Word Embeddings in Coreference Resolution.

A Comprehensive Comparison

Judicaël Poumay (judicael.poumay@uliege.be)
Ashwin Ittoo (ashwin.ittoo@uliege.be)

Introduction

Who am I

- Judicael Poumay
- PhD student at the University of Liège
- Research area : Natural Language Processing
    - The study and development of computational techniques aimed at analysing or generating natural language and speech.

My supervisor and grandsupervisor!

Today's topic

- A paper published in EMNLP 2021

"A Comprehensive Comparison of Word Embeddings

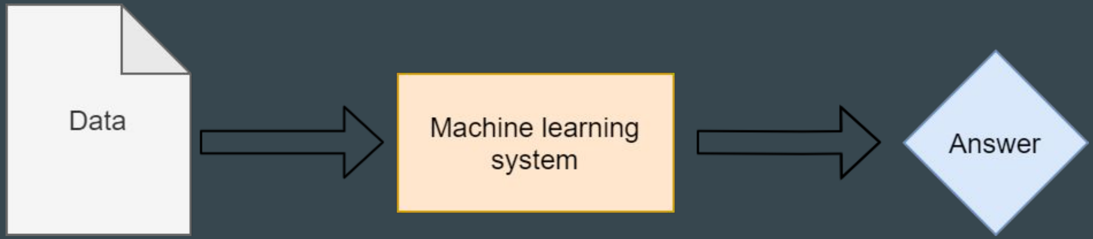
in Event & Entity Coreference Resolution"

Outline

- Outline
  - Word embeddings
  - Coreference resolution
  - Previous research
  - Results

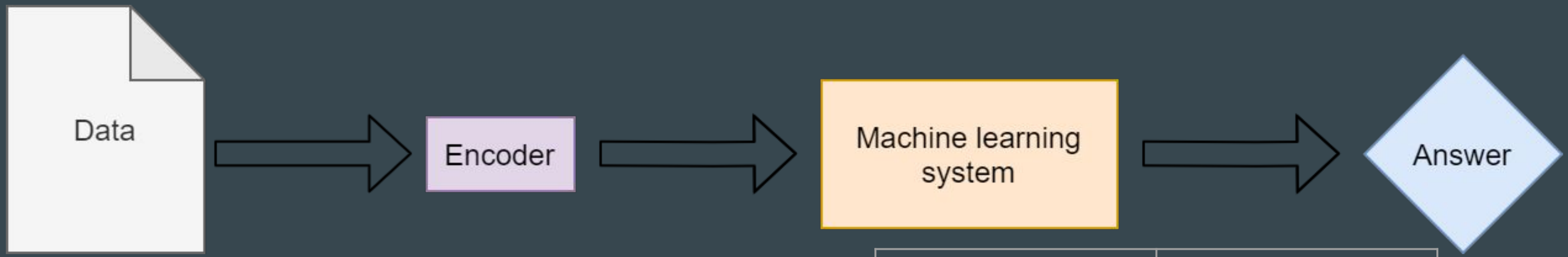
Word embeddings

The issue



- In NLP, we wish to feed textual data to machine/deep learning models
- Hence, we need a way to represent words
- To manipulate them algebraically

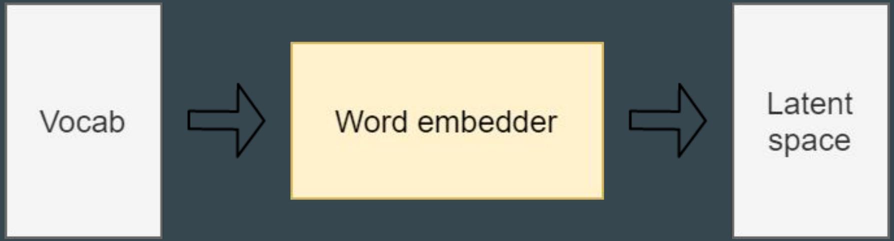
A trivial solution : one hot encoding



- Each word has one representation
  - Basically using dummy variables
- Issue :
  - Linearly independent vectors
    - Meaning algebraic relationships are not possible
  - Size of the vector = size of vocabulary

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,...)
...	...

