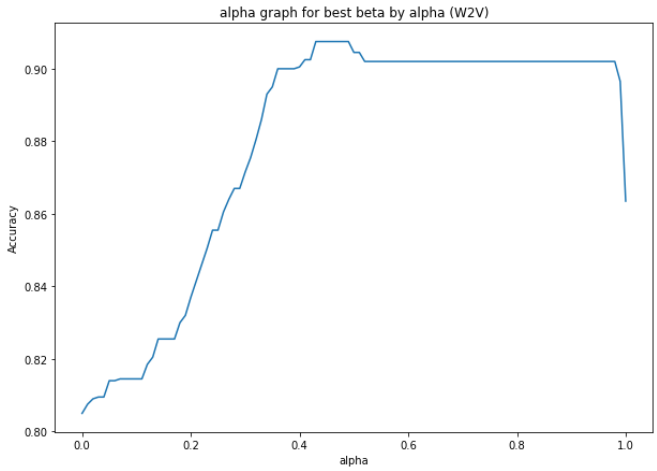
$$Threshold\_Class(A,B) = \begin{cases} 1 & if\ Similarity(A,B) \geq THRESHOLD \\ 0 & if\ Similarity(A,B) < THRESHOLD \end{cases}$$

α, β and THRESHOLD values are determined by a grid search on their values (Threshold between 0 and 1, α between 0 and 1, β between 0.5 and 1). We calculate for each combination the accuracy on a testing dataset, then take the values that maximizes the accuracy.

We obtain the following graphs for two different sentence embedding modules, these show that:

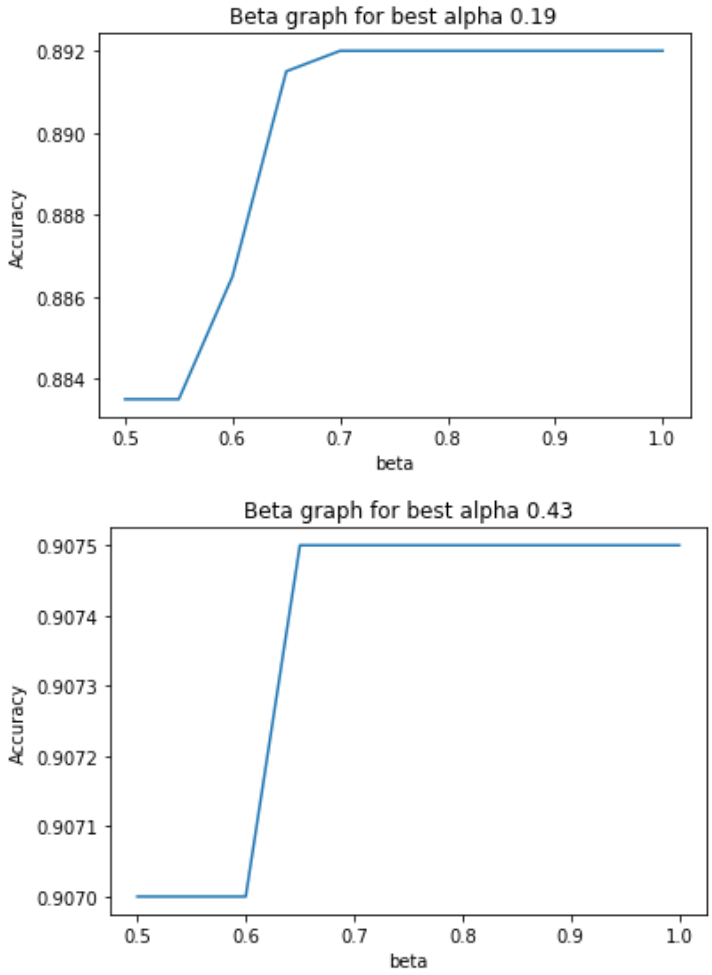
- When using a Mean W2V Sentence Embedding module, an α value equal to 0.19 yields the best accuracy at 89.20%.
- While using an SBERT Sentence Embedding module yields the best result for an α value of 0.43 with an accuracy of 90.75%.





Note that these results show that both NER and Sentence Embeddings are useful for the task, as we can see that both the curves have their maximum for an α that combines both representations with a considerable degree (0.19 for W2V + NER, and 0.43 for SBERT + NER).

