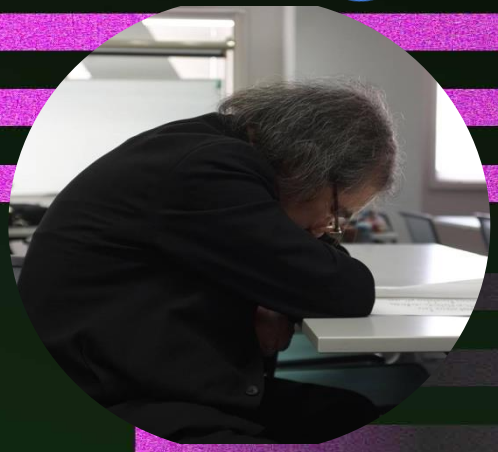
AIは意味をどのように扱っているのか？
–– ChatGPT の不思議 ––



Agenda

はじめに：ChatGPTに対するGoogleとMicrosoftの反応

Part 1：Google検索と意味理解

Part 2：意味理解への様々なアプローチ

Part 3：意味の分散表現論の登場

Part 4：意味の分散表現論の発展

はじめに

ChatGPTに対する
GoogleとMicrosoftの反応

Google vs. Microsoft

今回のセミナーのテーマは、AI技術が「意味」をどう扱っているのかということなのですが、ChatGPTの登場が、IT技術者を驚かせただけでなく、業界の再編につながる可能性をもつインパクトをIT業界に与えていることを、最初に紹介しようと思います。

21世紀初頭からITの大躍進を支えたGoogleの検索+広告というビジネスモデルに、OpenAIへの巨額の投資を通じてMicrosoftが挑戦する姿勢を明確にしています。

GoogleでChatGPTが「コード・レッド」を引き起こしたと報告

OpenAIが開発したAIチャットボットChatGPTは、ありとあらゆるクエリに直接回答できることから人気を博しましたが、水曜日(2022/12/21)のNew York Timesの報道によると、どうやらGoogleで警鐘が鳴らされているようです。

Timesが話を聞いたGoogleの幹部は、ChatGPTのようなAIチャットボットは、Google検索で見つけた広告や電子商取引に大きく依存している検索大手のビジネスを根底から覆す可能性があると述べています。

https://www.cnet.com/tech/services-and-software/chatgpt-caused-code-red-at-google-report-says/

タイムズが入手したメモと音声録音の中で、同誌はスンダー・ピチャイCEOが「グーグルのAI戦略を定義する」ための会議に出席し、「ChatGPTがもたらす脅威に対応するため、社内の多数のグループの作業をひっくり返した」と述べている。"

Googleはコメントを求めたところ、すぐに返答はありませんでした。

ChatGPTは、ネット上にある利用可能なデータを使って、多くの質問に対して会話形式で回答を与えるAIチャットボットです。Google検索が、インターネット上にある潜在的な答えを探し出し、その答えを読み解くためのリンクを提供するのとは異なり、ChatGPTは、これまで人間が書いたことのないユニークで斬新な答えを提供します。例えば、「ピカチュウがスパムむすびを食べるという俳句を作ってください」と依頼すれば、ChatGPTは納得のいく答えを返してくれます。

タイムズ紙が閲覧したメモによると、グーグルは独自のAI技術を積極的に構築している一方で、社会にどのような影響を及ぼすかを恐れて、一般への公開を遅らせているそうです。

その理由のひとつは、回答が現在オンラインで入手可能な人間が作ったデータに基づいていること。つまり、人種差別や偏見、誤った情報がチャットボットの学習モデルに入り込み、不愉快な答えを与えてしまう可能性があるということです。2016年にマイクロソフトがAIチャットボットを導入して間もなく、急遽オフラインにせざるを得なかったのはそのためです。Facebookの親会社であるMetaもチャットボットを導入しましたが、すぐに人種差別的な答えを出すようになりました。

当面、グーグルは、広告とeコマース販売で稼ぎ、前四半期の収益の8割近くを占めた検索事業に依存し続ける。チャットボットは自然言語で回答を与えることを目的としているため、広告を統合することは難しいかもしれない。

それだけでなく、膨大なデータプールをスクラビングして信憑性のある回答を提供するために必要な処理は、高価になる可能性があります。ある試算によると、OpenAIは1ヶ月あたり300万ドルを費やしており、これはChatGPTがまだベータ版で、人々がアカウントを作る必要があり、高負荷でオフラインになることもある状態での話です。それでも、Google独自のLaMDAのような大規模な言語モデルに基づいて構築されたチャットボットが普及すれば、大学の小論文を殺す（あるいは大学の小論文には最適）だけでなく、Googleの主な収益源も殺すことになるかもしれない。

Google LaMDA: our breakthrough conversation technology

But the most important question we ask ourselves when it comes to our technologies is whether they adhere to our AI Principles. Language might be one of humanity's greatest tools, but like all tools it can be misused. Models trained on language can propagate that misuse — for instance, by internalizing biases, mirroring hateful speech, or replicating misleading information. And even when the language it's trained on is carefully vetted, the model itself can still be put to ill use.

https://blog.google/technology/ai/lamda/

しかし、私たちのテクノロジーに関して最も重要なのは、「AI原則」を遵守しているかどうかということです。

言語は人類にとって最も偉大な道具のひとつかもしれませんが、あらゆる道具と同様に誤用される可能性があります。例えば、偏見を内在化させたり、憎しみのこもった話し方を反映させたり、誤解を招く情報を複製したりすることで、言語に基づいて学習したモデルが誤用を広めてしまう可能性があるのです。また、学習させた言語が慎重に吟味されたものであったとしても、モデル自体が悪用される可能性があります。

Our highest priority, when creating technologies like LaMDA, is working to ensure we minimize such risks. We're deeply familiar with issues involved with machine learning models, such as unfair bias, as we've been researching and developing these technologies for many years. That's why we build and open-source resources that researchers can use to analyze models and the data on which they're trained; why we've scrutinized LaMDA at every step of its development; and why we'll continue to do so as we work to incorporate conversational abilities into more of our products.

LaMDAのような技術を作る場合、そのようなリスクを最小化することを最優先しています。

私たちは、長年にわたり機械学習技術の研究開発を行ってきたため、不当な偏りなど機械学習モデルの問題点を熟知しています。だからこそ、研究者がモデルや学習データの分析に利用できるリソースを構築してオープンソース化し、LaMDAの開発の各ステップで精査してきたし、会話能力をより多くの製品に取り入れるために今後もそうしていくのです。

Microsoftのナデラ氏、OpenAIでGoogleに新たな狙いを定める

2023/01/11 Bloomberg

Microsoft Corp.のSatya Nadella氏は、検索やアプリなどの分野に高度な人工知能ツールを織り込む競争において、このソフトウェアの巨人がGoogleに対して優位に立つことを目指し、同社にとって過去最大の新興企業投資を検討しているようだ。

同社の計画に詳しい関係者によると、同社は、話題のAIボット「ChatGPT」を開発したOpenAIに100億ドル規模の投資を行う方向で協議を進めているとのことです。検討中の案では、最終的な条件は変更される可能性があるものの、マイクロソフトが数年にわたって現金を注入することになっていると関係者は述べている。

https://www.bloomberg.com/news/articles/2023-01-10/microsoft-s-nadella-takes-fresh-aim-at-google-with-openai-talks

マイクロソフトはすでにOpenAIのパートナーであり、2019年にこのスタートアップに10億ドルを投資している。

ワシントン州レドモンドに本拠を置く同社は現在、開発者向けに言語AIを使用して、同社のGitHub部門のプログラミングツール「Copilot」に自動化を加えており、こうした技術を同社の検索エンジン「Bing」やOffice生産性アプリケーション、Teamsチャットプログラム、セキュリティソフトウェアに加えたいと考えている。このソフトウェアメーカーは、OpenAIの画像作成ツール、同じくバイラルヒットとなった「DALL-E」をデザインソフトウェアに投入している。

検索業界では長い間、基本的に手出しができない存在だったグーグルが突然弱さを見せたため、ナデラCEOはこの関係を強化しようとしている。

アルファベット・インクは、検索エンジンを使って特定のキーワードを検索し、どの情報が有益かをユーザーに判断させるというモデルを採用している。

これに対し、ChatGPTは政治やコンピュータ・プログラミングなどのトピックに関する質問に詳細な説明で応え、その質問と回答の形式により、ユーザーは完全に理解するまで掘り下げていくことができる。また、Google検索のような青色のリンクの羅列ではなく、会話や質問に対する回答など、人間らしい自然な応答が可能である。

Part 1

Google検索と意味理解

- Google検索と「知識表現」-- Knowledge Graph
- EntityモデルのFolksonomy
- Google検索とChatGPT

検索＋広告技術と意味理解はつながっている

この章で強調したいことは、Googleの検索+広告というビジネスモデルを支える技術は、言語の意味理解と繋がっていると言うことです。

それを意味のEntityモデルといいます。また、検索者の「意図」を知ることにも、Googleは強い関心を持っていました。

しかし、そのEntityモデルを作っているのは人間でした。このEntityモデルのFolksonomyは、AIによる自律的な意味理解とは異なるものです。

ただ、僕は、ChatGPTでの「人間のフィードバック」重視は、それと同じ弱点を持つのではと考えています。

Google検索と「知識表現」
Knowledge Graph

Google検索技術の変化

Googleの検索技術を、クローラーが収集した膨大なWeb上のテキストデータに対する「文字列検索」と、その結果のPage Rankに基づく表示の順序付けと、それへの広告の紐付けと考えるのは、少し古い見方である。

Googleの検索技術は、2010年の前後に大きく変化している。

先行したのは、2008年にアメリカでサービスが始まった「音声検索」サービスである。ちなみに、日本でのGoogle音声検索のリリースは、2009年で、世界で二番目に早かった。

2012年、Googleは Knowledge Graphという新しい検索技術をスタートさせる。

Google Knowledge Graphに基づく新しい検索の目標

- 正しい「もの」を見つける。(Find the right thing)
- 最良の要約を得る。(Get the best summary)
- さらに深く、さらに広く。(Go deeper and broader)

知識のグラフでの表現と検索

- 例えば、「アインシュタインの誕生日」を知ろうとすれば、以前の文字列ベースの検索では、「アインシュタイン」を検索して、検索エンジンが返す膨大な数の（PageRankで順序付けされているのだが）文字列「アインシュタイン」を含むURLのドキュメントを、検索者が確認して、その中で彼の誕生日の情報を見つける必要がある。
- グラフ・ベースの検索では、Person Entityノードであるアインシュタインの Property **birthdate** の値をしらべれば、それはすぐにわかる。
- 同様に、Person Entityのもつ Propertyである、award, colleague, jobTitle, worksFor などの値を取得すれば、もっと多くの情報が簡単に得られる。

知識のグラフでの表現と検索
Entity Model

- Knowledge Graph は、”people”, “places”, “things” といった実世界のエンティティを記述する数百万のエンティティを持っている。これらのエンティティが、このグラフのノードを形成する。
- これらのエンティティの「ノード」と、ノード間の関係あるいはアクションを表す「エッジ」のボキャブラリーは、次に見る組織 Schema.orgによって、与えられている。

こうしたアプローチをEntity Modelという

Welcome to Schema.org から

Schema.org は、インターネットやWebページ、e-メールのメッセージやさらにもっと多くのものの上にある構造化されたデータに対して、スキーマを作りあげ、維持し、それを広めるというミッションを持った共同のコミュニティ活動である。

Schema.org は、**Google, Microsoft, Yahoo, Yandex** のスポンサー支援を得ている。そのボキャブラリーは、メーリングリスト public-schemaorg @w3.org と GitHub https://goo.gl/WvCyP3 を用いて、オープンなプロセスで開発されている。

folksonomyアプローチ

http://schema.org/

エンティティの型（Type）のサンプル

Types = “is-a”

- Book
- BookSeries
- EducationalOrganization
- Event
- GovernmentOrganization
- LocalBusiness
- Movie
- MovieSeries
- MusicAlbum
- MusicGroup
- MusicRecording
- Organization
- Periodical
- Person
- Place
- SportsTeam
- TVEpisode
- TVSeries
- VideoGame
- VideoGameSeries
- WebSite

is-a → Person
/people/person

is-a → Film Actor
/film/actor

is-a → Politician
/government/politician

One entity, many types.

一つのエンティティは、多くの型を持つ

Entityは階層を持つ
schema.org での Entityの階層の例

- Thing > CreativeWork > Book
- Thing > CreativeWork > CreativeWorkSeries > BookSeries
- Thing > Organization > EducationalOrganization
- Thing > Event
- Thing > Organization > GovernmentOrganization
- Thing > Organization > LocalBusiness
  Thing > Place > LocalBusiness
- Thing > CreativeWork > Movie

schema.org でのPersonのProperty

Properties = “has-a”

- gender
- birthdate
- birthplace
- profession
- nationality
- ethnicity
- parents
- children
- religion
- education
- jobTitle
- **follows**
- **knows**
- **owns**
- **seeks**

... and more

Action

Google Knowledge Graph Search API

- The Knowledge Graph Search API lets you find entities in the Google Knowledge Graph. The API uses standard schema.org types and is compliant with the JSON-LD specification.

- “Taylor Swift” についての検索サンプル

```
https://kgsearch.googleapis.com/v1/entities:search?query=taylor+swift&key=API_KEY&limit=1&indent=True
```

```
{
 "@context": {
  "@vocab": "http://schema.org/",
  "goog": "http://schema.googleapis.com/",
  "resultScore": "goog:resultScore",
  "detailedDescription": "goog:detailedDescription",
  "EntitySearchResult": "goog:EntitySearchResult",
  "kg": "http://g.co/kg"
 },
 "@type": "ItemList",
 "itemListElement": [
  {
   "@type": "EntitySearchResult",
   "result": {
    "@id": "kg:/m/0dl567",
    "name": "Taylor Swift",
    "@type": [
     "Thing",
     "Person"
    ],
    "description": "Singer-songwriter",
```

```
"image": {
        "contentUrl":
```

知識のグラフによる表現とそれに対する検索は意味理解の一つの形である

「テーラー・スイフトの年齢は？」という問いに対して、「33歳」と答えることは、「テーラー・スイフトの年齢は？」という文の意味を理解していることになる。

「I lve you.」の意味は、「私はあなたを愛している」という意味である。（翻訳による意味理解）

命令文や、疑問文の「意味」を考えると、こうした意味のPragmaticな解釈は、ある意味では自然なものだ。多くのプログラムの意味論は、こうした形をとる。

しかも、こうした知識の表現の構造は利用者の言語に依存しない。

ボイス・アシスタントの意味へのアプローチ
Intentモデル

ボイス･アシスタントでは、人間の側の発話の意図を機械が把握し、その意図に応えて正しく反応するのが、機械にとっての「意味理解」であるという、意味のPragmatic(「語用論」的)な解釈が取られます。

命令文 ー 命令された動作を行えばいい
疑問文 ー 答えを与えればいい ー＞検索を実行

ボイス･アシスタントの代表的な製品である Alexa / Echo では、人間の発話の意図は、Intent という一つの文字列で表現されます。Intentは、基本的には、機械に対する人間からの命令を簡潔に示したものです。こうした単純な意味モデルをIntentモデルと呼びます。

EntityモデルのFolksonomy

GoogleのKnowledge Graph

GoogleのKnowledge Graphは、情報をEntityの構造に対応させる技術です。あるものの「意味」は、対応するEntityが持つ「情報」として解釈されます。

Entityは、現実のものであれバーチャルなものであれ、具体的なものであれ抽象的なものであれ、概念としては「実在するもの」とみなされます。

ある種の「概念実在論」と言っていいと思います。

Entityモデル

Entityモデルでは、Entityとその関係の構造が表現するものは実在的なものの構造の反映です。

こうした関係をEntityの「存在論 “Ontology” 」といいます。少し大袈裟な気もしますが。

ですから、Entityを用いれば、具体的な言語表現からは独立な、かつ、全ての言語で共通な情報を与えることが可能です。

（もっとも、Entityとその階層を定義しているSchema.org は、Entityを英語の名前で定義しています。）

Entityモデルの弱点

ただ、そうした理念に導かれたEntityモデルですが、現実的な弱点がありました。

それは、Entityの階層にしろ、それぞれのEntityの持つPropertyにしろ、それらは決してアプリオリに与えられるものではなく、それを人間が経験的に必要に応じて決めざるを得なかったことだと僕は考えています。

folksonomy

それは、「folksonomy (分類の仕方はみんなで決める)」というEntityモデル構築の実践的方法論の中核に関わる問題です。

folksonomyという指針がなければ、Entityモデルは発展しなかったのは確かだと思いますが、それは、本来の「存在」を客観的に反映した意味の世界を網羅的につくるというより、直接的にはネット上の人間の活動の欲求を反映した、平凡で陳腐な分類作業が中心になっていったように思います。

schema.orgでのEntityとその階層の例

- Thing > CreativeWork > Book
- Thing > CreativeWork > CreativeWorkSeries > BookSeries
- Thing > Organization > EducationalOrganization
- Thing > Event
- Thing > Organization > GovernmentOrganization
- Thing > Organization > LocalBusiness
  Thing > Place > LocalBusiness
- Thing > CreativeWork > Movie

schema.orgでのEntityとその階層の例

- Thing > Place
- Thing > Organization > SportsOrganization > SportsTeam
- Thing > CreativeWork > Episode > TVEpisode
- Thing > CreativeWork > CreativeWorkSeries > TVSeries
  Thing > CreativeWork > TVSeries
- Thing > CreativeWork > Game > VideoGame
  Thing > CreativeWork > SoftwareApplication > VideoGame

EntityモデルとIntentモデルの進化

Entityモデルの変化

Entityモデルの変化で、特徴的な出来事は、ある時点で Entity の一つとして、ActionというEntityが導入されたことです。

Entityの世界は、静的な世界で動きがありません。例えて言えば「名詞」しかない言語の世界です。そうした言語で、世界の意味の記述は不十分にしかできないのは明らかです。

Intentモデルの変化

一方、当初のIntentモデルは、動的な、いわば「動詞」だけからなる、もっと言えば、「動詞の命令形」だけからなる世界でした。

ただ、世界がどんなにダイナミックな運動の中にあるとしても、動詞だけの言語で、ましてや「命令」で世界を記述することはできません。

Intentモデルの変化

Intentモデルの最初の変化は、「スロット」の導入でした。それは、「動詞」の「目的語」として「名詞」の導入にあたるものです。

Intentモデルの第二の変化は、「Intent Signature」の導入で、当初は「動作命令」の文字列ラベルに過ぎなかったIntentに、Entityモデルを援用しながら「文」の性格を与えるものです。

Intentモデルの変化

こうした単純なものから複雑なものへのIntentモデルの「進化」は、意味の把握を追求しようとして人工言語モドキの中でおきた「進化」なのですが、とても印象的なものでした。

そこでは、EntityモデルとIntentモデルは融合しています。ただ、その過程を複雑なEntityモデルが、単純なIntentモデルを吸収したと捉えることはできないと僕は考えています。

Google検索とChatGPTについて

Google検索に求められる変化

前回、Googleの検索が、ChatGPTに置き換わることは現状では難しいだろうとと述べたのですが、Google検索技術の支柱の一つであるEntityモデルが、後発のIntentモデルの進化系より、優れているようには思えません。

Googleの検索技術の変化が求められているのは、確かなことだと思います。

そもそも、人力だけで、網羅的な知識のEntityモデルを作るのは可能とは思えません。

GPT-3 の能力

そもそも、人力だけで、網羅的な知識のEntityモデルを作るのは可能とは思えません。

その点、GPT-3のような大規模言語モデルは、人間の言語活動が生み出した、大量の「構造化されていない（Entityなどには未だ分類されていない）」生のテキストを直接機械に与えます、

奇妙なことに、機械はそこから「何か」を見つけ出します。

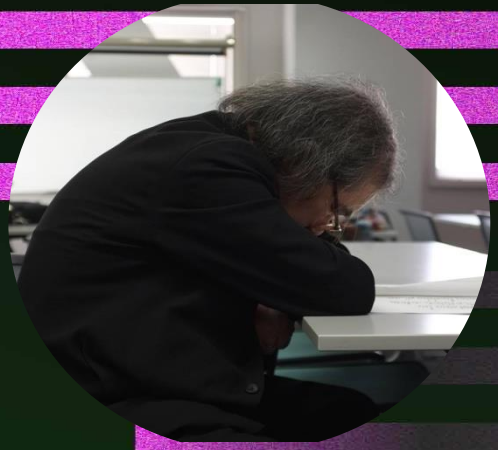
（この「何か」が何であるのかを、あらかじめ予想することができないことが問題なのですが。）

ChatGPTとGPT-3の違い

ただ、ChatGPTとGPT-3は、違うシステムです。

「人間のフィードバック」に基づくというGPT-3とは決定的に異なるChatGPTの独自のアプローチが、ある種の形を変えたfolksonomyの導入として、システム自体の陳腐化を引き起こすことを僕は心配しています。

そのあたりの問題は、あらためて考えてみようと思います。



Part 2

意味理解への様々なアプローチ

- 文と意味の構成性Compositionality から問題を整理する
- 様々な言語・意味モデルの比較

言語・意味モデルの成功と失敗

ここでは、いくつかの言語・意味モデルを比較したいと思います。

基本的な評価の観点は、「文と意味の構成性 Compositionality」と、「語と文の、言語をまたいだ意味の共通性」の二つです。あわせてFolksonomy的な人力を介さないことも重要な論点です。

大きな成功を収めた大規模言語モデルは、構成的なアプローチを全体としては欠いています。ただ、それらは、語のレベルでも文のレベルでも、「意味の分散表現」をとっています。

このセミナーでは、「意味の分散表現」にフォーカスして、AIでの意味の扱いを追いかけようと思います。

大規模言語モデルは、
意味をどのように扱っているのか？

今回のセミナーのタイトルを、「AIは意味をどのように扱っているのか？」にしました。ただ、注意してもらいたいことがあります。それは、ここでいう「AI」のコンセプトは狭いものだということです。

ここでの「AI」は、基本的には、Googleニューラル機械翻訳に始まり、BERT、GPT-3、ChatGPTへと続く、現在の「大規模言語モデル」のことを指しています。

正確に言えば、今回のセミナーのテーマは、「大規模言語モデルは、意味をどのように扱っているのか？ 」というものです。

大規模言語モデルは、自然言語の意味を扱うことに成功した、最初のモデル

では、なぜ、意味の理解に大規模言語モデルは、重要なのでしょうか？それに答えるのは比較的容易です。

人間は、自然言語を扱う理論をいろいろ作ってきたのですが、それに基づいて実際のコンピュータ上で、意味を含んだ自然言語を扱うシステムを実装することには、なかなか成功しませんでした。

大規模言語モデルが重要なのは、それがコンピュータ上で自然言語の意味を扱うことに成功した、事実上最初の、そして現状では唯一の実装されたモデルだからです。

大規模言語モデルの成功の出発点は、語の意味の多次元ベクトルによる表現

大規模言語モデルの出発点となったのは、「語の意味の多次元ベクトルによる表現」でした。

今回のセミナーの、第二部「意味の分散表現の登場」、第三部「大規模言語モデルの成立」という構成は、今回のセミナーの主要な目的が、大規模言語モデルの成功を跡づけようとしていることを示しています。

ただ、この第一部「意味理解への様々なアプローチ」では、大規模言語モデル以外の、意味理解のアプローチを紹介しようと思います。

文と文の意味の基本的特徴

文と意味の「構成性（compositionality）」

- 文は文法に従って語から構成される
  文のレベルでは、その構成性は文法によって担われる

- 文の意味は、語の意味に依存して構成される
- 文の意味は、文の文法的構成に依存する
  - 文法的に正しいが意味がない文もある
    Colorless green ideas sleep furiously.
  - 二つの意味を持つ、あいまいな文
    I saw a man with a telescope.