A better solution : word embeddings

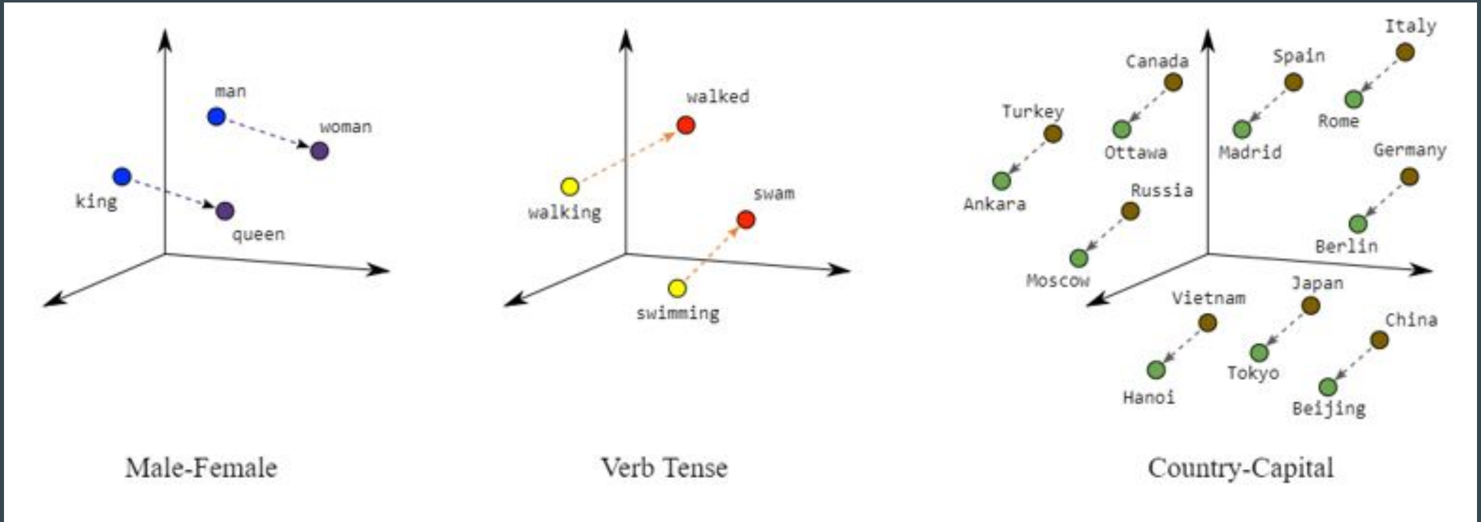


- An essential tool in NLP
- Used to compress one hot encoding
- Produces fixed size vectors called word embeddings

- Trained once on large datasets
- Reusable for many downstream tasks (**huge transfer learning ability**)

A better solution : word embeddings

- The resulting vectors carry semantic information
  - Due to the learned topology of the latent space where the embeddings lies



Linear algebra with words : The semantic information is encoded in the algebraic relationships between the words

Any questions?

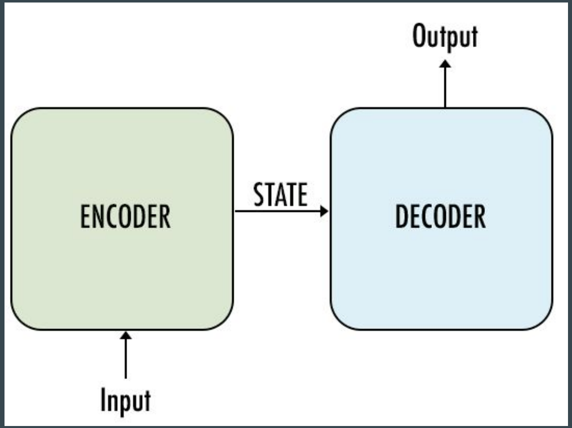
How does it work?