Concerning β, the following graphs show the evolution of the accuracy by varying β from 0.5 to 1 with α fixed to the best performing value (0.19 for W2V on the left and 0.43 for SBERT on the right):



By analyzing the graphs above, we notice that β values close to 1 yield better accuracies in this specific case as the three entity types “PERSON”, “ORG” and “PRODUCT” are very informative in our company conversations. Note that this can be different in other use cases or for other companies, hence the necessity to use this β parameter and to tune it properly.

It is also interesting to mention that the tuning of the different parameters (α, THRESHOLD, β) have to be performed once for a given use-case/enterprise to calibrate the architecture. For this task, a calibration/tuning dataset is necessary, it is composed of pairs of e-mails/conversations tagged as either similar (containing the same information, label = 1), or dissimilar (label = 0).

In our case, we compared our approach to the baseline on the forged dataset, we obtained the following results:

Approach	Accuracy	Processing Time
Pure NER	86.40%	0.585 *ms/message*
Pure Sentence Embeddings (Mean of W2V)	76.60%	0.0375 *ms/message*
Pure Sentence Embeddings (SBERT)	80.50%	6.9875 *ms/message*
Our solution (Mean of W2V) **α = 0.19**	89.20%	0.83 *ms/message*
Our solution (SBERT) α = 0.43	90.75%	7.975 *ms/message*

As we can see, the solution based on SBERT only is very inefficient and time-consuming. In addition, the Word2Vec only based solution has a relatively low accuracy despite the very fast processing speed. Thus, a good compromise would be using W2V with NER as this approach is 9.6 times faster than the SBERT + NER solution at inference time once the entities dictionary is properly filled. In addition to that, using exclusively sentence embeddings is not possible as named entities are needed in our use case to track the mentioned entities (e.g. projects, clients, persons).

Note that the slow processing time on SBERT is due to the complexity of the model. As it is executed on CPU, the computations take time. In addition to that, we execute inference on messages one by one, and using batching could improve SBERT performance.

The last part of the solution is described in step 3 and explains how the conversation are chosen for comparison.

Processing Method for step 3:

The whole solution will be connected to an external engine able to walk through the communication graph to compare conversation by conversation and identify conversation propagation within the graph.

The diagram below depicts the algorithm proposed in the solution. It describes from the top to the bottom how conversations are identified based on fingerprint similarity and how each conversation list of the persons (nodes) in the communication graph is filled.

As a quick reminder, the communication graph is a graph which model the interactions (communication) between persons. Each person is represented by a node and a link is created between two or more nodes as soon as a conversation has been carried on between these nodes. Several data are associated to each node and each link to characterize these interactions.

For our solution, each node hosts a list of conversations that is empty at the very beginning of the deployment. This list is in fact a dictionary with an ID of conversation as key and a fingerprint as value. Each time, a node A communicate with a node B, a fingerprint is calculated for this communication.

The processing method of step 3 will associate this fingerprint with a conversation ID and update the dictionary depending on the already existence of the conversation. In addition to the lists associated to the nodes, there is a central list which store all conversations ID. A merged fingerprint is associated for each conversation.

The method will first look for similarities of the current conversation with past conversations stored in list of A node. If a similarity is detected, the dictionary is updated so that the fingerprint value for this conversation is now a merge of the previous fingerprint and the new calculated one. This method mimics the evolution of the conversation from A node point of view. A merge is performed by calculating the mean value between both fingerprints. One can imagine a variant with a weighted mean value to give, more or less importance, to the new conversation. On the B side, the method checks if this conversation does not already exist and add it the same way as on A side.

If the conversation is new for A (i.e. not in existing list), then the method will check if the conversation does not already exist in the central list which represents all conversations of the community. If no similarity is detected, a new conversation ID is generated, and a new entry is added in A, B and Central List dictionary with the calculated fingerprint as value. If a similarity is detected, then this conversation ID is used to create a new entry in A list with the calculated fingerprint as value. The method on B side is then the same as described above.

This method ensures an automatic update in real time of all conversation running in a community and give the ability to detect similar conversations of groups which have no communication link between them. To do so, it is just required to walk through the Central List and for each conversation, extract the nodes where this conversation exists. The communication graph will then reveal which nodes are interconnected and which nodes are not.

End of document