意味の「同一性」 / 意味の共通表現の存在

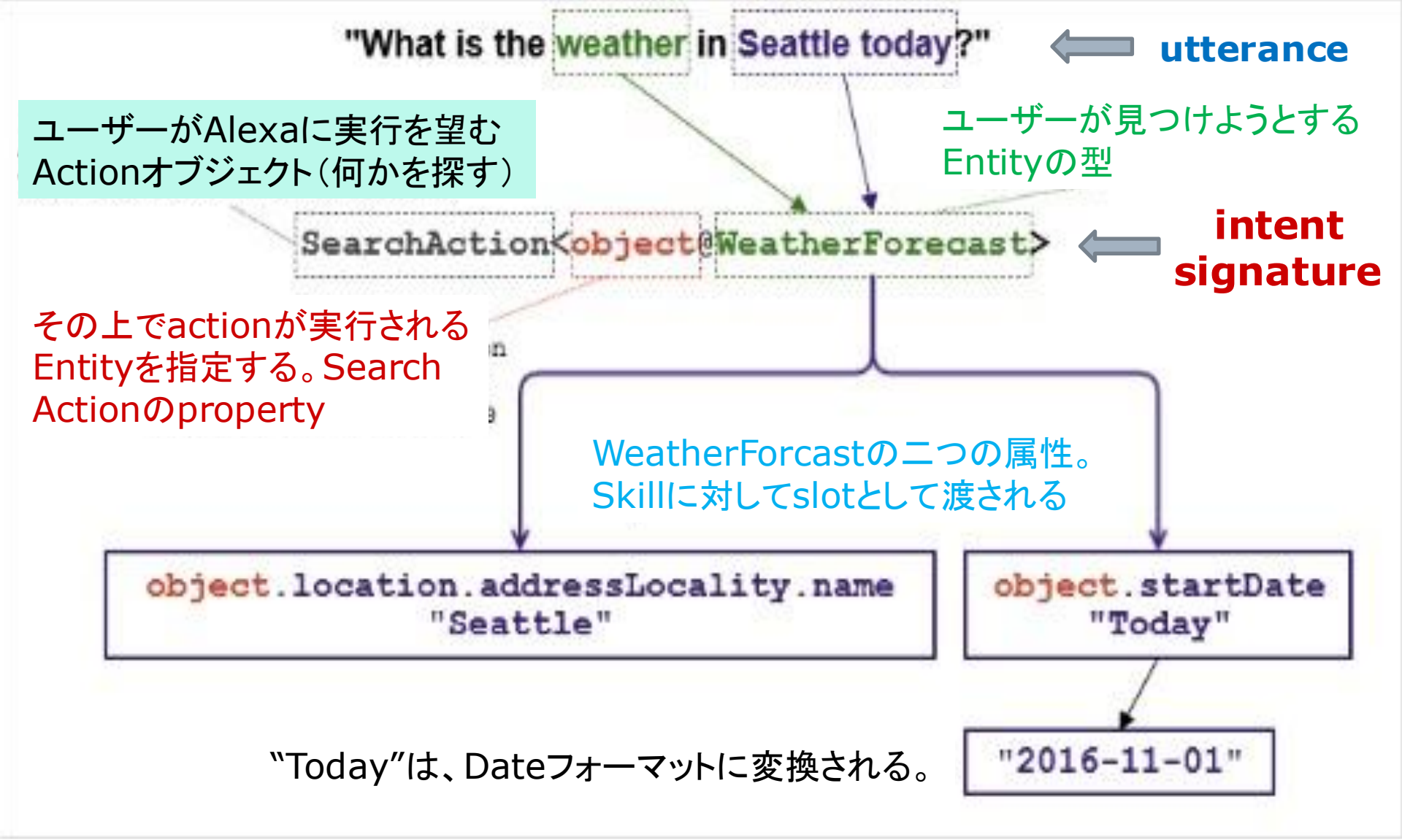
- 言語が異なっていても、「同じ」意味を持つ語が存在する。
- 言語が異なっていても、「同じ」意味を持つ文が存在する。

機械翻訳技術から派生した大規模言語モデルの成功は、この「意味の共通表現」が存在し、それを機械が発見する能力を持つことによるものだと、考えることができる。

現在の主な自然言語処理の実装は文法・文の構成性を考慮していない

	語の意味の「表現」	文の構成性 Grammar / Syntax	文の意味の「表現」 Semantics
Intentモデル	X	X	Intent Signature
Entityモデル	Entityの Ontology	X	Action Typeの Property
機械翻訳モデル	多次元ベクトルによる分散表現（明示的な利用）	X	多次元ベクトルによる分散表現（暗黙の利用）

AlexaのIntent Signature



```
<!--  John listened to Pink with Steve at Anna's apartment on his iPod. -->
<script type="application/ld+json">
{
 "@context": "http://schema.org",
 "@type": "ListenAction",
 "agent": {
  "@type": "Person",
  "name": "John"
 },
 "object": {
  "@type": "MusicGroup",
  "name": "Pink!"
 },
 "participant": {
  "@type": "Person",
  "name": "Steve"
 },
 "location": {
  "@type": "Residence",
  "name": "Ann's apartment"
 },
 "instrument": {
  "@type": "Product",
  "name": "iPod"
 }
}
</script>
```

Entityモデルでの
Action Type を用いた、
文の意味の表現

Action Property

- actionStatus
- agent
- endTime
- instrument
- error
- location
- object
- participant
- result
- startTime
- target

https://schema.org/Action

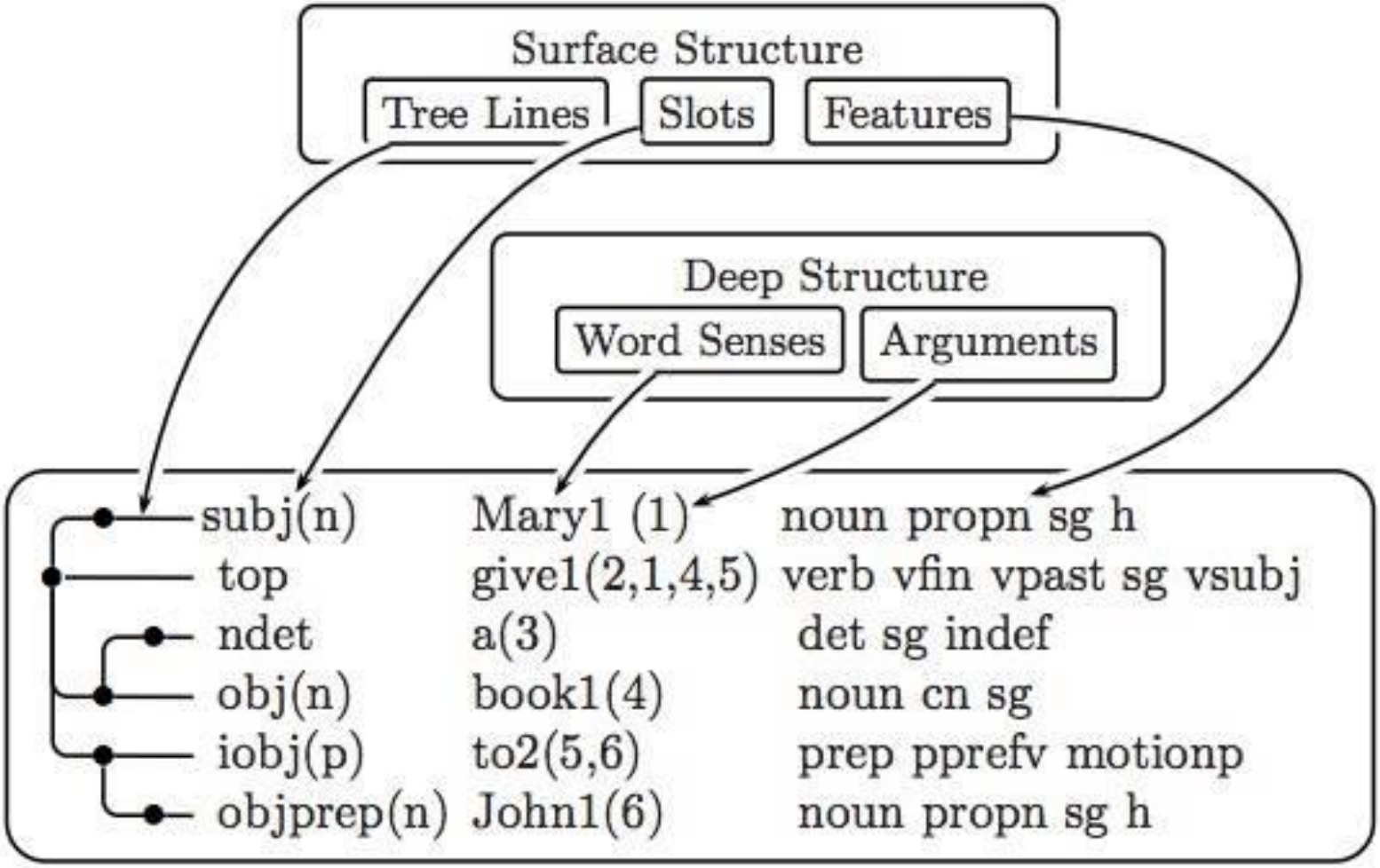
様々な言語・意味モデルの比較

文法と意味の表現を意識したモデルも存在する

	語の意味の「表現」	文の構成性 Grammar / Syntax	文の意味の「表現」 Semantics
Word2Vec	多次元ベクトルによる分散表現	—	—
IBM Watson内のプロジェクト	辞書項目	English Slot Grammar	論理式
Combinatory Categorial Grammar	辞書項目	Categorial Grammar	型付きラムダ式 Curry—Howard対応
DisCoCat	多次元ベクトルによる分散表現	Pregroup Grammar	多次元ベクトルによる分散表現 Functor Semantic

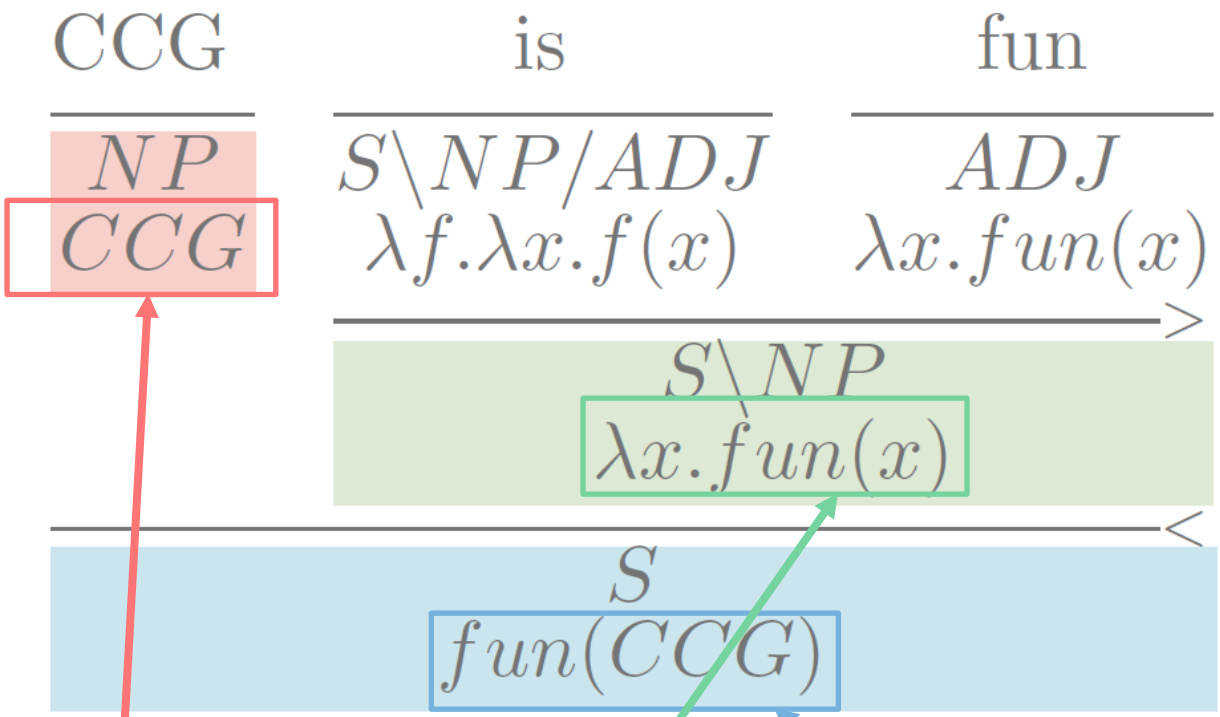
IBM Watson

"Mary gave a book to John."
Tree Line + Word Sense + Features



"CCG is fun"

Parsing with CCGs

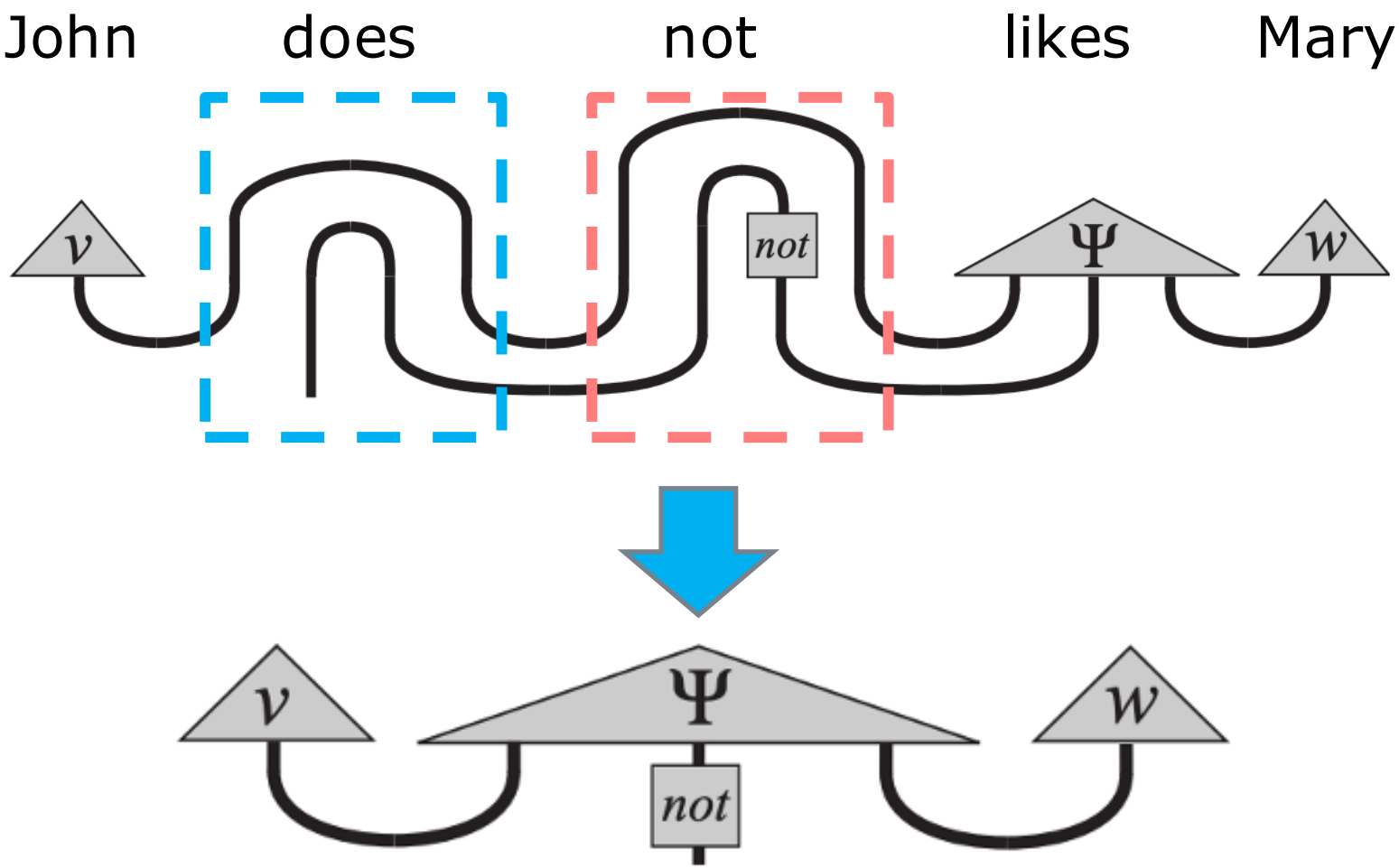


Combine categories using operators

$$B : g \quad A\backslash B : f \Rightarrow A : f(g) \quad (<)$$

DisCoCat

“John does not like Mary” の意味を表す図形とその変形



大規模言語モデルとCompositionality

大規模言語モデルの今後の発展を考えるうえで示唆的なのは、GoogleのKristina Toutanovaらが、compositionalityに注目を始めた事だと考えている。

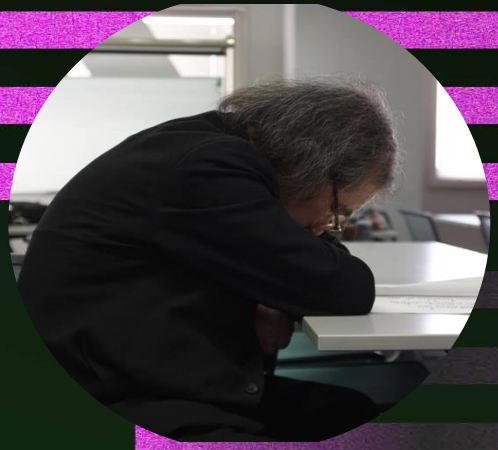
彼女らの最新の論文は、” Evaluating the Impact of Model Scale for Compositional Generalization in Semantic Parsing” では、”Despite their strong performance on many tasks, pre-trained language models have been shown to struggle on out-of-distribution compositional generalization.” と述べている。

参考リンク

- IBM Watsonについては、その問題意識とアプローチを僕は高く評価している。今回のセミナーのまとめページに、以前の資料を再掲した。
  https://www.marulabo.net/docs/meaning/

- CCGについては、「人工知能と意味の形式的理論」の第三部をみてほしい。
  https://www.marulabo.net/docs/semantics/

- DisCoCatについては、マルレク「ことばと意味の「構成性」について」がテーマとして取り上げている。
  https://www.marulabo.net/docs/discocat/



Part 3

意味の分散表現論の登場

- Bengioの「次元の呪い」
- HintonのEncoder / Decoder
- Word2Vec -- 語の意味のベクトル表現
- RNNの不思議な力

意味の分散表現論の登場
語の意味ベクトルの存在

意味の分散表現、まず、語の意味の分散表現として生まれます。それは、直接に文の意味を統計的に攻略することの困難の認識から始まります。Bengioはそれを「次元の呪い」と呼びました。

HintonのAutoencoderは、ニューラルネットに、データの次元を低減させるものでした。MikolovらのWord2Vec は、「語の意味ベクトル」を生成する効率的な方法を提案するとともに、その分散表現が、奇妙な特徴を持つことを発見します。

一方で、RNN / LSTM が不思議な能力を持つことが、発見されます。

Ilyaは、膨大な量のドキュメントを学習させて、文法的に正しい自然な文章をコンピュータに生成させることに初めて成功します。

ただ、その文章には意味がありませんでした。

Bengioの「次元の呪い」

A Neural Probabilistic Language Model

Yoshua Bengio et al.

http://goo.gl/977AQp

2003年

初期の言語への統計的アプローチの失敗

初期の、大量の言語データを統計的に処理すれば、言語の特質がわかるだろうという楽観的な見通しは、うまくいかなかった。

例えば、次の文の下線部に、三つの単語{to, two, too}から、一つ選んで文章を作るという問題を考えよう。

For breakfast I ate _____eggs.