- Distributional semantic hypothesis
  - “A word is characterized by the company it keeps”
  - Given a word, we can study the distribution of words that tend to surround it
    - We call this the context of that word
- The hypothesis states that each word has a distribution P(context | word)
  - To create word embeddings, we learn to approximate this distribution

- Example the phrase “A _ meowing loudly”
  - We can predict that the missing word is *cat* considering the verb *meowing*
  - These words tend to appear together

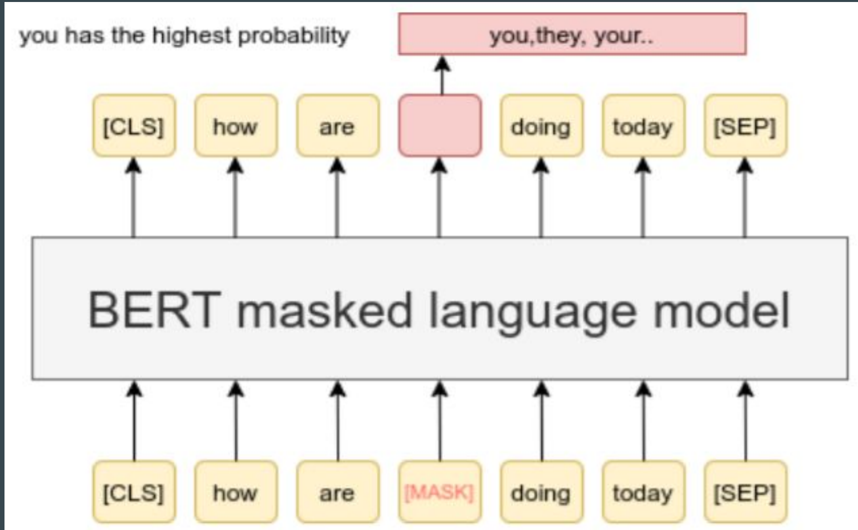
How does it work?

- We train an encoder-decoder architecture
  - Used for machine translation, sequence to label sequence, ...
- These architecture produce encodings as a byproduct of their training
- Both encoder and decoder are trained simultaneously
  - The link between the encoder and decoder is where the magic happens



How does it work?

- Tasks : language modelling
  - Using encoder-decoder
  - Predict the correct word given a context
  - Unsupervised learning (no labelled data required)
- Once trained, we can find the embedding of a word in the link between encoder & decoder



Any questions?

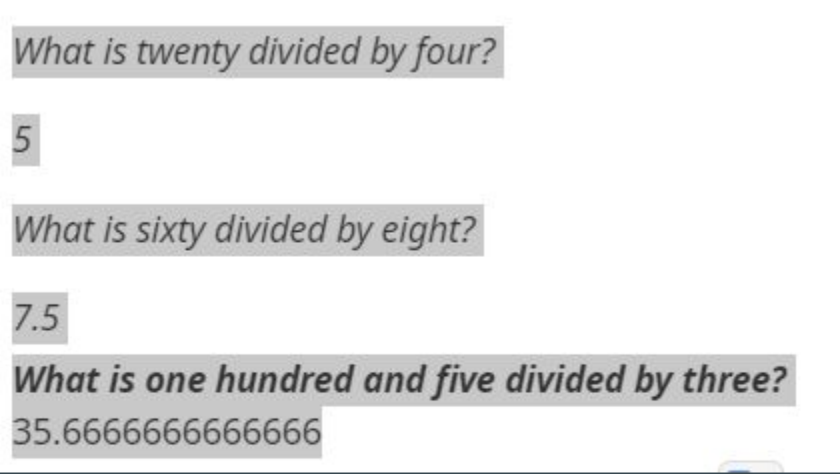
Word embeddings families

- Static (dimension typique : 300):
  - **Each word has a unique encoding**
  - E.g. GloVe, Word2Vec, FastText
- Contextual (dimension typique : 1024):
  - **The encoding of words changes depending on their context**
  - More powerful as it can help with polysemy
  - E.g. Elmo, BERT, GPT-*
- Character (dimension typique : 50) :
  - **Works at the character level instead of taking words as atomic blocks**
  - Robust to spelling mistakes and unknown words
  - However, many words are one letter away from another (e.g. “cat” and “can”)
    - Especially small ones

