Word embeddings

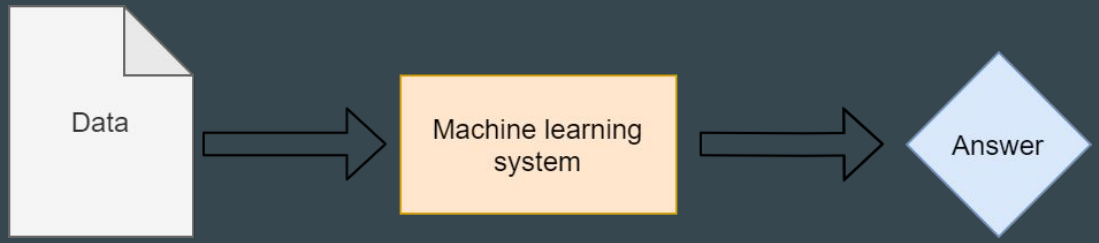
...

Et la topologie du langage

Le contexte :

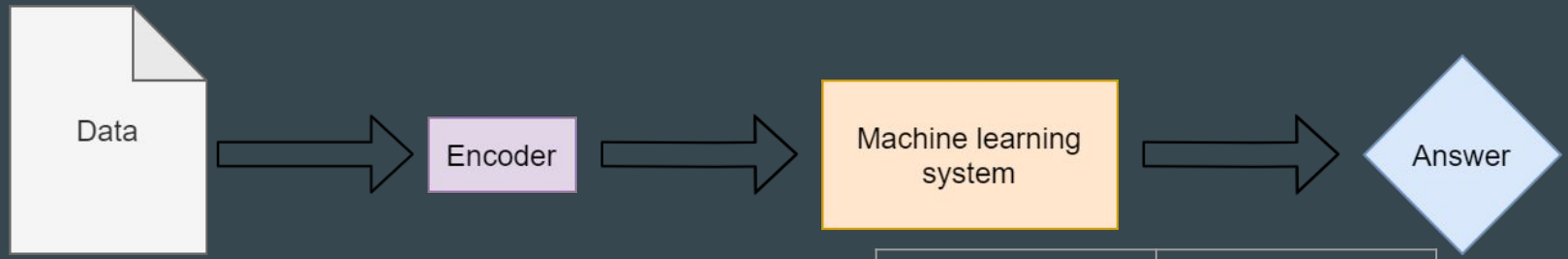
Des chiffres et des lettres

Le problème :



- Appliquer des méthodes de machine learning
- Sur des données textuelles
- Problème : Les lettres n'ont pas de sens mathématique “A” ⊁ “B”

La solution : l'encodage



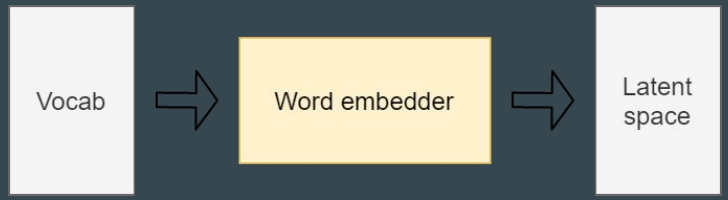
- One hot encoding
- Un vecteur unique pour chaque mot
- Problèmes :
  - Vecteurs linéairement indépendants
    - Aucune interaction entre mots
  - Taille des vecteurs = taille du vocabulaire

Vocabulaire	Vecteurs
système	(1,0,0,0,0,0,...)
données	(0,1,0,0,0,0,...)
langage	(0,0,1,0,0,0,...)
code	(0,0,0,1,0,0,...)
nez	(0,0,0,0,1,0,...)
...	...