しかし、機械に10億語もの用例集 “Very Very Large Corpora”を学ばせても、正解率が、100%に届かないのだ。

http://research.microsoft.com/pubs/66840/acl2001.pdf

すべての文例を網羅した用語集は存在し得ない

それは複雑さの尺度を見誤ったことによる。

26文字のアルファベット15文字以内で構成される語の数は、高々、$\mathbf{26^{15}}$である。ただし、語彙が10万個ある言語での10個の語からなる文は、$100000^{10} = \mathbf{10^{50}}$種類もある！

この$_1$ 文$_2$ は$_3$ 10$_4$ 個$_5$ の$_6$ 語$_7$ から$_8$ できて$_9$ いる$_{10}$
10語文というのは、そんなに長い文章ではない。が、10^{50}というのは、とてつなくも巨大な数である。

すべての語彙を集めた辞書は存在するかもしれないが、すべての文例を網羅した用語集は存在し得ない。

Bengioの「次元の呪い」

Bengioは、早くから、言語処理に現れる組み合わせの数の爆発を意識していた一人である。2003年の論文で、彼は、それを「次元の呪い」**Curse of Dimentionality** と呼んだ。

Yoshua Bengio et al.
A Neural Probabilistic Language Model
http://goo.gl/977AQp

この論文で、Bengioは重要な問題提起をする。

featureの分散表現で、「次元の呪い」と戦う

この論文で、彼は、次のような方法を提案する。

1. 語彙中のそれぞれの語に、R^mに実数値の値を持つ、分散した語の特徴ベクトル(word feature vector)を対応づける。
2. 語の並びの結合確率関数を、この並びの中の語の特徴ベクトルで表現する。
3. 語の特徴ベクトルとこの確率関数のパラメーターを、同時に学習する。

要は、語の「特徴ベクトル」という、語の「意味」の対応物を導入しようということだと僕は理解している。こうしたアプローチは、Tomas Mikolovらの **Word2Vec** に受け継がれていく。

少し乱暴にまとめてみる

- 「文」全体を相手にすると、その数はあまりに多すぎて、いくら統計的手法を使っても、言語の特徴をとらえるのは難しい。「次元の呪い」で勝ち目はない。それより遥かに数の少ない「語」の特徴づけから始めよう。「語」から「文」を攻めよう。
- それにしても「語」の特徴が一つの数字で表されるとは思えない。m個の数字の組 すなわちm次元のベクトルとして、「語」の特徴を表すことにしよう。
- 「文」は「語」の並びである。「文」が、「語」の並びとしてある確率的特徴を持つなら、それは「語」の特徴と結びつけることができるはずだ。
- コンピュータは、語の特徴ベクトルと「語」の並びの結合確率関数のパラメーターを、同時に学習すればいい。

Encoder / Decoder
HintonのAutoencoder

ここでは、AIでの意味の扱いの中心的な役割を果たす
Encoder / Decoder の枠組みを紹介する。

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton et al.

https://www.cs.cmu.edu/~tom/10701_sp11/slides/DeepNets_science2006.pdf

2006年

論文の概要

高次元の入力ベクターを再構成する小さな中央層を持つ多層のニューラル・ネットワークを訓練することで、高次元のデータを低次元のコードに変換することができる。

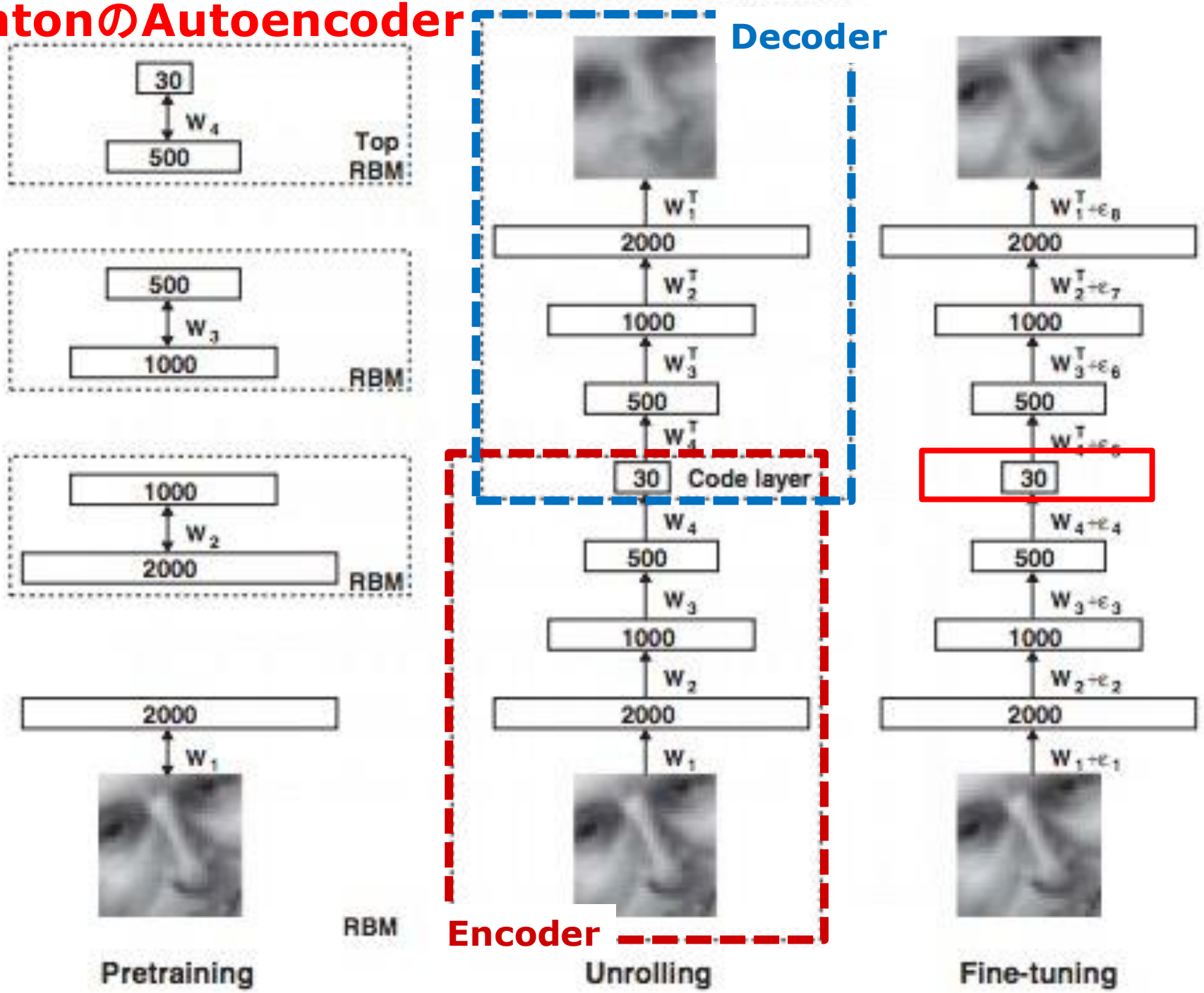
このようなAuto Encoderネットワークの重みを調整するのに勾配降下法を利用できる。ただし、それは、重みの初期値が、良好な解に近い場合にのみうまく働く。

我々は、ディープAuto Encoderネットワークが、低次元のコードを学習することを可能にする重みの初期化の効率的な方法について述べる。この方法は、データの次元を下げるツールとしての主成分分析より、ずっと優れている。

Autoencoder

- 次の図の中央、下の方の赤い枠がEncoderである。Encoderは、2000次元のベクトル（2000 pixelの画像データ）を、30次元のベクトルに変える。上の方の青い枠のDecoderは、この30次元のベクトルから、2000次元のベクトルを生成する。（こうして、画像が復元される）
- 論文では、このAutoencoderを微調整する方法が示されているのだが（下図の右側。画質が改善されている）、それについては割愛する。
- 注目して欲しいのは、ここでは、入力に与えられたデータ自身が、教師用のデータの役割を果たすので、その意味では、ラベルづけられた教師用のデータを必要としないということ。
- Autoencoderとは、「**自己エンコーダ**」の意味である。

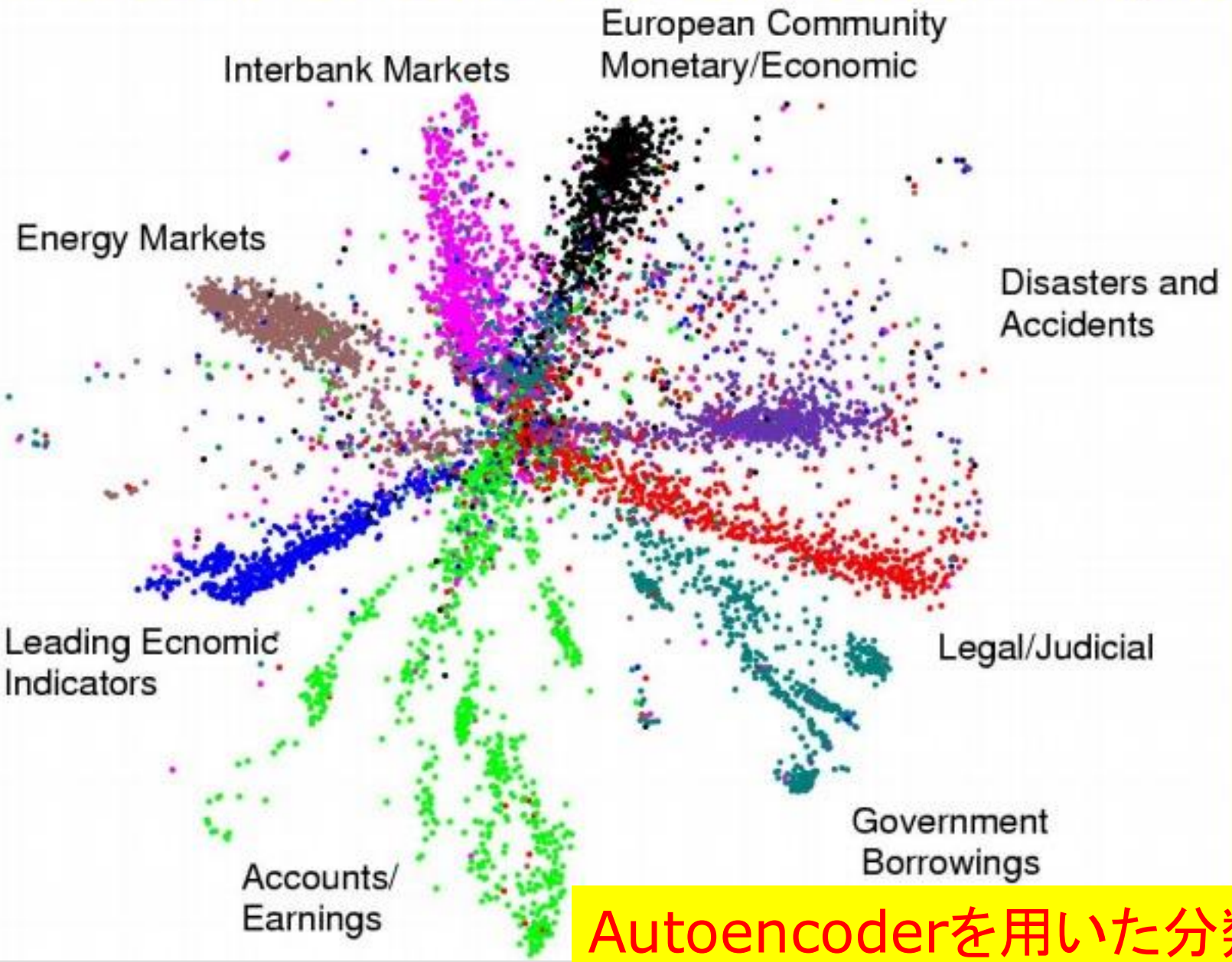
HintonのAutoencoder



書籍の分類へのAutoencoderの利用

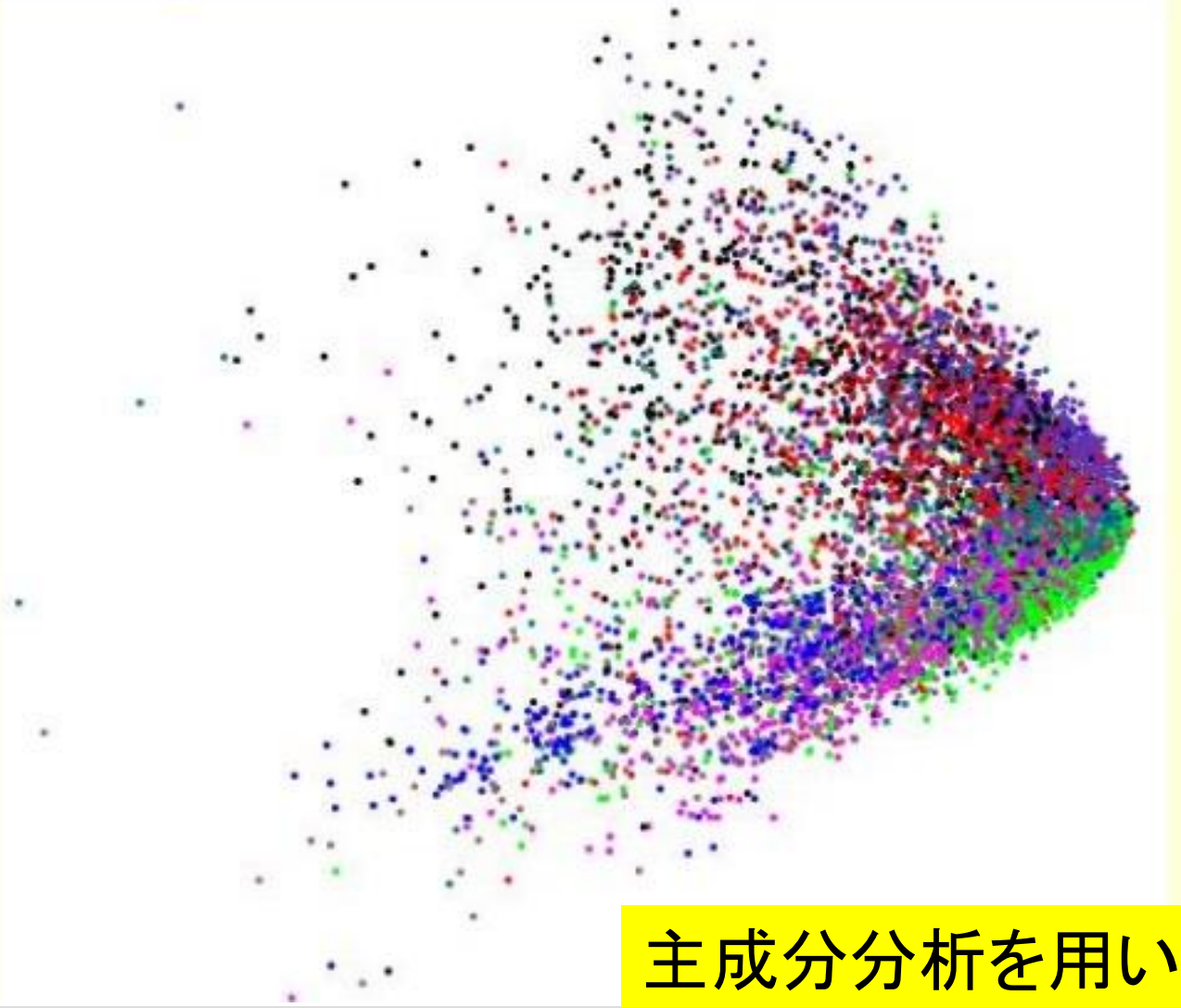
- 先の例は、「画像圧縮技術」の一種として理解してもいいのだが、この論文で、Hintonは、もっと面白い例を紹介している。書籍の分類に、このAutoencoderを使おうというものである。
- よく使われる単語を2000個ほど選ぶ。ある本にこれらの単語が何個含まれているかをカウントする。そうすると、ある本に2000次元の整数からなるベクトルを対応づけることができる。このベクトルをAutoencoderの入力に与えて、Autoencoderがこのベクトルを出力に再現できるように訓練をする。Autoencoderの中央のボトルネックの部分を、10次元のベクトルにすると、ある本に10個の数字を対応づけることができる。
- Hintonは、40万冊のビジネス書を対象に、この方法で得られた10個の数字が、書籍の分類に有効かどうかを実験した。結果を、二次元に可視化したものが、次の図だ。見事に、分類に成功している。

First compress all documents to 2 numbers.
Then use different colors for different document categories



Autoencoderを用いた分類

First compress all documents to 2 numbers using a type of PCA
Then use different colors for different document categories



主成分分析を用いた分類

Semantic hashing（意味的ハッシング）

- 重要なことは、「画像」と「書籍」では、対象のデータの性質はまるで異なるのだが、Autoencoderは、そのいずれに対しても、高次元のデータを低次元のデータに変換しているということである。別の言葉で言えば、**それは、対象の高次元のデータから、低次元のデータを、元の情報のエッセンスとして取り出しているのである。**
- Hintonは、こうしたAutoencoderの働きを、**Semantic hashing（意味的ハッシング）**と呼んでいる。
- SHA-1のようなハッシングでは、ハッシュ化されたデータから元のデータを復元することは不可能なのだが、Semantic hashingされたデータは、データの次元は低いものの、元の情報の中核部分を保持している。

Word2Vec --「語の意味ベクトル」

MikolovらのWord2Vec論文が出るのは、Bengioの論文から10年後の2013年だった。

Word2Vec論文

2013年に、Google（当時）のTomas Mikolovらは、語が埋め込まれたベクター空間が、言語学的に（文法的にも、意味論的にも）面白い性質を持っていることを発見する。

Tomas Mikolov,Wen-tau Yih,Geoffrey Zweiget

Linguistic Regularities in Continuous Space Word Representations

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rvecs.pdf

Word2Vec 二つ目の論文とコード公開

Jeff Deanが先の論文に興味を持ち、共同で研究を始め二つ目の論文を発表。

Tomas Mikolov,Kai Chen,Greg Corrado,Jeffrey Dean
Efficient Estimation of Word Representations in Vector Space
https://arxiv.org/pdf/1301.3781.pdf

Google Codeに、オープンソースとして公開され、大きな関心を集める。
https://code.google.com/p/word2vec/

論文の概要

連続空間言語モデルは近年、様々なタスクで優れた結果を示している。本論文では、入力層の重みによって暗黙的に学習されるベクトル空間での単語表現について検討する。

我々は、これらの表現が言語の構文的、意味的規則性を驚くほどよく捉えていること、また、各関係が関係固有のベクトルオフセットによって特徴付けられることを見出した。

これにより、単語間のオフセットに基づくベクトル指向の推論が可能となる。例えば、男性と女性の関係は自動的に学習され、誘導されたベクトル表現により、"King - Man + Woman" は "Queen" に非常に近いベクトルとなる。

単語ベクトルが構文規則性を捉えることを、構文類推問題（本論文で提供）により実証し、ほぼ4割の問題に正解することができる。

また、ベクトルオフセット法を用いてSemEval-2012 Task 2の問題に解答し、単語ベクトルが意味的な規則性を捉えていることを実証した。驚くべきことに、この方法は過去の最良のシステムを凌駕している。

第二論文の概要

我々は、非常に大規模なデータセットから単語の連続ベクトル表現を計算するための2つの新しいモデルアーキテクチャを提案する。

これらの表現の品質は単語の類似性タスクで測定され、その結果は異なるタイプのニューラルネットワークに基づく、これまでで最も性能の良い技術と比較される。

その結果、16億語のデータセットから高品質の単語ベクトルを学習するのに1日もかからず、より低い計算コストで精度が大幅に向上することが確認された。

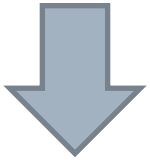
さらに、これらのベクトルは、我々のテストセットにおいて、構文的および意味的な単語の類似性を測定するための最先端の性能を提供することを示す。