Example
- Let's have a drink in the bar
- I have to study for the bar
- Bring me a chocolate bar

GPT-3

- 175 billion parameters
- $4.6 million training cost
- 45TB dataset
  - 2.5 billion articles
- Trained to generate text
  - Learned a rudimentary arithmetic understanding



Coreference resolution

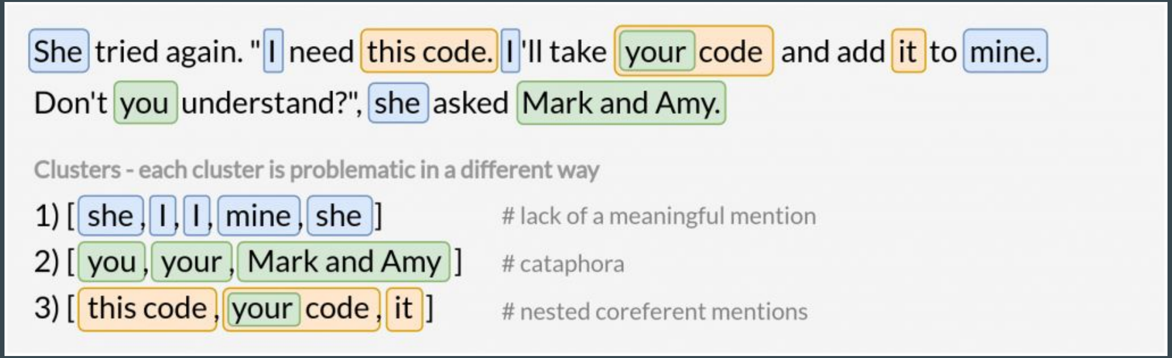
What are corefences

- Coreferences are different words or phrases that refer to the same thing.

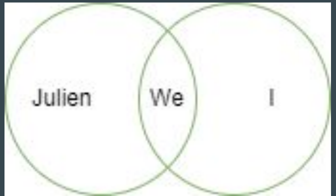
- Two kinds of coreferences
  - Entity (usually a term)
  - Event (usually sentences)
- Two settings for resolution
  - Within documents
  - Across documents

The task of coreference resolution

- The goal is to cluster corefering mentions or phrases



- There are many challenges
  - Defining what constitutes an entity or event
    - Many datasets disagree
  - Nested mentions
  - Overlapping coreference : Julien and I, We would like



Applications of coreference resolution

- Applications include a wide array of downstream tasks
  - Machine translation
  - Chatbots
  - Question answering
  - Text summarization
  - Natural language understanding
  - ...

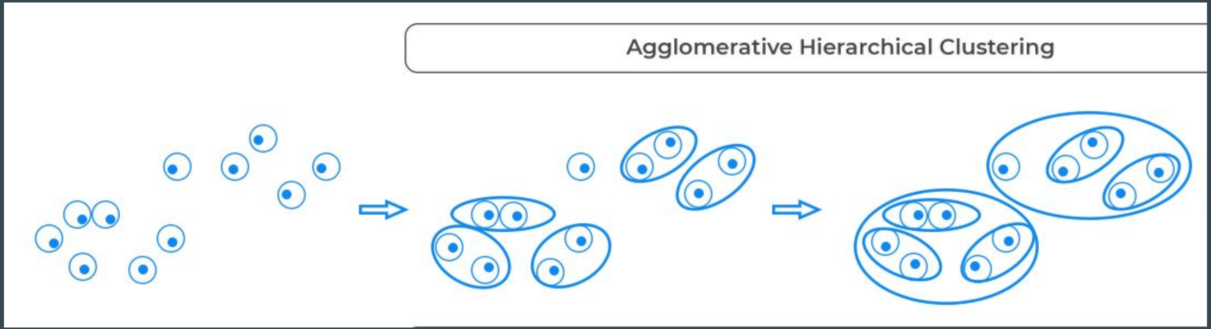
Previous research

The state of the art

- Barhom 2019 : Revisiting joint modeling of cross-document entity and event coreference resolution
- This paper proposed a simple method to perform
  - event and entity
  - within and cross document
  - coreference resolution
- First paper to work on both kind of coreferences in both settings
- They achieved state of the art performance
  - On a dataset called ECB+ which is a standard in this field