Word embeddings

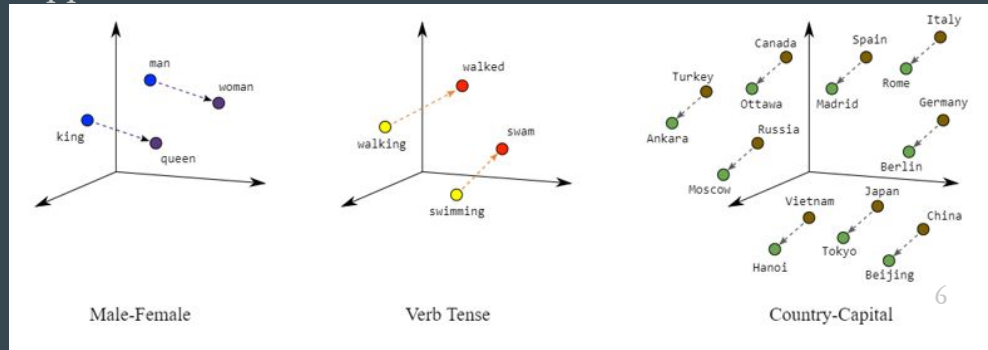
Lier la prose à l'algèbre

La solution : word embedding



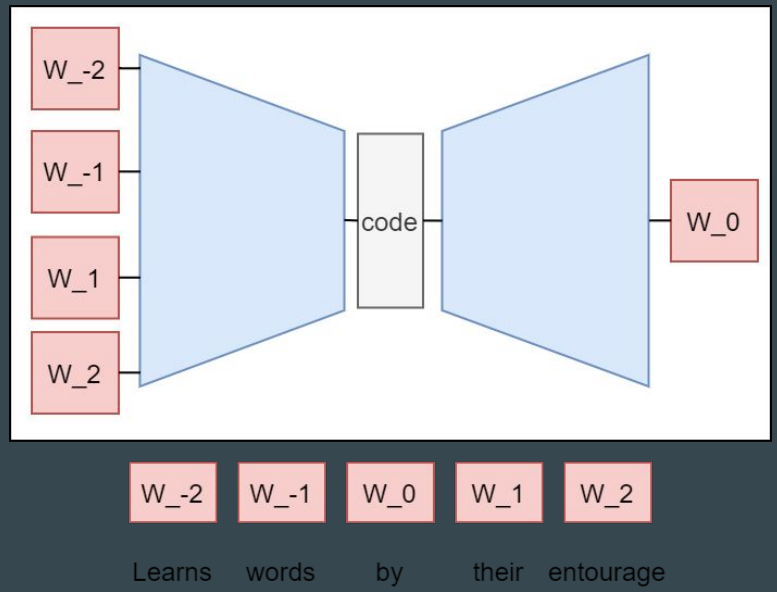
- Devenu essentiel en NLP
- Utilise le one-hot encoding comme entrée
- Renvoi des vecteurs denses de taille fixe (de 50 à 1024)
- De sorte que ces vecteurs ait un sens sémantique
  - Du fait de leurs positions dans l'espace par rapport au autres mots

King - Man + Woman = Queen



Comment ça marche?

- Architecture neuronale de type encoder-decoder
- Language Model (LM) et tâches connexes
  - LM : prédire le mot suivant
  - CBOW : deviner un mot à partir du contexte
- **Données non annotées**
- Entraîné une fois
- Réutilisable à l'infini (**transfer learning**)

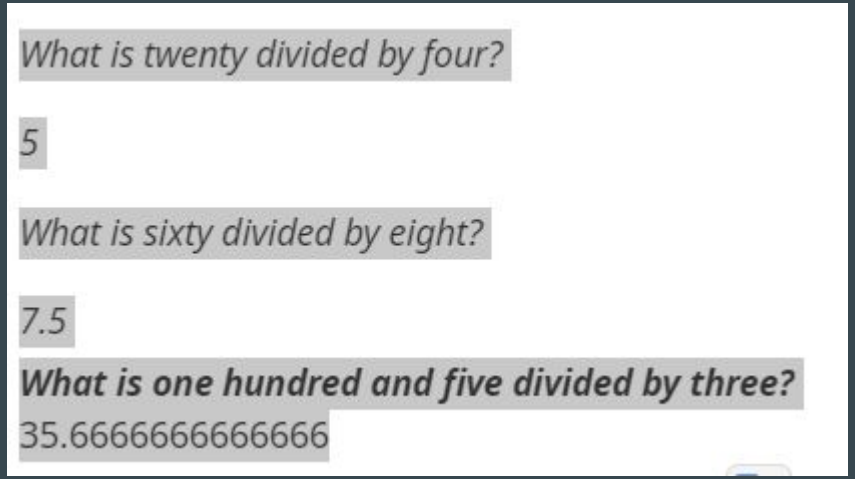


Les familles de word embeddings

- Static (dimension typique : 300):
  - Un vecteurs unique pour chaque mot
  - E.g. GloVe, Word2Vec, FastText
- Contextual (dimension typique : 1024):
  - Pour chaque mot, le vecteur change en fonction du contexte (mot aux alentours)
  - E.g. Elmo, BERT, GPT-*
- Character (dimension typique : 50) :
  - Vois les mots comme des séquence de lettres
  - Robuste aux fautes d'orthographe et aux mots inconnus

GPT-3

- 175 milliards de parameters
- $4.6 million en coût d'entraînement
- 45TB de données d'entraînement
  - Texte brut
  - 20kB = 10k mots ~= 1 article de 20 pages
  - 45TB ~= 2,5 Milliards d'articles
- Peut aussi produire du texte ou discuter
  - N'a jamais appris l'arithmétique directement



Un premier article :

“Is Bigger (Always) Better? Word Embeddings in Event Coreference Resolution.”