「語の意味ベクトル」の性質

どんな語が、与えられた語の近くに埋め込まれるか？

- 似た意味を持つ言葉は、似たベクトルを持つ。
- 似た言葉で置き換えても、正しい文は、正しい文に変わる。

“a **few** people sing well”　　正しい文



“a ***couple*** people sing well”　　正しい文

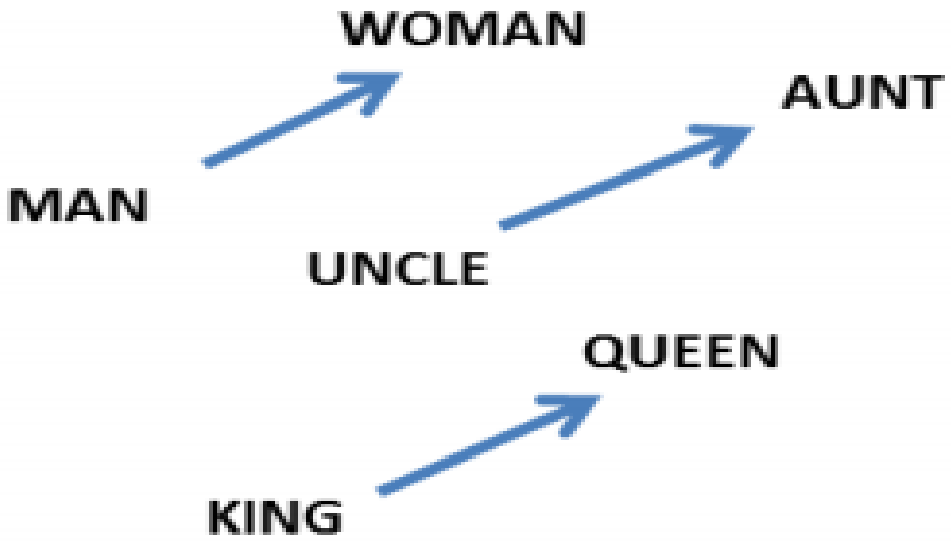
- 意味が似ていなくても、同じクラスの言葉で置き換えても、正しい文は、正しい文に変わる。

“the **wall** is **blue**” ⟶ “the ***ceiling*** is ***red***”

意味を変換するベクトルは共通？

Word Embeddingは、もっと面白い性質を持つ。下の図のように、男性から女性へのベクトルがあるように見える。

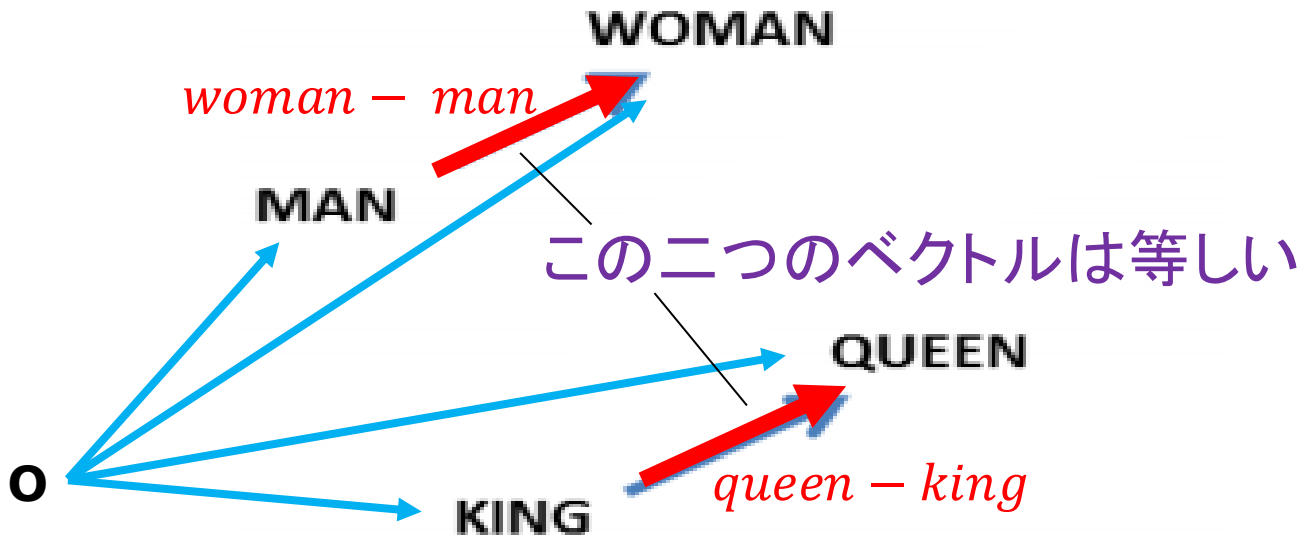
W(“woman”)−W(“man”) ≃ W(“aunt”)−W(“uncle”)
W("woman")−W("man") ≃ W("queen")−W("king")



意味を変換するベクトルは共通？
Vector Offset Method

Word Embeddingは、もっと面白い性質を持つ。下の図のように、男性から女性へのベクトルがあるように見える。

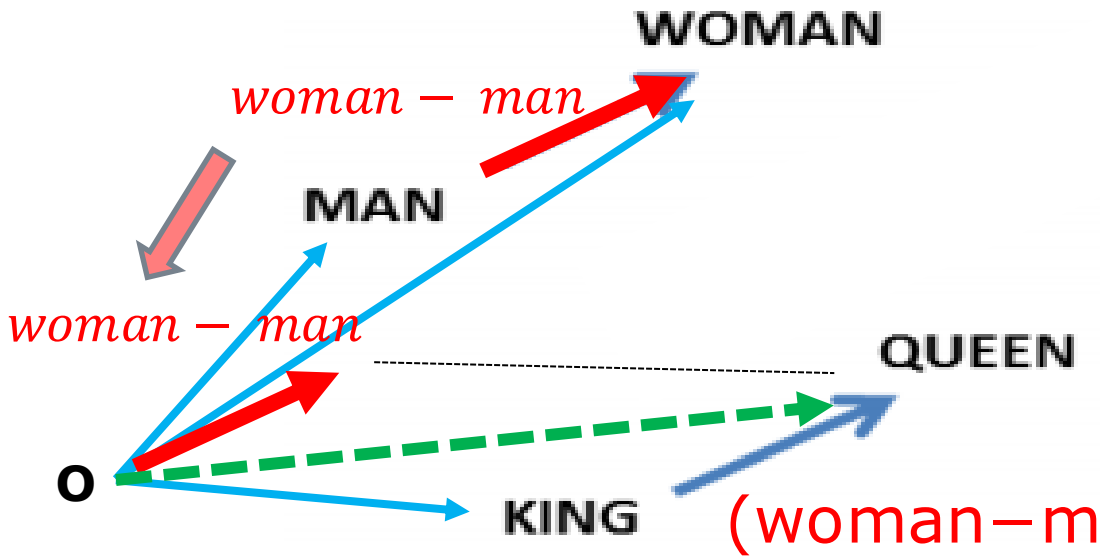
W(“woman")−W(“man") ≃ W(“queen")−W(“king")



King - Man + Woman = Queen
Vector Offset Method

次のような変形もできる。

W(“woman")–W(“man") + W(“king") ≃ W(“queen")



(woman–man) + king ≃ queen

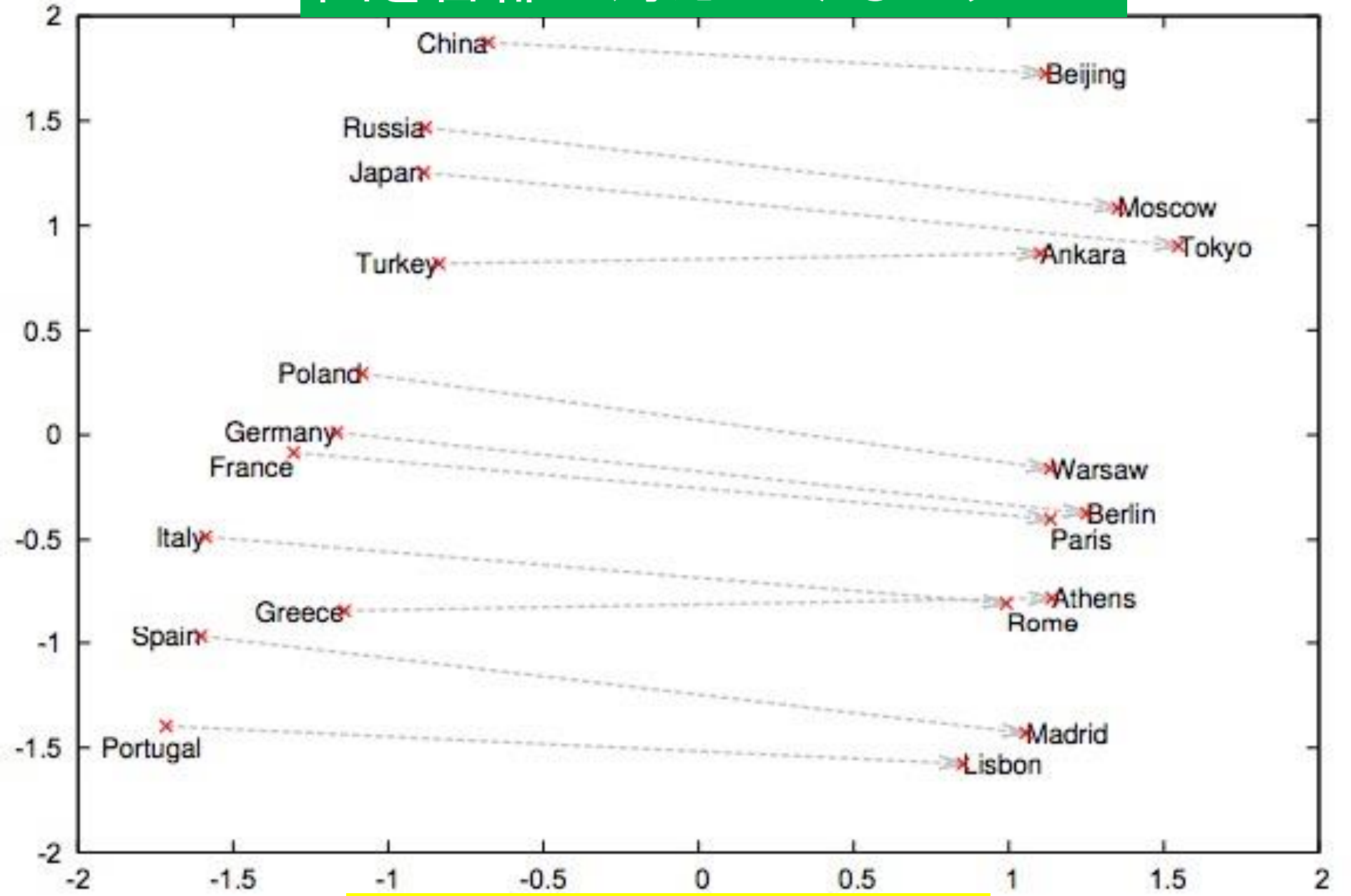
Word2Vecが対応を見つけたベクトルの例

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

文法的な対応

Category	Relation	Patterns Tested	# Questions	Example
Adjectives	Base/Comparative	JJ/JJR, JJR/JJ	1000	good:better rough:___
Adjectives	Base/Superlative	JJ/JJS, JJS/JJ	1000	good:best rough:___
Adjectives	Comparative/ Superlative	JJS/JJR, JJR/JJS	1000	better:best rougher:___
Nouns	Singular/Plural	NN/NNS, NNS/NN	1000	year:years law:___
Nouns	Non-possessive/ Possessive	NN/NN_POS, NN_POS/NN	1000	city:city's bank:___
Verbs	Base/Past	VB/VBD, VBD/VB	1000	see:saw return:___
Verbs	Base/3rd Person Singular Present	VB/VBZ, VBZ/VB	1000	see:sees return:___
Verbs	Past/3rd Person Singular Present	VBD/VBZ, VBZ/VBD	1000	saw:sees returned:___

国を首都に対応づけるベクトル



http://arxiv.org/pdf/1310.4546.pdf

「語の意味へのベクトル:で重要なこと
語の意味の近さを定義できる

「語の意味のベクトル表現」でもっとも重要なことは、このアプローチによって、語の意味の近さを定義できることである。

語vと語wのベクトル表現を$\vec{v}$と$\vec{w}$する。この時、

語vと語wの意味の近さ$Similarity(v,w)$を、ベクトル$\vec{v}$と$\vec{w}$の内積$\vec{v}\cdot\vec{w}$で定義する。

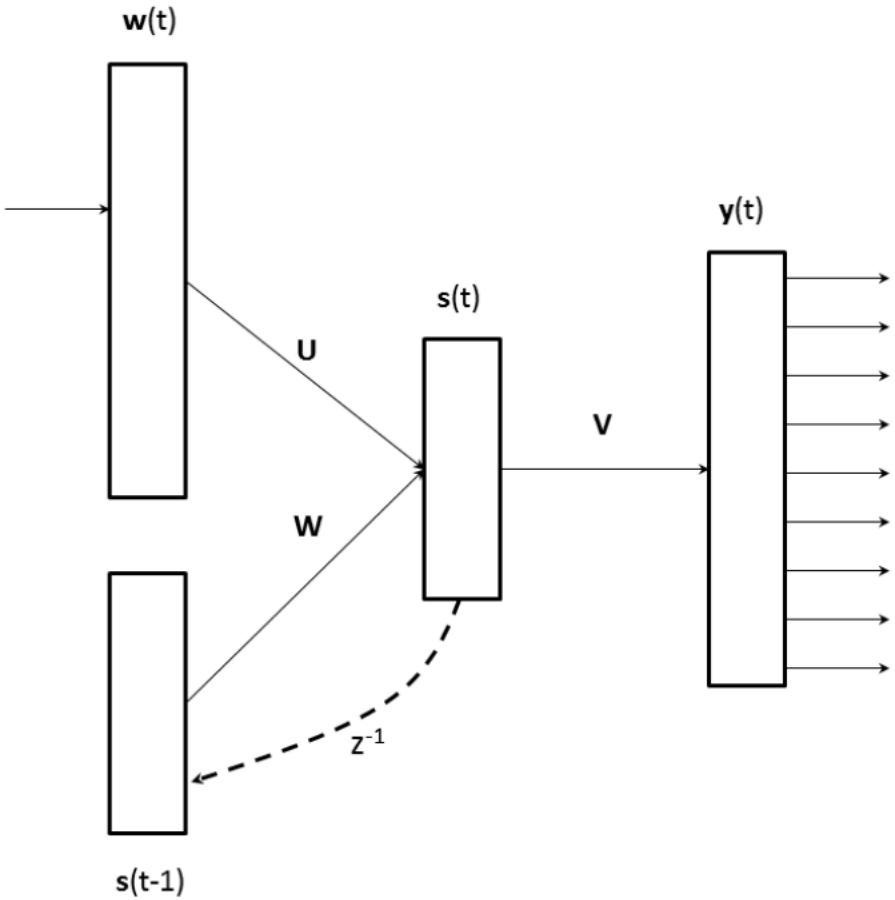
$$Similarity(v,w)=\vec{v}\cdot\vec{w}=|\overrightarrow{v}||\overrightarrow{w}|cos\theta$$

ここに、θは、ベクトル$\vec{v}$と$\vec{w}$のなす角である。これを、cosine-similarityと呼ぶ。

語はどのようにベクトルに変換されるのか

WordをVectorに変える方法

第一論文では、次のようなRNNが使われている。



$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right),$$

$$f(z) = \frac{1}{1+e^{-z}}$$

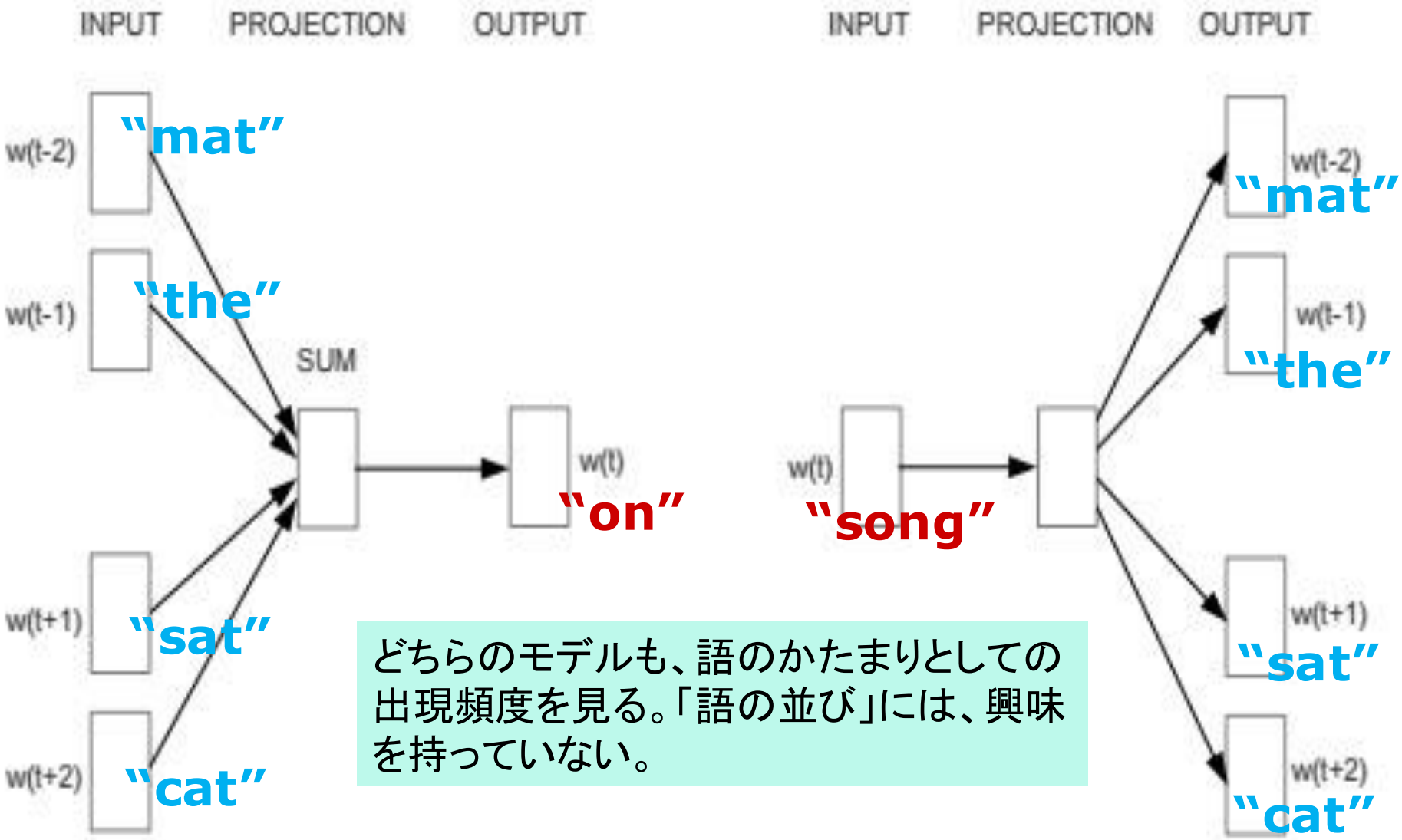
$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

WordをVectorに変える方法

第二論文で使われている方法は、次の二つである。

- **CBOW(Continuus Bag-of-Word”) モデル**
  複数の語の集まりから、一緒に出現しそうな一つの語の確率を調べる。

- **Skip-gram モデル**
  一つの語が与えられた時、一緒に出現しそうな複数の語の確率を調べる。



Continuous Bag-of-Words

CBOW

Skip-gram

http://arxiv.org/pdf/1301.3781.pdf

語の意味の分散モデル を作る
Tai-Danaeの説明

例えば好きな本でもいい、ある固定したコーパスがあったとしよう。
そこから、 context words $\{w_1, \ldots, w_n\}$ と呼ばれる語の集まりを選ぶ。それは、コーパス内のすべての言葉でもいいし、その一部でもいい。この$\{w_1, \ldots, w_n\}$ のw_iをベクトル空間Vの i 番目の標準的な基底と考えるのだ。

そうすれば、コーパス内のすべての語は、次のようなcontext wordの線形結合で表されるベクトル表現を持つことになる。

$$\boldsymbol{w} = \sum_{i=1}^{n} c_i \boldsymbol{w_i}$$

この係数c_iは、コーパス内で、語wが語w_iの近くに現れた回数を示す実数である。

語の意味の分散モデルの例

{sweet, green, furry} という語を含むある本があったとしよう。この三つの語をこのコーパスのcontext word に選んだら、これを基底として、次のようにベクトルで表す。

$$\mathbf{sweet} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{green} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{furry} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

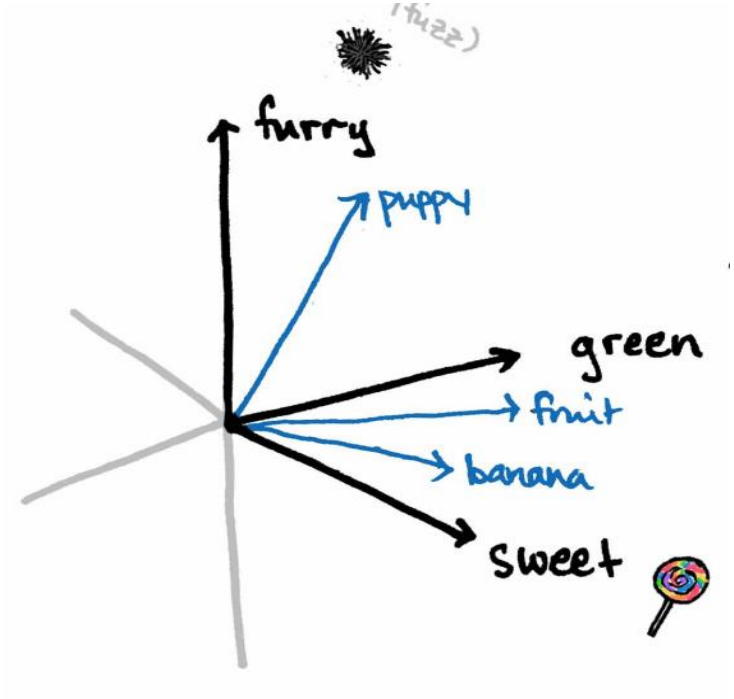
この時、この本の中の語 banana, puppy, fruit は、次のように表される。

$$\mathbf{banana} = \begin{bmatrix} 21 \\ 9 \\ 0 \end{bmatrix} \qquad \mathbf{puppy} = \begin{bmatrix} 8 \\ 1 \\ 32 \end{bmatrix} \qquad \mathbf{fruit} = \begin{bmatrix} 43 \\ 19 \\ 0 \end{bmatrix}$$