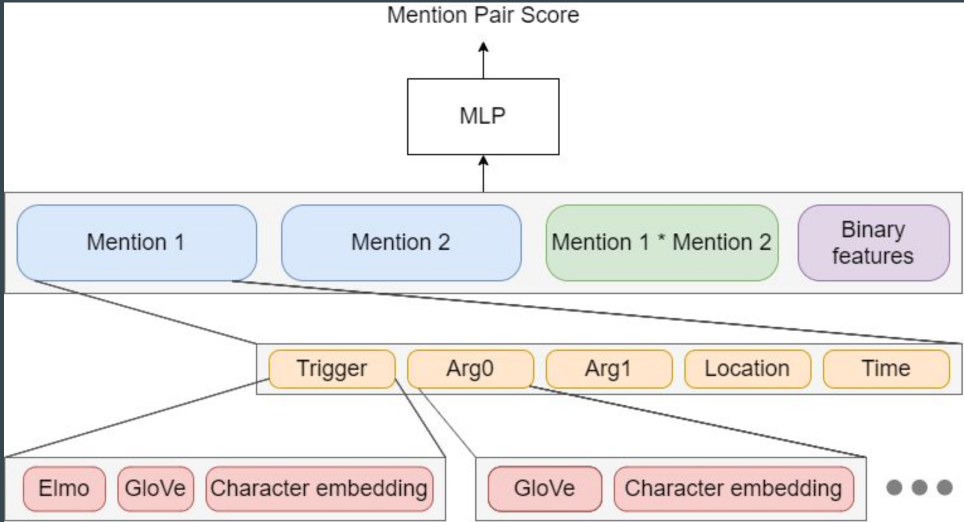
The state of the art

- The model consists of a shallow neural network
  - Trained to cluster events & entities
  - Agglomerative clustering
  - They task is two compare two mentions (or mentions clusters) and decide if they should be merged
    - If the network output > 0.5 , the mentions (or mentions clusters) are merged



The state of the art

- The complexity of the model is in its input
  - Each mention is represented with multiple word embeddings
    - Elmo as a contextual embedding
    - Glove as a static embedding
    - A character embedding
  - In total the input has a dimension of 8522
    - Making the network extremely wide
    - But also shallow : 2 layers



Any questions?