Questions de recherche et méthode

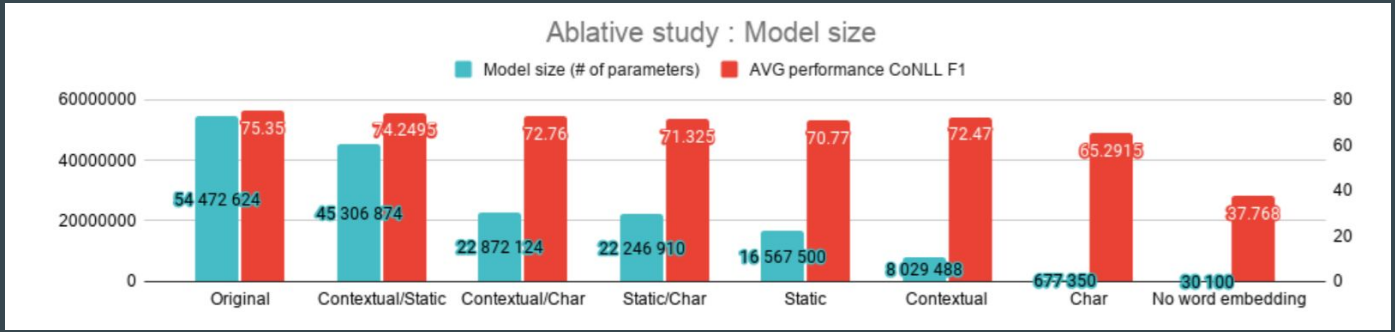
- Premier article soumis à NAACL
- Pose deux questions
  - Trade-off performance vs dimension?
    - Note : la taille du modèle $\propto$ (taille de l'input)^2
  - Performance entre et dans les familles?
- Basé sur la résolution de coréférence d'événements/entités
  - Grouper les phrases et les mots qui réfèrent à la même chose
  - Le SOTA utilise trois types d'embeddings (GloVe, Elmo, Character)
    - Parfait pour étudier nos deux questions
- 16 modèles qui utilise différents mix d'embeddings

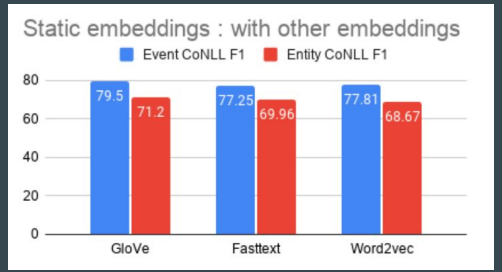
Figure 1: Example of coreferent events and entities.

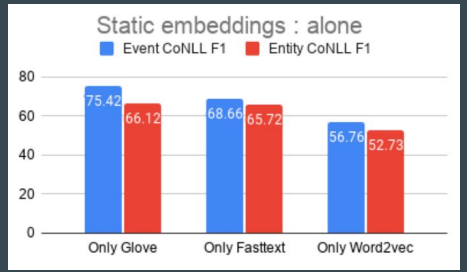
Model	Ctx.	Stat.	Char.
Group 1: Across family study			
Original (2019)	Elmo	GloVe	✓
Contextual/Static	Elmo	GloVe	X
Contextual/Char	Elmo	X	✓
Static/Char	X	GloVe	✓
Static	X	GloVe	X
Contextual	Elmo	X	X
Char	X	X	✓
No word embed	X	X	X
Group 2: Within family study: Static			
GloVe	Elmo	GloVe	✓
Word2Vec	Elmo	Word2Vec	✓
FastText	Elmo	FastText	✓
Only GloVe	X	GloVe	X
Only FastText	X	Word2Vec	X
Only Word2Vec	X	FastText	X
Group 3: Within family study: Contextual			
Elmo	Elmo	GloVe	✓
BERT	BERT	GloVe	✓
GPT-2	GPT-2	GloVe	✓
Only Elmo	Elmo	X	X
Only BERT	BERT	X	X
Only GPT-2	GPT-2	X	X

Resultats

- Une quantité importante d'information mutuelle découverte entre les familles
  - Amènes des rendements décroissants
  - => Le plus petit modèle atteint 86% de la performance du plus large avec 1.2% de sa taille
  - => Les différences de performance sont plus visibles quand on isole les embeddings







Resultats

- Le plus gros modèle converge en moins d'époques
  - Que le plus petit
  - Et est plus rapides à entraîner au total (Paradox)
- Les petits modèles conservent l'avantage en terme de
  - Temps d'exécution de l'inférence
  - Temps d'exécution d'une époque
  - Utilisation de la mémoire
  - Et leurs performances sont proches des plus gros modèles

Conclusion

- Les grands modèles peuvent être plus performants
  - En terme de performance prédictive
  - Mais aussi, paradoxalement, en terme de temps d'entrainement
- Cela dit,
  - L'intégration d'embeddings supplémentaires dans un modèle implique des rendements décroissants
  - Les petits modèles atteignent de bonnes performances et nécessitent moins de mémoire
    - De tels modèles pourraient être plus appropriés lorsque la mémoire ou le calcul (pour l'inférence) sont des facteurs limitants

Des questions?