語の意味の分散モデルの例

別の言葉で言えば、我々は、コーパスからのデータをこれらの語を三次元のベクトル空間に埋め込むために利用したのである。

語 banana の意味は (21, 9, 0)、語 puppy の意味は (8, 1, 32)、語 fruitの意味は (43, 19, 0) ということになる。



RNNの不思議な力

2011年のChatGPT

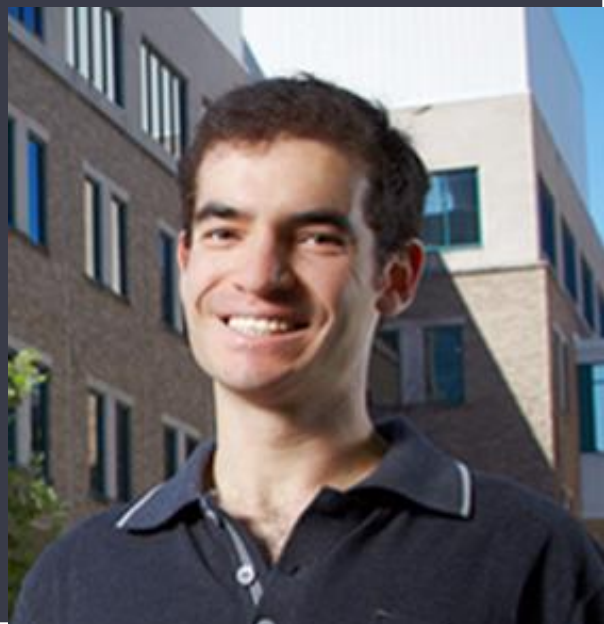
RNNによる文章の生成

Generating Text with Recurrent Neural Networks

Ilya Sutskever et al.

http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf

2011年



RNNによる文章の生成

Googleの Ilya Sutskeverは、文字数が5億文字にものぼるテキストを長い時間をかけてRecurrent Neural Nets に学習させ、次のページのような文章を生成することができた。

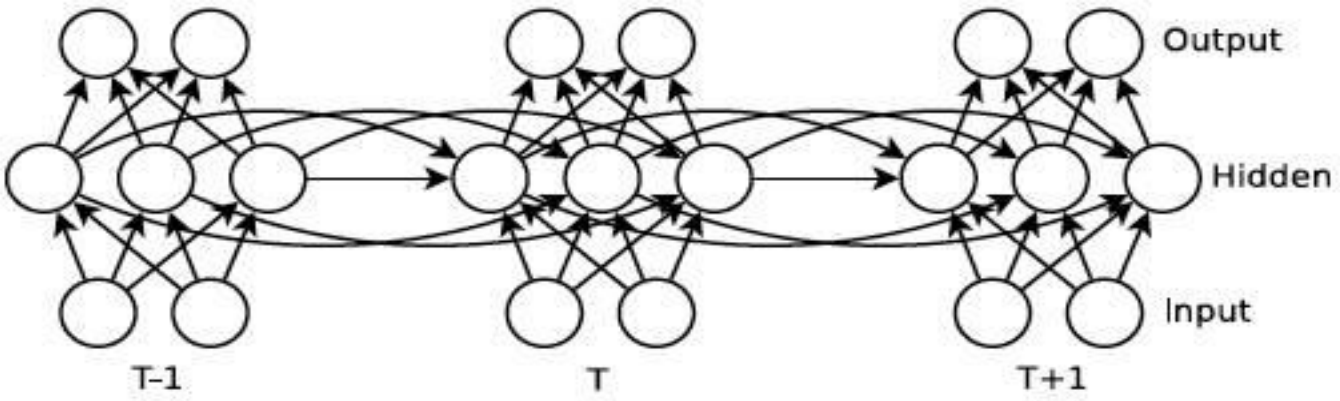


Figure 1. A Recurrent Neural Network is a very deep feedforward neural network whose weights are shared across time. The non-linear activation function used by the hidden units is the source of the RNN's rich dynamics.

WikiPediaで学習して生成された文章の例

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. In the show’s agreement unanimously resurfaced. The wild pasteured with consistent street forests were incorporated by the 15th century BE. In 1996 the primary rapford undergoes an effort that the reserve conditioning, written into Jewish cities, sleepers to incorporate the .St Eurasia that activates the population.

http://goo.gl/vHRHSn

New York Timesで学習して生成された文章の例

while he was giving attention to the second advantage of school building a 2-for-2 stool killed by the Cultures saddled with a halfsuit defending the Bharatiya Fernall ’s office . Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where .....

http://goo.gl/vHRHSn

2015年のChatGPT

RNNによる数学論文とソースコードの生成

The Unreasonable Effectiveness of Recurrent Neural Networks”

Andrej Karpathy

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

2015年

RNNによる数学論文とソースコードの生成

Karpathyは、Stacks Projectの膨大な数学論文 https://zbmath.org/software/31299 をRNNに学習させ、数学論文（モドキ）を生成してみせた。

彼はまた、Linuxのソースコードを RNNに学習させて、Cプログラムのソースコード（モドキ）を生成してみせた。

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciy in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets.

RNNが産み出した
数学論文モドキ

Stack Theory の教科書を
「学習」させたもの

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic
quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

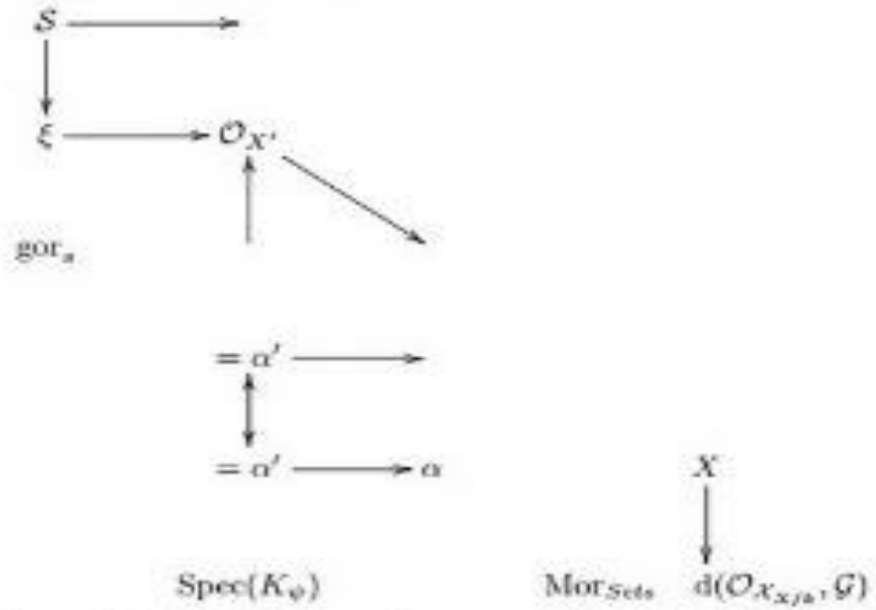
(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ v
finite type.

RNNが産み出した
数学論文モドキ

Stack Theory の教科書を
「学習」させたもの

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U.

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The fu
"field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_i}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\nu}^{\overline{y}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type o
If $\mathcal{F}$ is a scheme theoretic image points.

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??
sequence of $\mathcal{F}$ is a similar morphism.

RNNが産み出した
数学論文モドキ

Stack Theory の教科書を
「学習」させたもの

RNNは、Latexの構文規則を学習し、
こうしたシーケンスを構成できる。

```
\begin{proof}
We may assume that $\mathcal{I}$ is an abelian sheaf on $\mathcal{C}$.
\item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$
is an injective and let $\mathfrak q$ be an abelian sheaf on $X$.
Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\label{lemma-characterize-quasi-finite}
Let $\mathcal{F}$ be an abelian quasi-coherent sheaf on $\mathcal{C}$.
Let $\mathcal{F}$ be a coherent $\mathcal{O}_X$-module. Then
$\mathcal{F}$ is an abelian catenary over $\mathcal{C}$.
\item The following are equivalent
\begin{enumerate}
\item $\mathcal{F}$ is an $\mathcal{O}_X$-module.
\end{lemma}
```

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
 int error;
 if (fd == MARN_EPT) {
  /*
   * The kernel blank will coeld it to userspace.
   */
  if (ss->segment < mem_total)
   unblock_graph_and_set_blocked();
  else
   ret = 1;
  goto bail;
 }
 segaddr = in_SB(in.addr);
 selector = seg / 16;
 setup_works = true;
 for (i = 0; i < blocks; i++) {
  seq = buf[i++];
  bpf = bd->bd.next + i * search;
  if (fd) {
   current = blocked;
  }
```

RNNが産み出した
Cプログラム・モドキ

Linuxのソースコードを
「学習」させたもの

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
  return 0;
}
```

RNNが産み出した
Cプログラム・モドキ

Linuxのソースコードを
「学習」させたもの

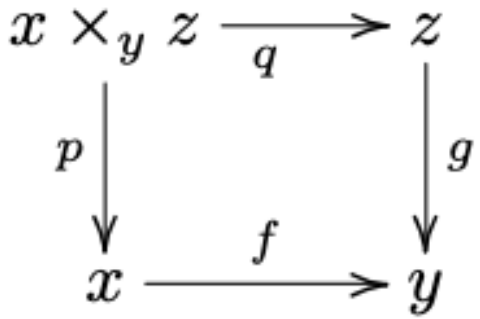
これらの生成されたテキストは「意味」を欠いている

生成された記事も数学論文もソースコードもは、意味をなさない。

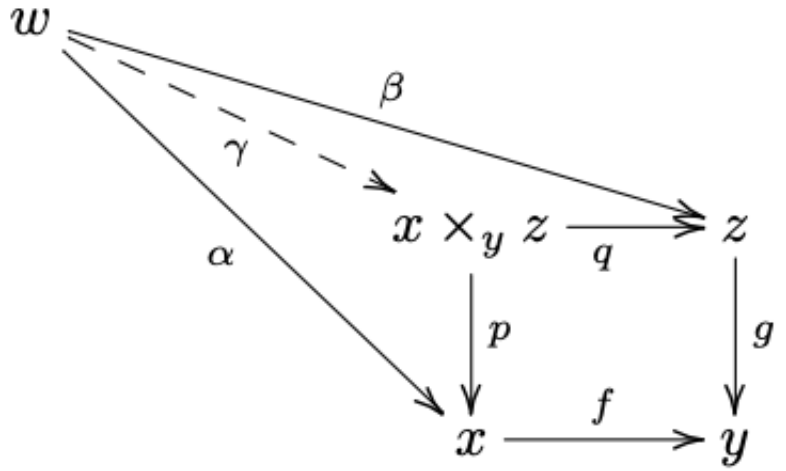
これらの生成されたテキストは「意味」を欠いている。

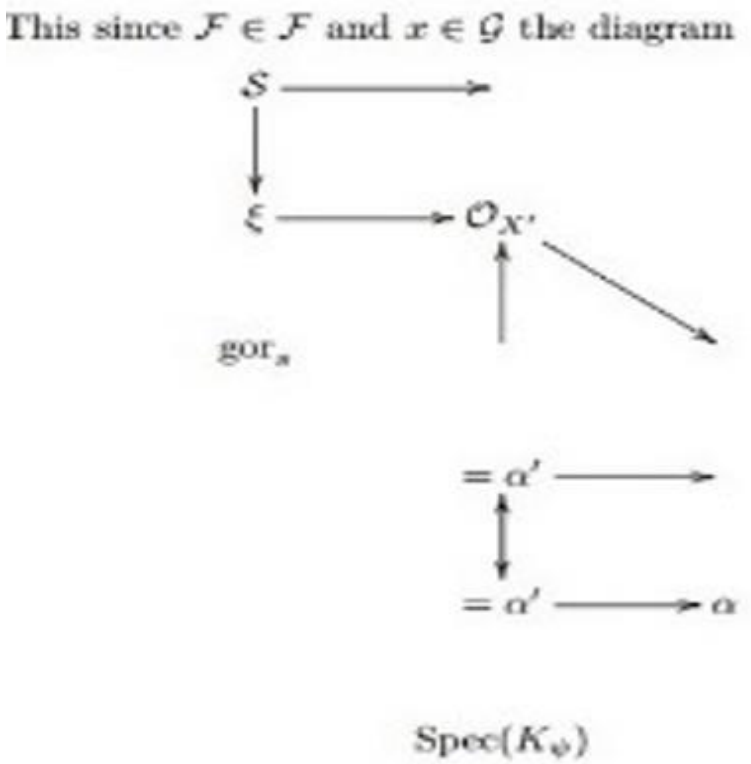
ただし、「文法的」「構文的」には、正しいように見える。

RNNは、文法を理解できている。

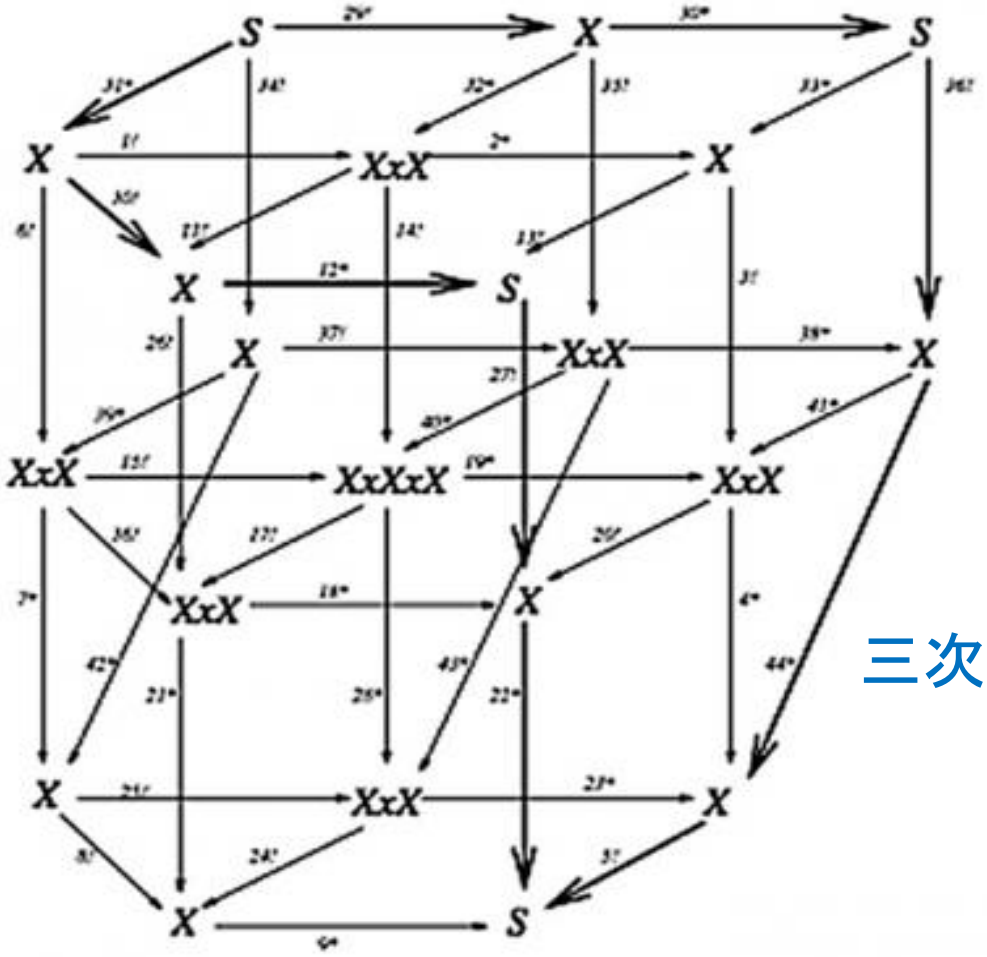


Diagram





RNNはDiagramの
生成に失敗している

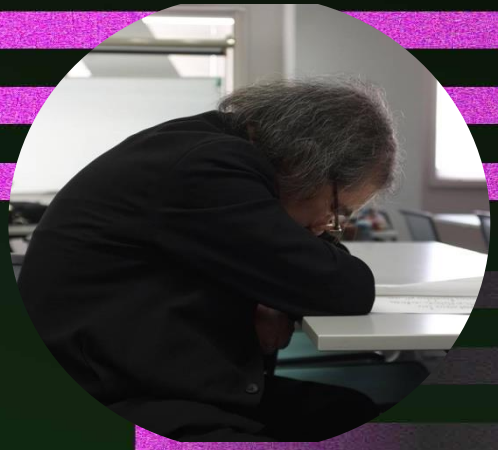


三次元のDiagram

For the convenience of further reference we numbered all the arrows. The right vertical face of the diagram is the diagram (2) defining the 2-morphism $Id \to \Omega\Sigma$ and the upper horizontal face is the diagram (1) defining the 2-morphism $\Sigma\Omega \to Id$. The whole diagram is the union of the front part which

Voevodsky “The Origins and Motivations of Univalent Foundations” より
https://www.ias.edu/ideas/2014/voevodsky-origins

画像を生成する能力と
数学的ダイアグラムを生成する能力は異なる



Part 4

意味の分散表現論の発展

- Sequence to Sequence
- Attention Mechanism
- Google ニューラル機械翻訳

文の意味ベクトルの発見

2014年に、Ilya Sutskever らは、シーケンスをシーケンスに変換するRNN(LSTM)の能力が、機械翻訳に応用できるという論文を発表します。

「我々の方法では、入力のシーケンスを固定次元のベクトルにマップするのに、多層のLong Short-Term Memory（LSTM）を利用する。その後、別の深いLSTMが、このベクトルから目的のシーケンスをデコードする。」

それでは、二つのSequence を結びつけているのはなんでしょう。それは二つのSequenceが「同じ意味」を持つということです。前段の入力のSequenceから作られ、後段の出力のSequenceを構成するのに利用される「固定次元のベクトル」とは、二つの文が「同じ意味」を持つことを表現している文の意味のベクトル表現に他なりません。

Sequence to Sequence
LSTMの機械翻訳への応用

2014年に、Ilya Sutskever らは、シーケンスをシーケンスに変換するRNN(LSTM)の能力が、機械翻訳に応用できるという論文を発表する。

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever et al.

https://arxiv.org/pdf/1409.3215.pdf

2014年

論文の概要

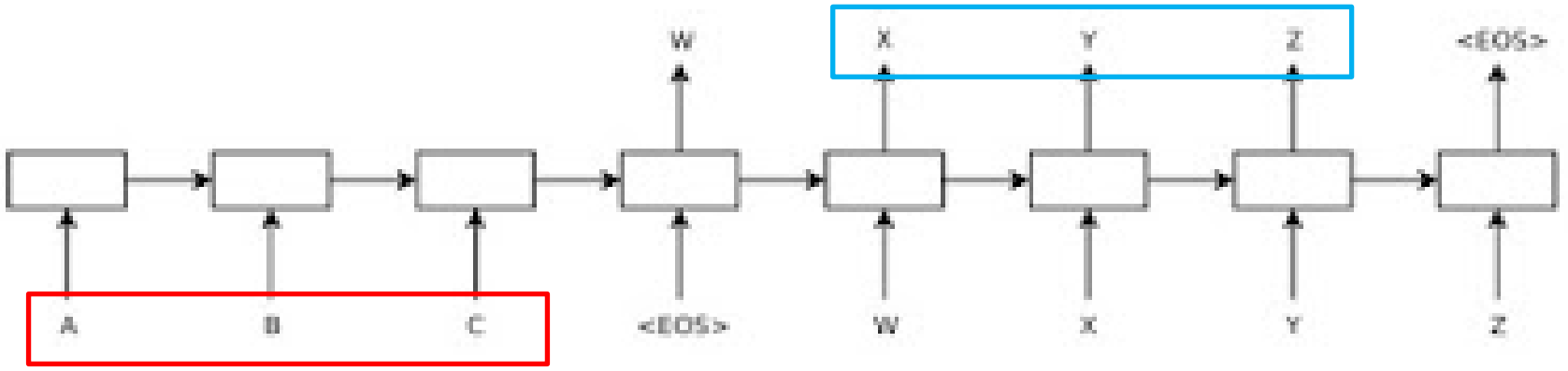
ディープニューラルネットワーク（DNN）は、難しい学習課題でも優れたパフォーマンスを達成する強力なモデルである。DNNは、ラベル付けられた訓練用のデータが利用可能な時には、いつもうまく機能するというものの、シーケンスをシーケンスにマップすることに、DNNを利用することはできない。

我々は、この論文で、シーケンスを学習するエンド・ツゥ・エンドの汎用のアプローチを提示する。そこでは、シーケンスの構造に最小限の前提しか課していない。

我々の方法では、入力のシーケンスを固定次元のベクトルにマップするのに、多層のLong Short-Term Memory（LSTM）を利用する。その後、別の深いLSTMが、このベクトルから目的のシーケンスをデコードする。