Our research

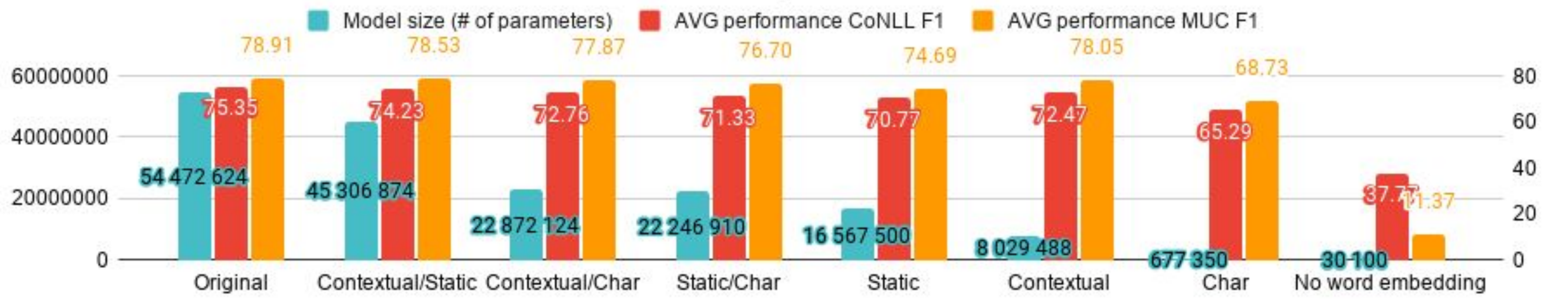
Introduction

- Our investigation
  - How various combination of embeddings perform?
  - How do embeddings compare?
    - Within families (static, contextual)
    - Across families (static, contextual, character)
- Embeddings studied
  - Static : GloVe, Word2Vec, FastText (dim : 300)
  - Contextual : ELMo, BERT, GPT-2 (dim : 1024)
  - Character : CNN based (dim : 50)
- We derived 16 models from the original
  - Using various embeddings combination

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

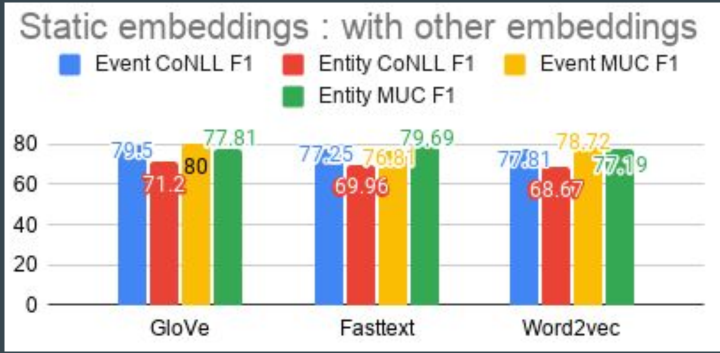
Results - Diminishing returns

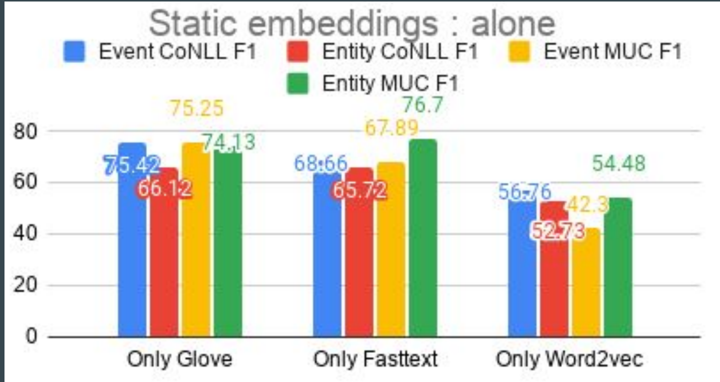
- Ablation analysis : removing embeddings from the original model
  - Model size ~= input² and input = sum of the length of the word embeddings used
- Contextual model (Only Elmo)
  - achieves 96% of the Original model performance
  - with 14.7% of its size
- Character model (Only character embedding)
  - achieves 86% of the Original model performance
  - with 1.2% of its size
- Using more than one embedding significantly but modestly increase performance



Results - Comparing static embeddings

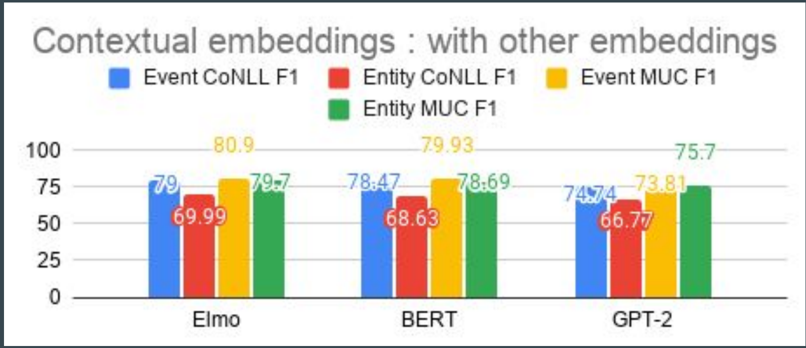
- Varying static embeddings in the original model
  - No clear difference
- Varying static embeddings alone
  - Word2Vec is clearly the worse
  - GloVe works best in Event CR
  - FastText works best in Entity CR

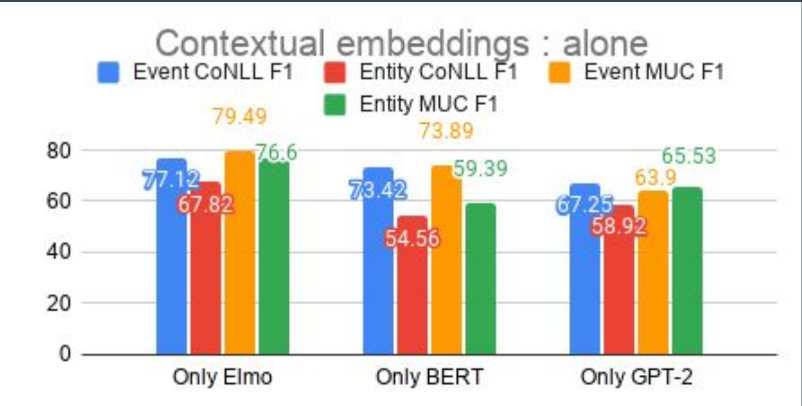




Results - Comparing contextual embeddings

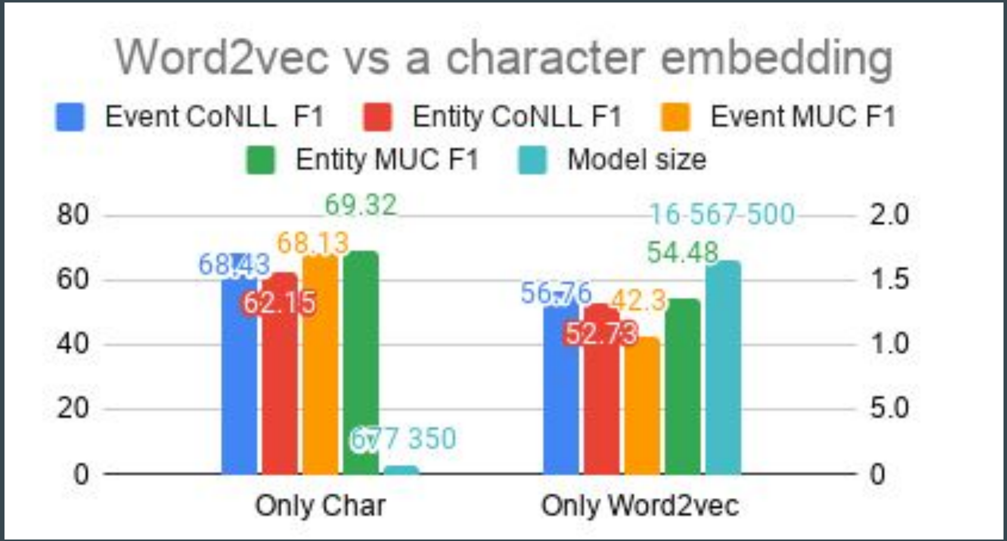
- For a similar reason, we compare them alone
- Elmo is the best overall





Results - Word2vec

- Character embedding > word2vec
  - Not only is the character embedding more accurate
  - It leads to a radically smaller model (~24x)



Results - Runtime

- Original model vs character embedding model
  - Character embedding model is faster to train per epoch and at test time
  - However the original model requires fewer epoch to converge and overall took less time to train

Conclusion

- We can get SOTA performance by bloating a model with every possible embeddings
  - However, we get diminishing returns
  - Contextual model (Only Elmo)
    - achieves 96%of the Original model performance
    - with 14.7% of its size
- When comparing embedding, it is best to isolate them
- Some embeddings are better suited for a specific task
  - GloVe works best in Event CR
  - FastText works best in Entity CR
- Word2vec underperform compared to a character embedding
- Bigger model can converge faster than smaller ones

The end : Any questions?