Sequence to Sequence

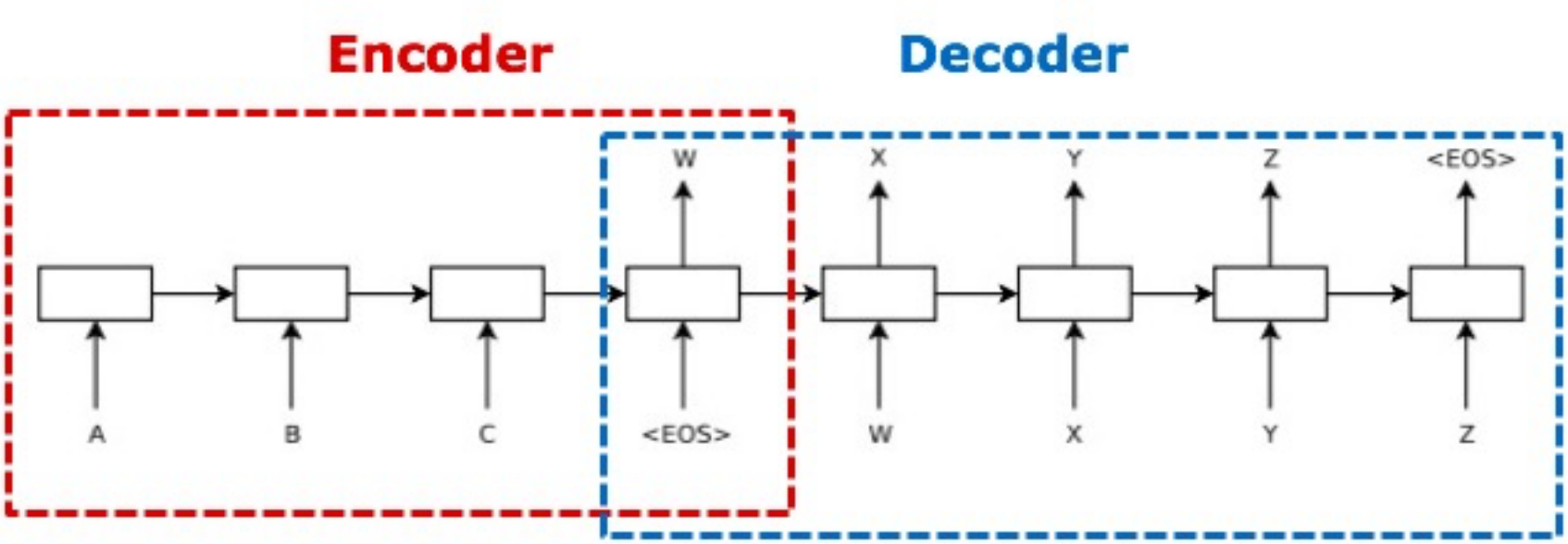
この論文の次の図を見て欲しい。



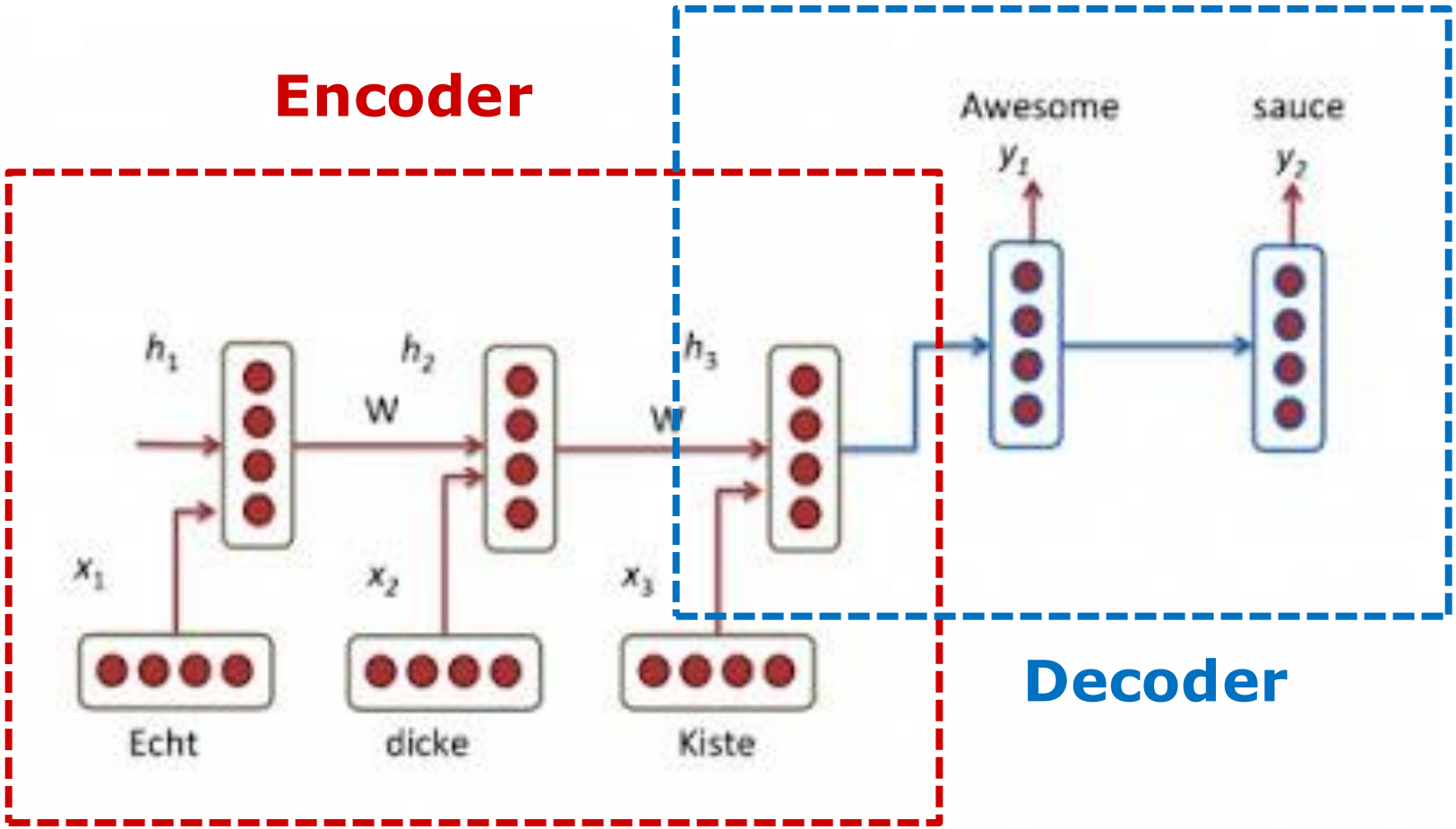
この図は、このシステムが、ABCというシーケンスが与えられた時、xyzというシーケンスを返すことを表している。<EOS>は、End of Sequence でシーケンスの終わりを表す特別な記号である。（これが、シーケンスの構造に課せられた「最小限の前提」である。）

これが、先に見たEncoder-Decorderのパターンであることは、次のようにしてわかる。

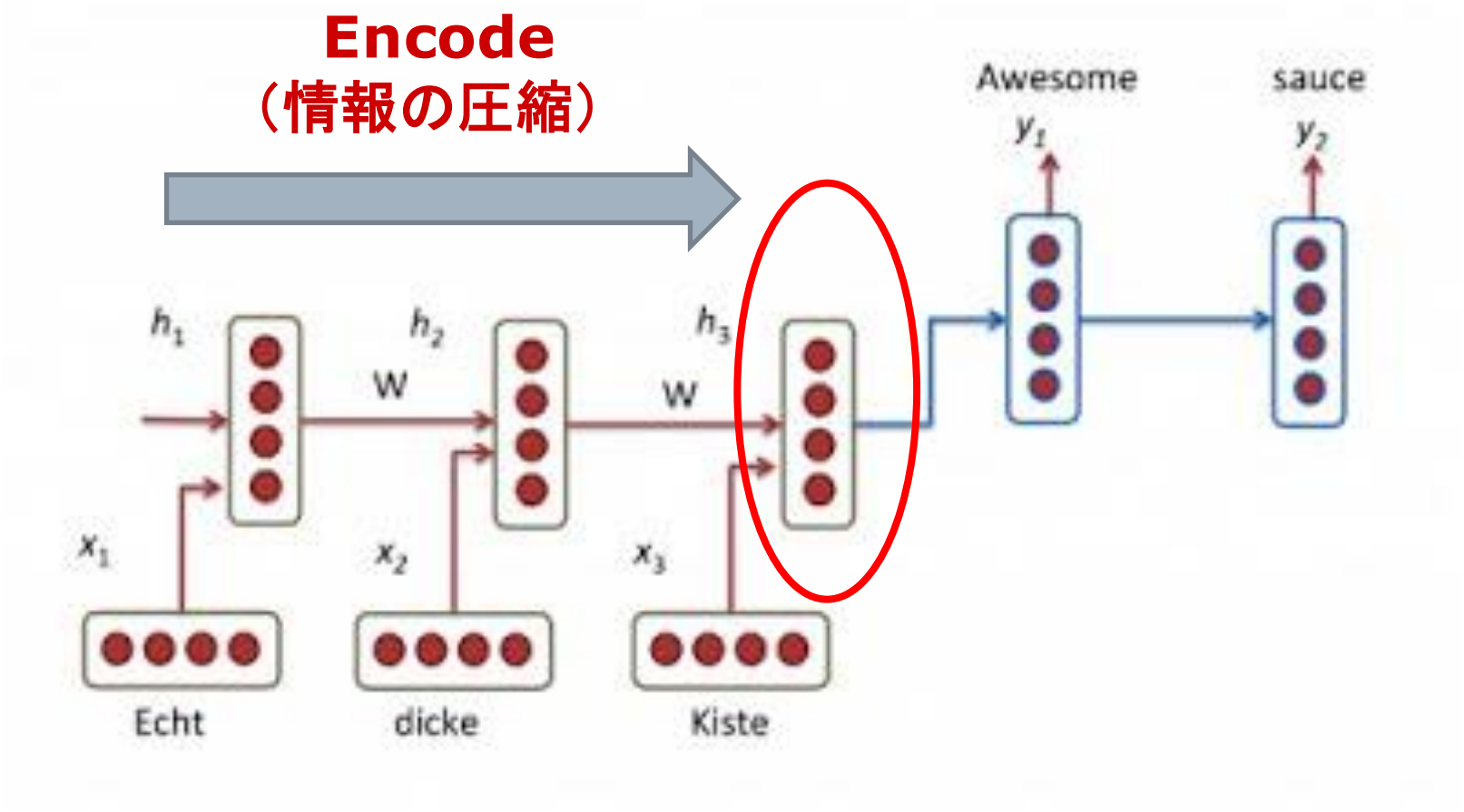
先行するLSTM群は、入力シーケンスABCを受け取って、それを固定長のベクトwに変換している。後行のLSTM群は、そのベクトルwを受け取って、それから出力シーケンスxyzを生成する。すなわち、先行のLSTM群をEncoder、後行のLSTM群をDecoderと考えることができる。中間に生成され、両者で共有されるwは、先のHintonのボトルネック部だと思えばいい。



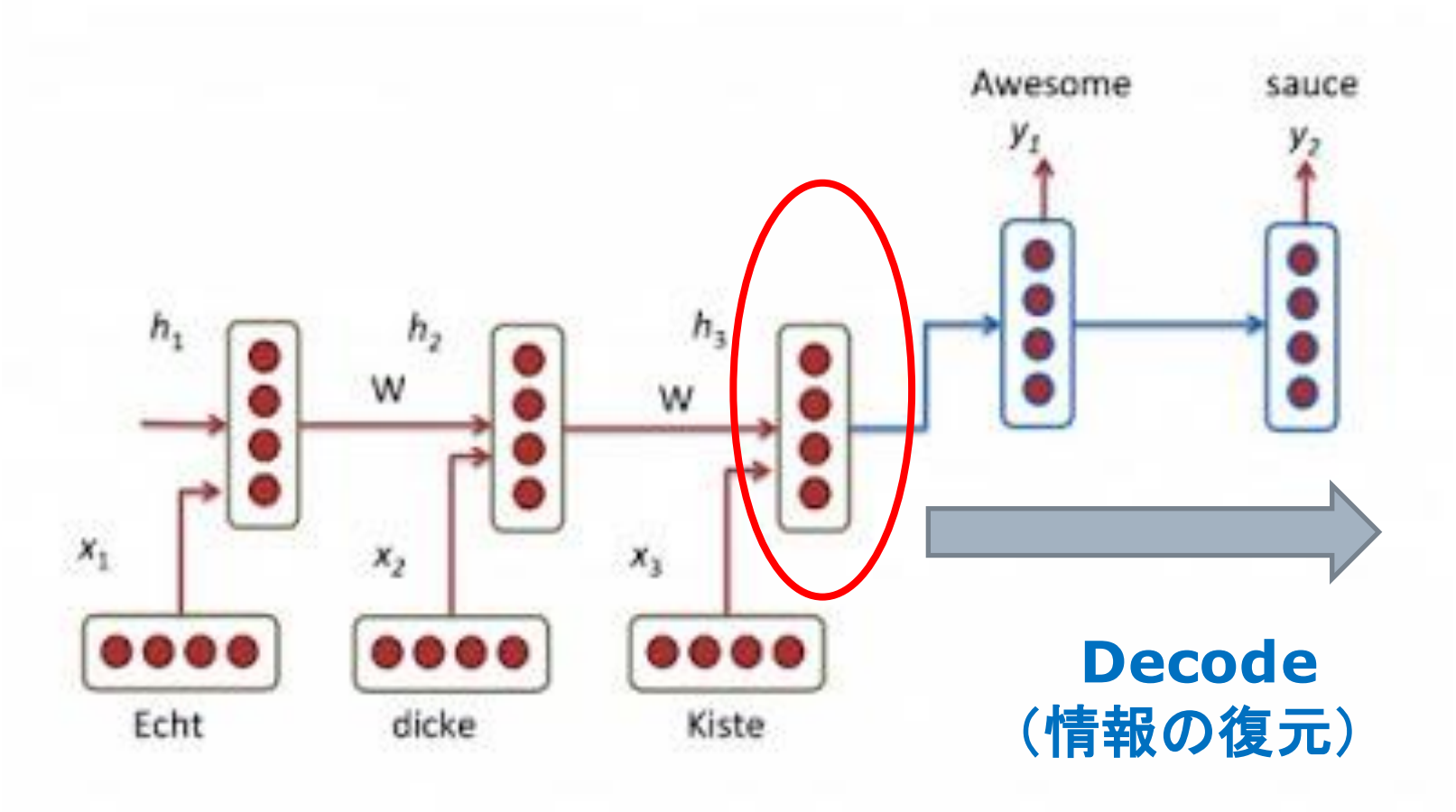
次の図(https://goo.gl/JGckBP から)は、こうしたメカニズムで、RNNが、独文の "Echt dicke Kiste" を英文の"Awesome sauce" に翻訳する様子を表している。(ここでは、文章の終わりを表す<EOS>は、省略されている)



ここでは、Encoder部が、文章の最後にとるRNNの内部状態 h_3 が、そのままDecoder部に渡されることが示されている。入力シーケンスの情報のエッセンスが、この内部状態 h_3に凝縮されていると考えればいい。



AutoencoderのDecoder部が、圧縮された情報から元の情報を復元しようとするように、ここでは、その情報から、「**同じ意味**」を持つ、別の言語の文章を復元しようとする。



Ilya Sutskever らは、このアーキテクチャーで、英語をフランス語に翻訳するシステムを作成し、BLEUのスコアで、34.81という高得点をたたき出した。

この時のシステムは、5段重ねのLSTMで構成され、それぞれが8,000次元の状態からなる384M個のパラメーターを持つものだった。

Attention Mechanism

Ilyaたちのアプローチでは、どんな長さの文でも、その意味は固定長のベクトルで表現されることになる。

Bengioのグループは、それでいいのかと批判し、新しいメカニズムを提案する。

Neural machine translation by jointly learning to align and translate

Bahdanau, D., Cho, K., and Bengio, Y

https://arxiv.org/pdf/1409.0473.pdf

2016年

論文の概要

近年、ニューラル機械翻訳として提案されたモデルは、多くの場合、Encoder-Decoderのファミリーに属している。そこでは、ソースの文が固定長ベクトルにエンコードされ、そこからデコーダが 翻訳文を生成する。

この論文では、固定長ベクトルの使用が、この基本的なEncoder/Decoderアーキテクチャの性能を改善する上でのボトルネックになっていると推論し、モデルに自動的に、ターゲット・ワードを予測するのに重要なソース・文の一部分について、（ソフト）検索を可能とすることによって、これを拡張することを提案する。

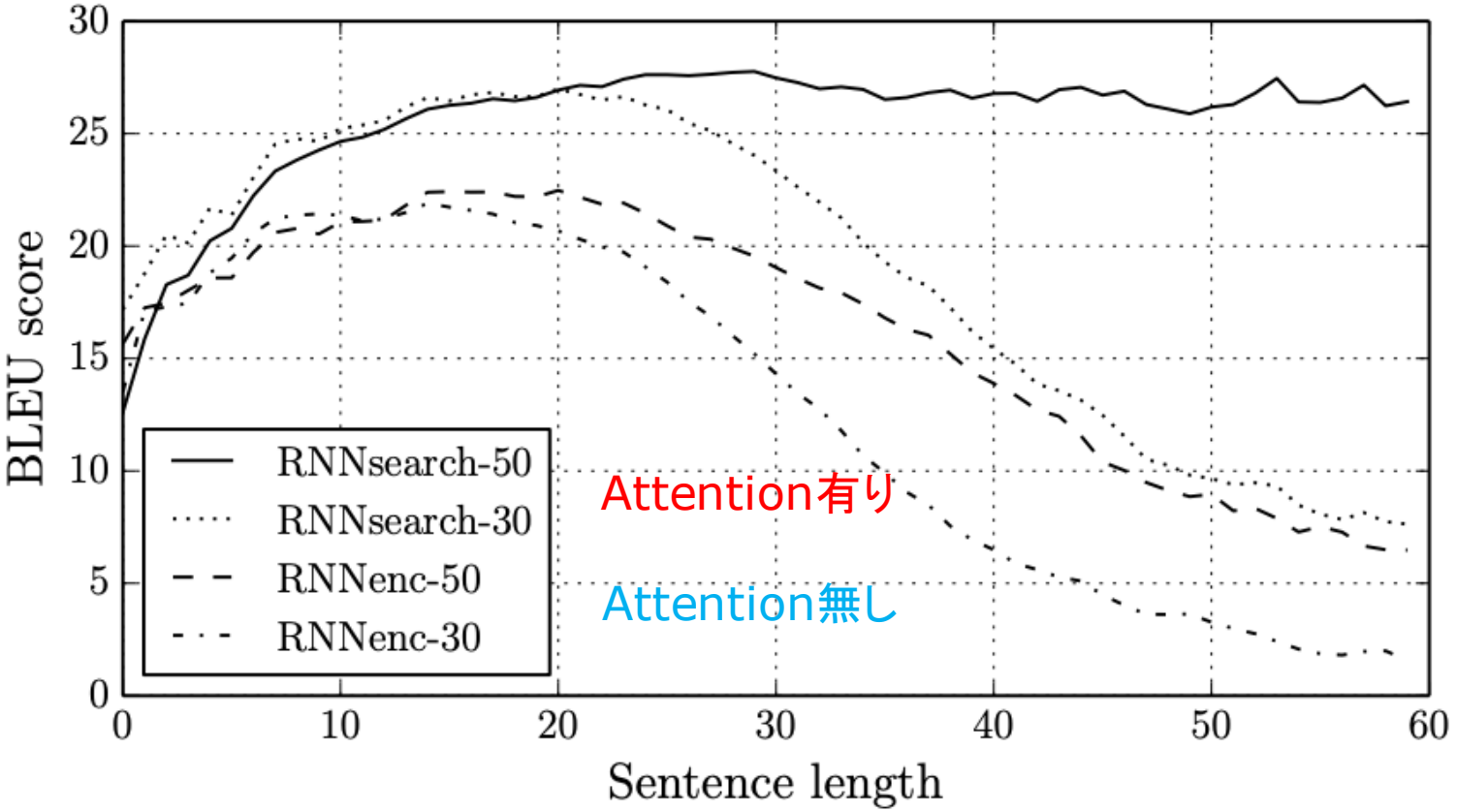
その際、これらの部分を明示的にハードセグメントとして形成する必要はない。

固定長ベクトルがボトルネック

先に見た、Ilya Sutskever らの翻訳システムでは、翻訳さるべき文は、Encoderで、一旦、ある決まった大きさの次元（例えば8000次元）を持つベクトルに変換される。このベクトルからDecoderが翻訳文を生成する。入力された文が、長いものであっても短いものであっても、中間で生成され以降の翻訳プロセスすべての出発点となるこのベクトルの大きさは同じままだ。このシステムでは、長くても短くても入力された文全体が、一つの固定長のベクトルに変換されるのだ。

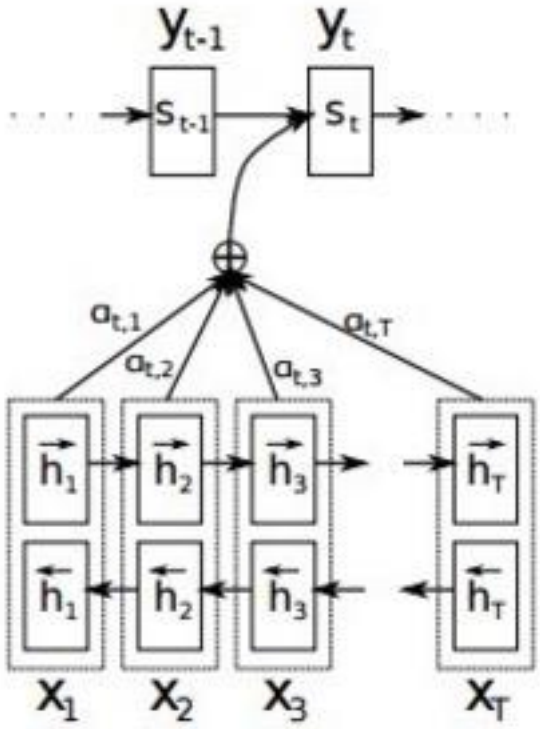
確かに、そこは翻訳の精度を上げる上でのボトルネックになりうる。事実、Ilya Sutskever らのシステムでは、文の長さが長くなるにつれて、翻訳の精度が低下されるのが観察されるという。

文の長さと翻訳の精度


BLEU score
Sentence length
RNNsearch-50
RNNsearch-30
RNNenc-50
RNNenc-30
Attention有り
Attention無し
0
5
10
15
20
25
30
0
10
20
30
40
50
60

この論文の基本的アイデア

文全体に一つの固定長のベクトルを割り当てるのではなく、翻訳時に、ソース・文の一部分を 改めて見直して、その部分から提供される情報を翻訳に生かそうということだ。



$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

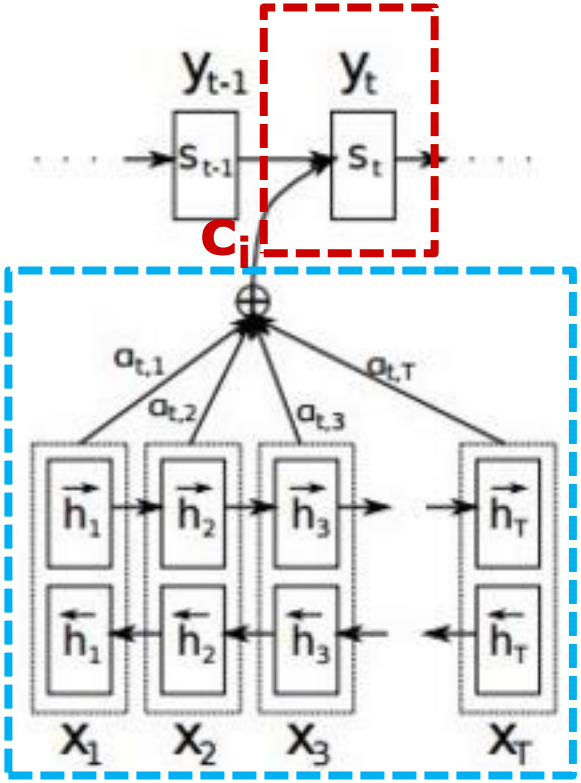
「ここで、yはデコーダによって生成された翻訳された単語であり、xは原文の単語である。上記の図は双方向のリカレント・ネットワークを使用しているが、それは重要ではない。逆方向は無視していい。

重要な部分は、各デコーダの出力するワード y_tが、Encoderの最後の状態だけでなく、すべての入力状態の重みづけられた結合に依存することである。

aは、出力ごとに、それぞれの入力状態をどの程度考慮されるべきかを定義する重みである。したがって、$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

aは、通常、1に合計されるように正規化される（それらは、入力状態に対する確率分布である）。」

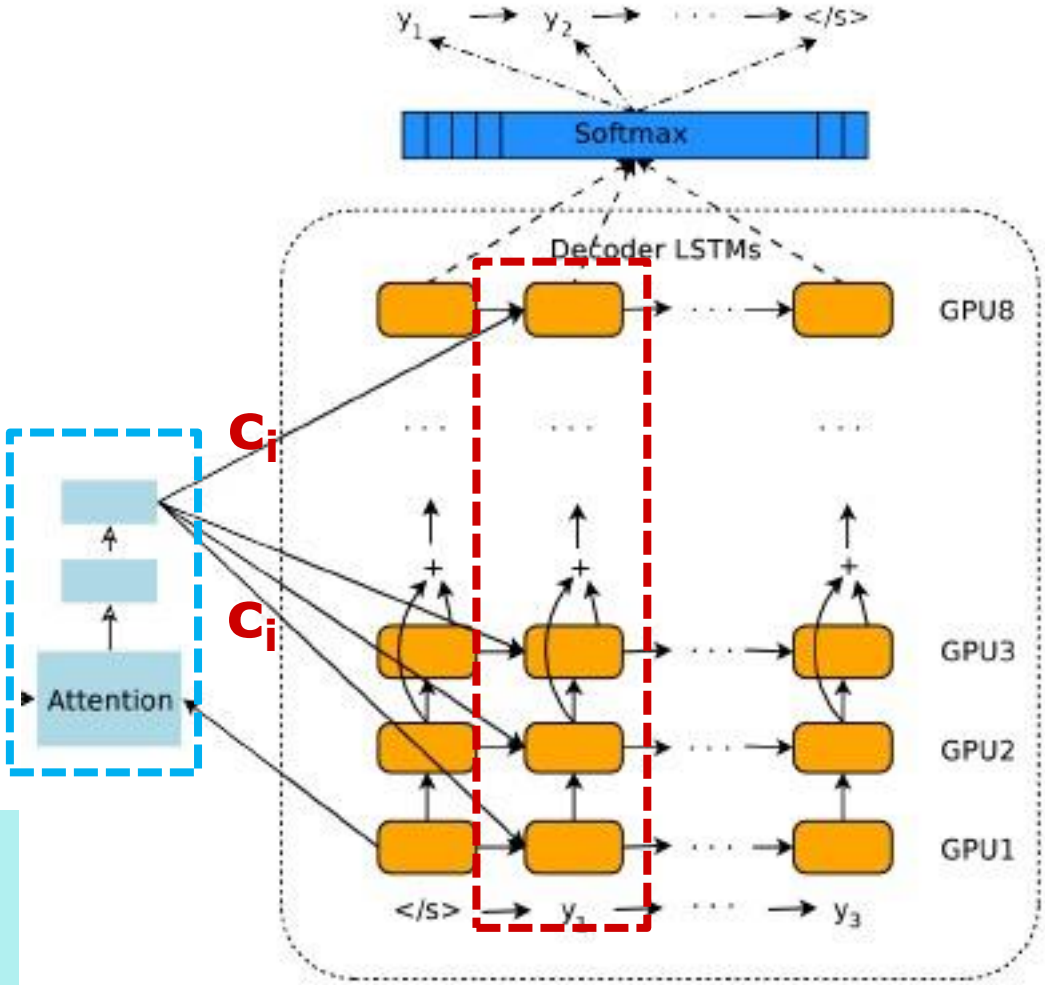
Bahdanau et al.



Decoderの内部状態 s_iは、
先行するノードの内部状態s_{i-1}と
先行するノードの出力y_{i-1}と、
Context $\mathbf{c_i}$で決まる。

$$s_i = f(s_{i-1}, y_{i-1}, \mathbf{c_i})$$

GNMT



このアイデアは、Googleの
ニューラル機械翻訳で生か
されている

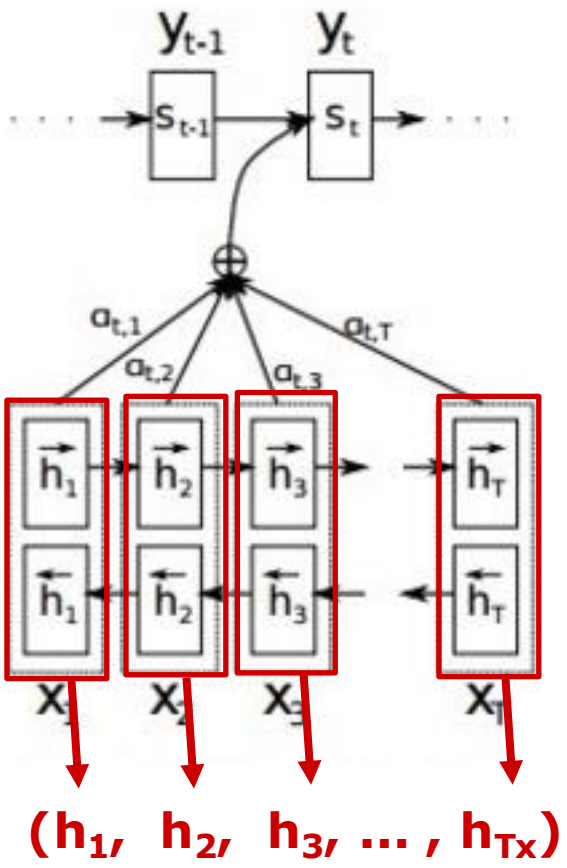
Annotation

このシステムでは、Encoderが、入力のシーケンスをAnnotationのシーケンス
$(h_1, h_2, \ldots, h_{Tx})$ に変える。

先の図には、Annotation h_iの名前は、直接には書き込まれていなかった。

上下に並んだ 逆向きの矢印を持つh_i を囲む四角が書かれているのが、この四角がAnnotation h_i である。ここでは、それを赤い四角で囲んだ。

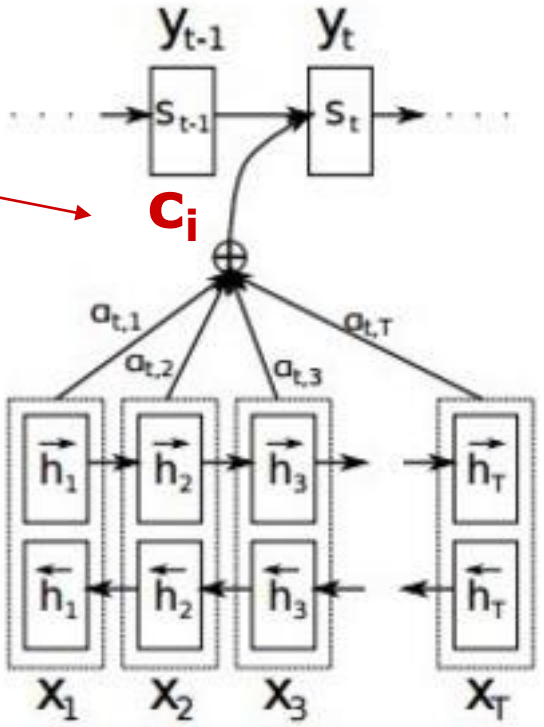
この論文では、右向きの隠れ層の状態と左向きの隠れ層の状態の「連結」としてh_iが実装されている。GNMTでも同様である。



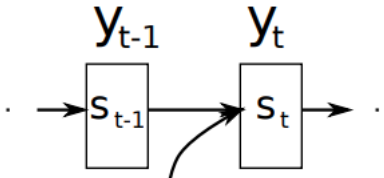
Context

Docoderが、入力の(x_1, x_2, … , x_{Tx}) から、t番目の語y_t を生成しようとする時に、このAnnotationのシーケンス
(h_1, h_2, … , h_{Tx}) からの情報は、次の式で、c_i に束ねられてDecoderに流れ込んでくる。

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$



このc$_i$ を、**Context**と呼んでいる。



$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

Decoderの内部状態

Decoderの内部状態 s_iは、先行するノードの内部状態s_{i-1}と先行するノードの出力y_{i-1}と、このContext c_iで決まる。

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

「直観的には、これは、DecoderにAttentionのメカニズムを実装する。Decoderは、ソースの文のいくつかの部分を、注意を払うべき文章だと決定する。Decoderに、Attentionのメカニズムを持たせることで、Encoderは、ソース文内のすべての情報を固定長のベクトルにエンコードする負担から解放される。この新しいアプローチにより、情報は、Annotationのシークエンス上に広がって拡散することができ、その情報は、Decoderによって選択的に取り出すことができる。」

Context c_iを定義している、それぞれのAnnotation h_jにかかる重みα_{ij}は、次の式で定義される。

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

先の図の$\alpha_{t,1}$, $\alpha_{t,2}$, ... , $\alpha_{t,T}$ を全部足し合わせると、1になる。SoftMaxと同じだ。

α_{ij}の計算の元になる、e_{ij}は、次のように計算される。

$$e_{ij} = a(s_{i-1}, h_j)$$

この指標e_{ij}（**Alijgnment Model**と呼ばれる）は、入力のi番目付近と、出力のj番目付近が、うまくマッチしているかを示すものだ。e_{ij}は、Decoderのi-1番目の状態 s_{i-1}（y_iを出力する直前の状態である）と、Encoderのj番目のAnnotation h_jで決まる。

Aligment Model

次の図は、次の英語とフランス語の翻訳が、下線部分で、語順が逆になることを反映している。

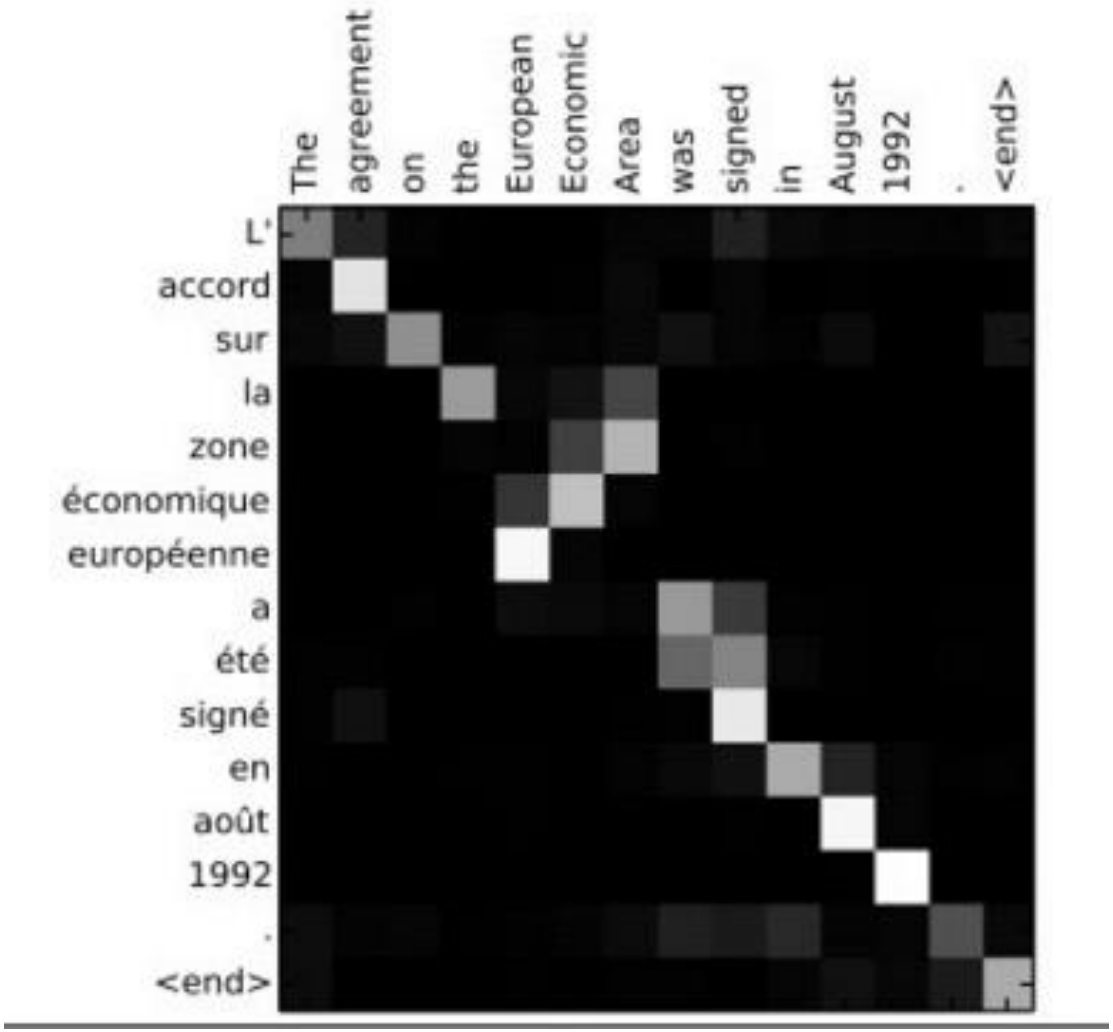
二つの言語の語順が、同じであれば、対角線上に、1が集まることになるのだが。

The agreement on the European Economic Area was signed in August 1992 .

L' accord sur la zone économique européenne a été signé en août 1992 .

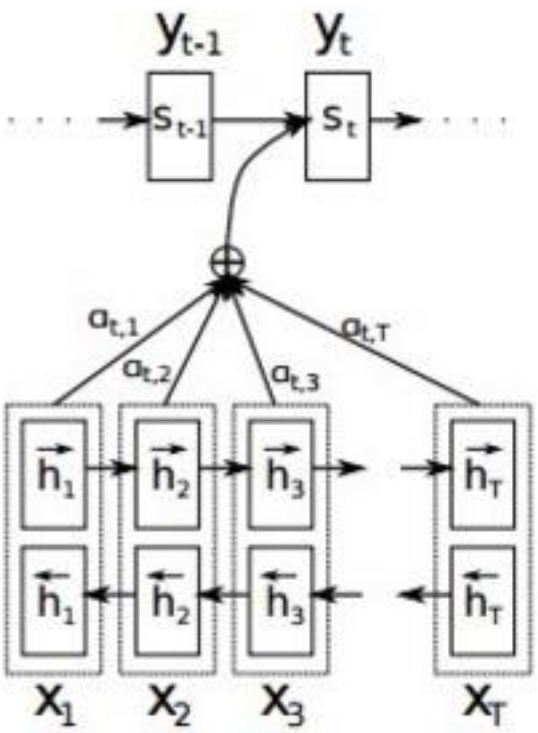
もっとも、Word Alignmentについては、90年代の「統計的機械翻訳モデル」においても、熱心に研究されていたので、こうしたアプローチは、新しいものではない。

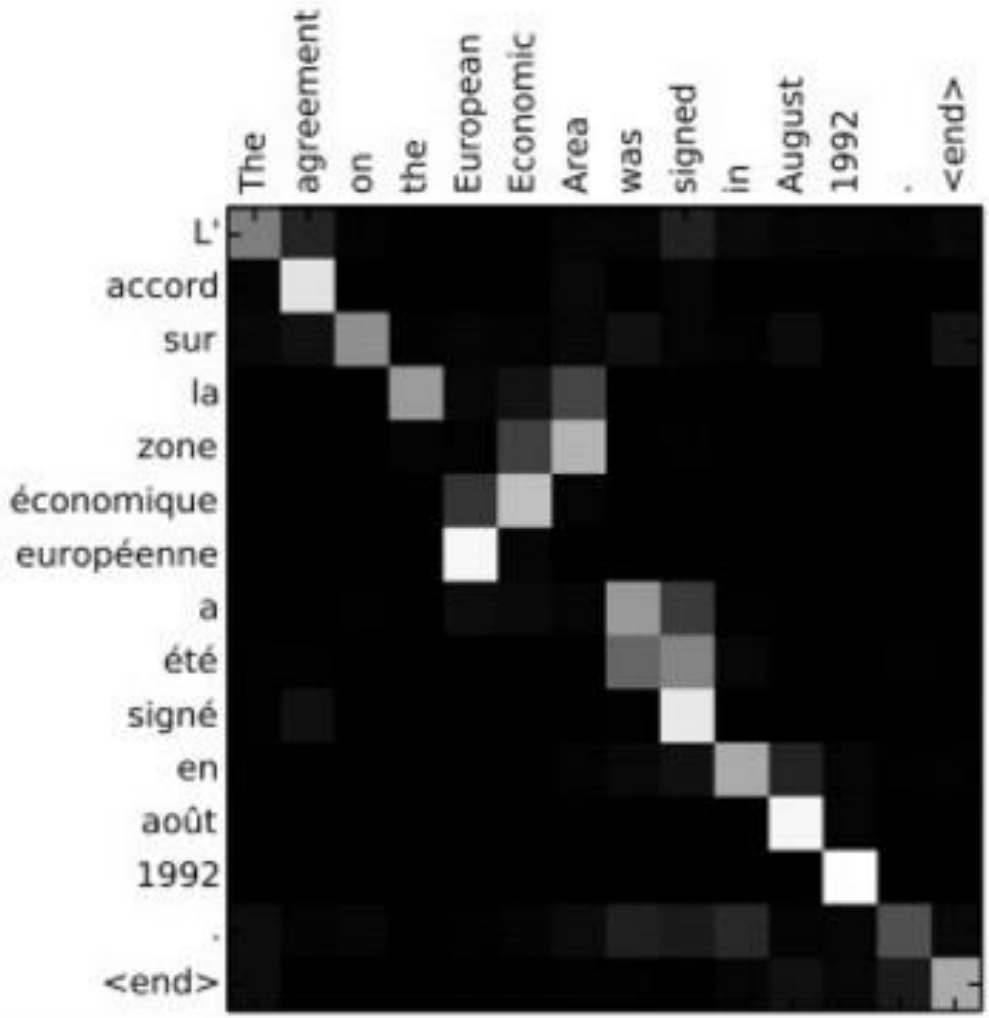
次の図は、英語からフランス語の翻訳で、英語の入力のj番目の語と、フランス語の出力のi番目の語について、α_{ij}の値をグレースケールで示したものである（白は1、黒は0）。



次の図は、英語からフランス語の翻訳で、英語の入力のj番目の語と、フランス語の出力のi番目の語について、α_{ij}の値をグレースケールで示したものである（白は1、黒は0）。

Annotationによる Word Alignment





Googleニューラル機械翻訳

Googleニューラル機械翻訳 GNMTの登場は、エポック・メイキングなものである。同時に、それは、ディープラーニングの自然言語処理技術の発展の自然な継承でもあり、その集大成でもある。

Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu et al.

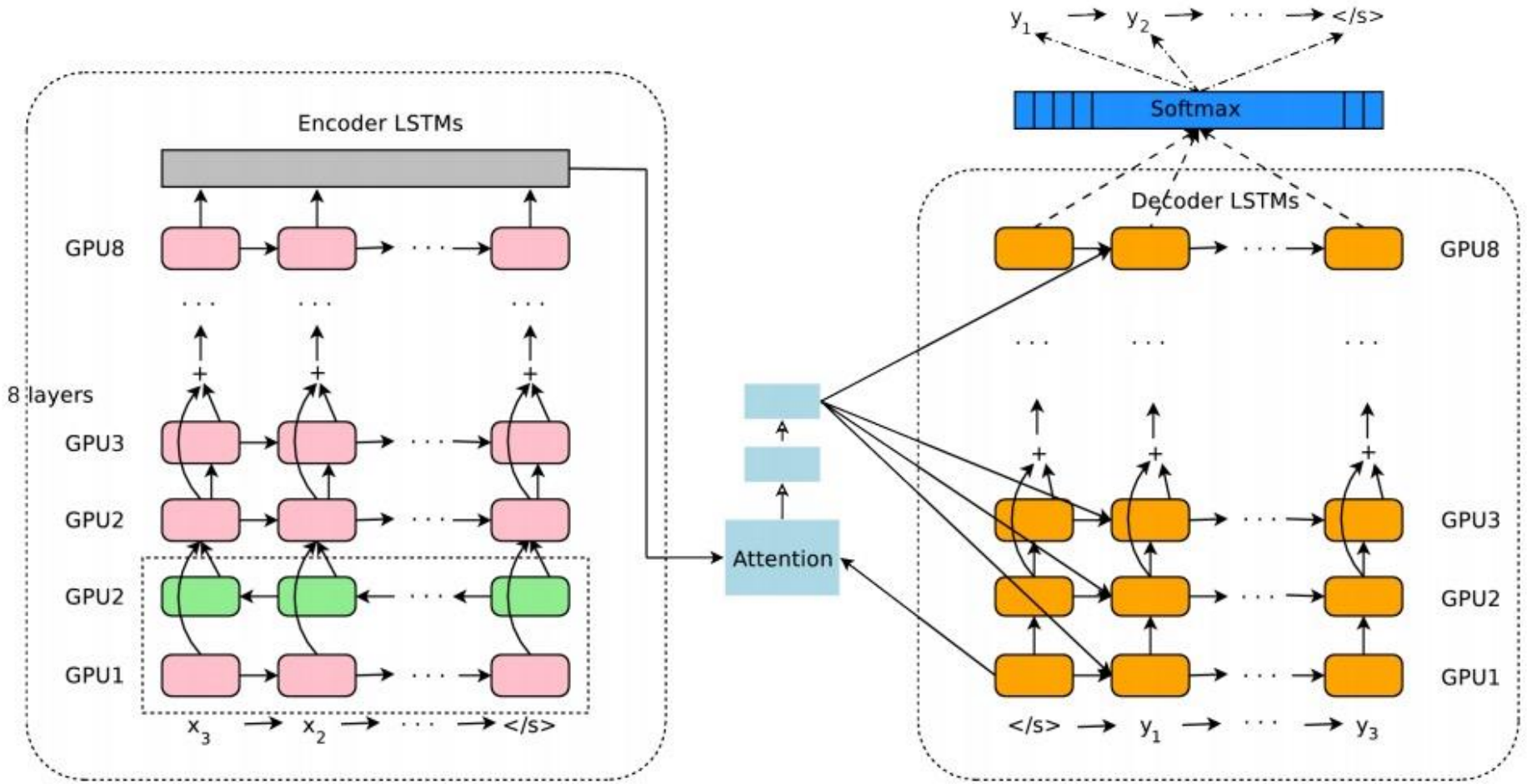
https://arxiv.org/pdf/1609.08144.pdf

2016年

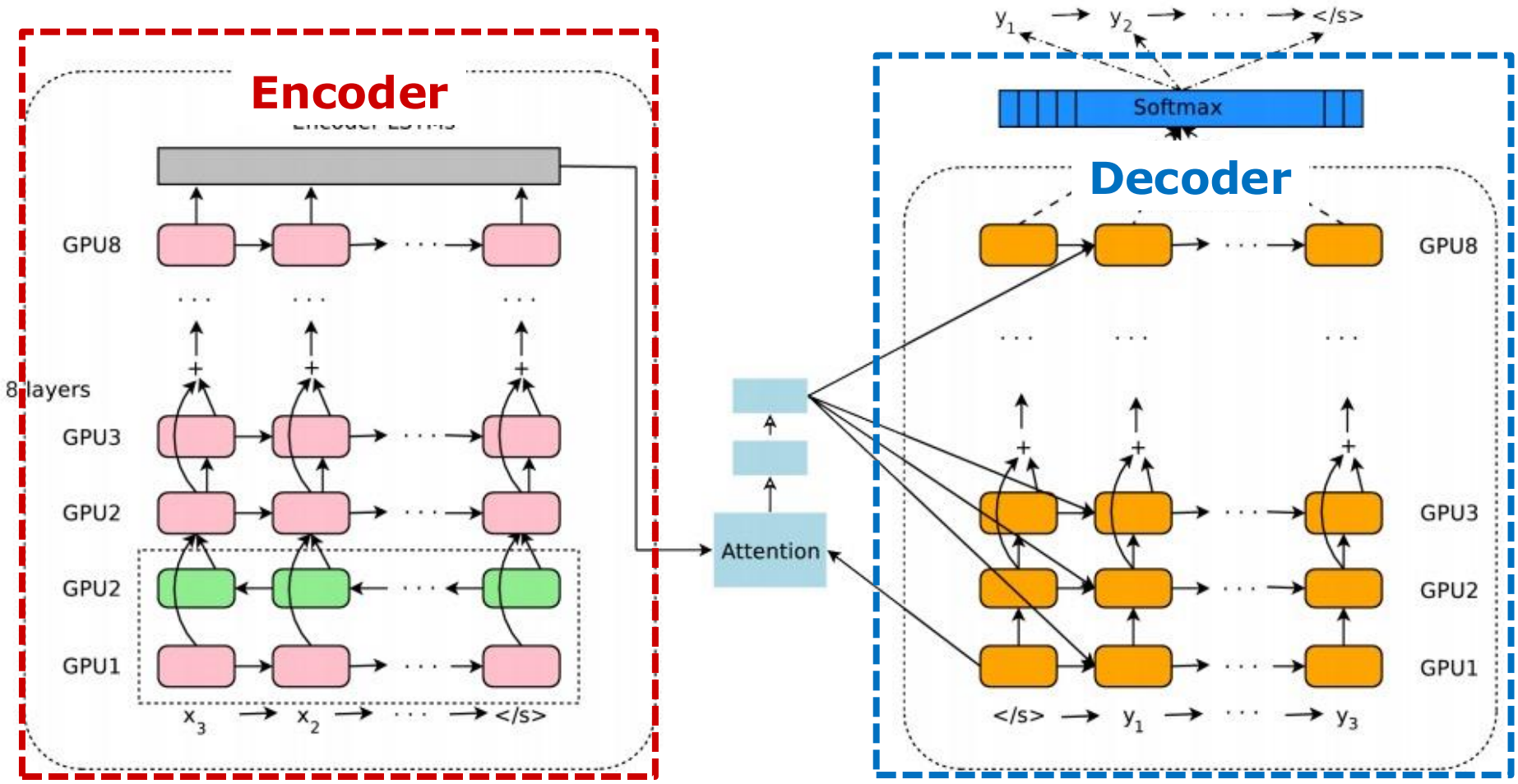
Googleニューラル機械翻訳システムの概観

最初に、GNMTのシステムを概観する。

GNMTのアーキテクチャーの概念図

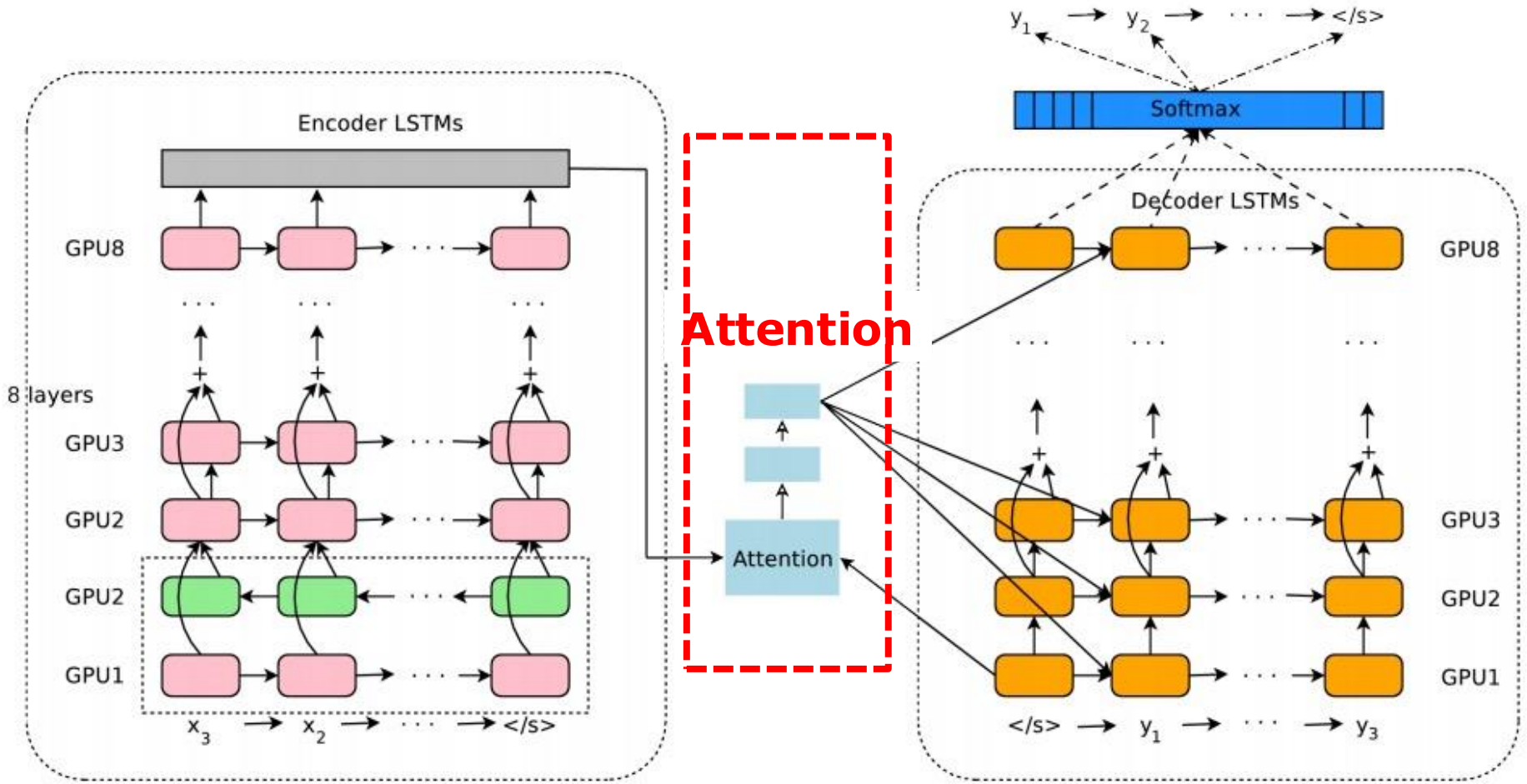


Encoder / Decoder



左側に、LSTMを8段重ねにした Encoder LSTMがあり、
右側には、同じくLSTMを8段重ねにした Decoder LSTMがある。

Attention Mechanism

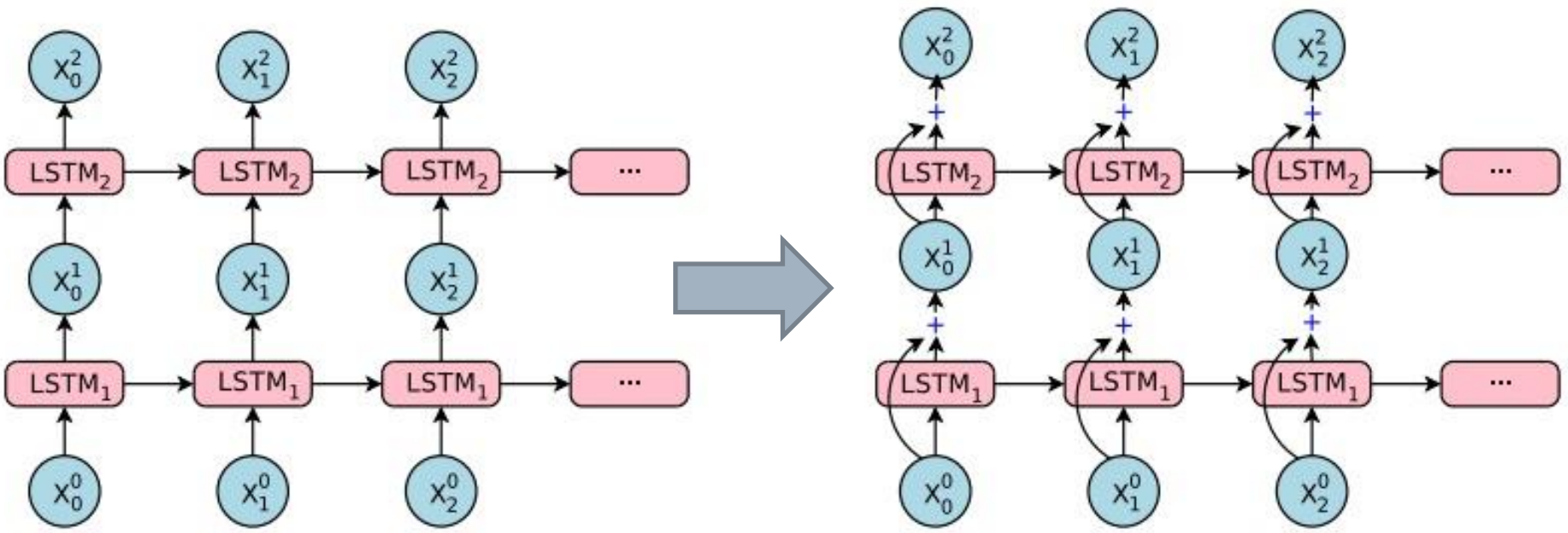


EncoderとDecoderの中間に、Attentionと記された領域がある。
ここからの出力Attention Context は、Decoderのすべての
ノードに供給されている。

Residue Connection

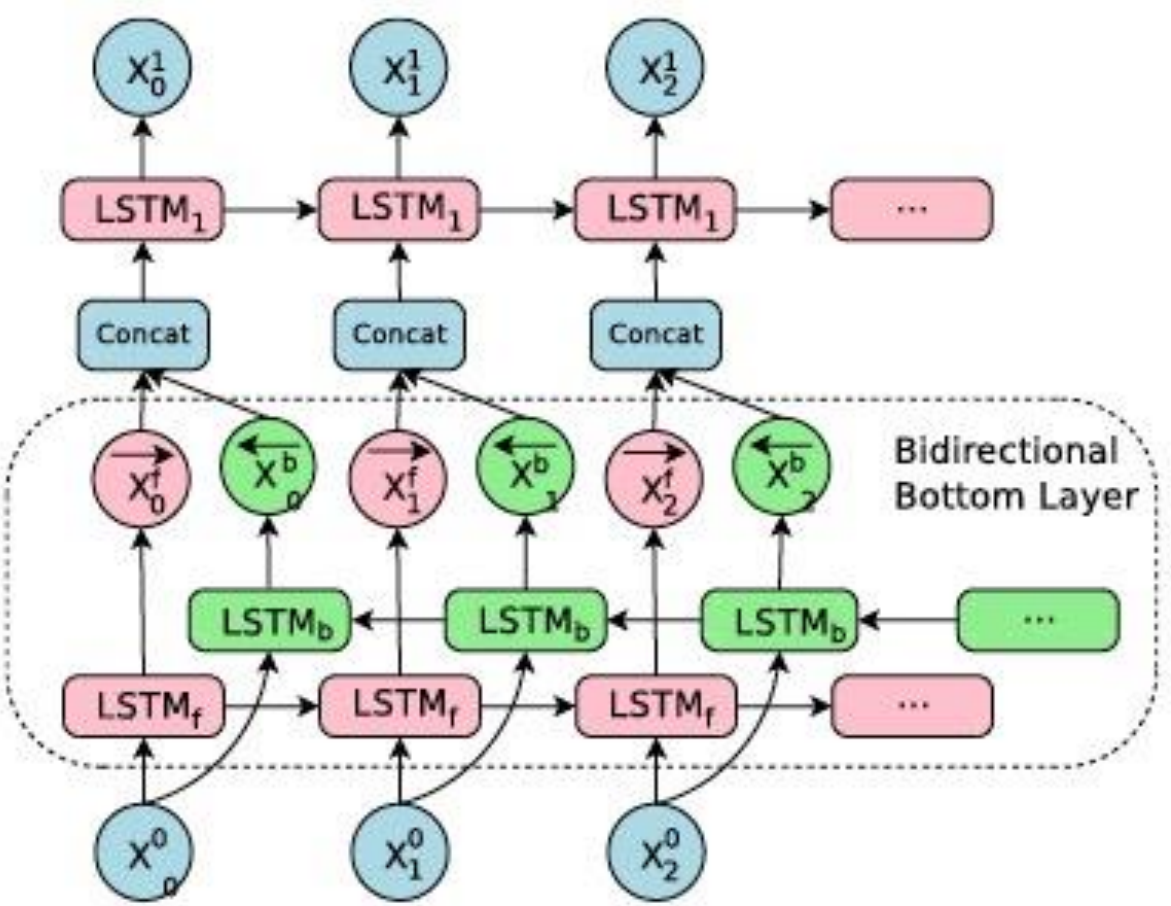
通常のStacked LSTM

GNMT



細かく見ると、LSTMの段の積み重ねに、特徴があるのがわかる。
積み重ねられたLSTMは、一つ下のLSTMからの出力を受け取るだけではなく、もう一つ下のLSTMからの出力をも受け取っている。
これを Residue Connectionと呼ぶ。

Bi-directional Encoder for First Layer



Encoder側の一番下の方の二つのLSTMの処理が、
逆向きに走っている。

Quantizable Model and Quantized Inference

LSTM

with Residual connection

$$\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{W}_6, \mathbf{W}_7, \mathbf{W}_8]$$

$$\mathbf{i}_t = \text{sigmoid}(\mathbf{W}_1\mathbf{x}_t + \mathbf{W}_2\mathbf{m}_t)$$

$$\mathbf{i'}_t = \tanh(\mathbf{W}_3\mathbf{x}_t + \mathbf{W}_4\mathbf{m}_t)$$

$$\mathbf{f}_t = \text{sigmoid}(\mathbf{W}_5\mathbf{x}_t + \mathbf{W}_6\mathbf{m}_t)$$

$$\mathbf{o}_t = \text{sigmoid}(\mathbf{W}_7\mathbf{x}_t + \mathbf{W}_8\mathbf{m}_t)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i'}_t \odot \mathbf{i}_t$$

$$\mathbf{m}_t = \mathbf{c}_t \odot \mathbf{o}_t$$

Input Gate

Forget Gate

Output Gate

$$\mathbf{s}_i = \max(\text{abs}(\mathbf{W}[i,:]))$$

$$\mathbf{WQ}[i,j] = \text{round}(\mathbf{W}[i,j]/\mathbf{s}_i \times 127.0)$$

All accumulator values (c^i_t and x^i_t) are represented using 16-bit integers
All matrix multiplications are done using 8-bit integer multiplication
All other operations (Activation, elementwise operation) 16bit

Google多言語ニューラル機械翻訳

GNMTの多言語拡張についての論文である。

Google 多言語ニューラル機械翻訳

多言語の翻訳を、言語ペアの数だけのシステムによって行うのではなく、一つのシステムで行うことにより、さらに多くの言語へのスケール・アップが容易になる。システムが単純になり、コーパスの少ない言語にもメリットが生まれる。

ゼロ・ショット翻訳が可能であることを示し、また、インターリンガの存在を示唆するなど、非常に刺激的である。

言語に対するいくつかの「仮説」と、ディープラーニングでの結果をある種の「実験」として、結びつけようとするスタイルは新しいものである。

Google’s Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation

Melvin Johnson et al.

https://arxiv.org/pdf/1611.04558.pdf

2016年

論文の概要

「我々は、単一のニューラル機械翻訳（NMT）モデルを使用して、複数の言語どうしを翻訳する、シンプルで洗練されたソリューションを提案する。」

「共有ワードピースのボキャブラリを使用することで、多言語NMTはパラメータを増やさずに、単一のモデルを利用することができる。これは、従来の多言語NMTの提案よりも大幅に簡単なものだ。」

「我々のモデルは、訓練中に明示的には見られなかった言語ペアの間の暗黙的な橋渡しを実行することも学ぶことができた。それは、翻訳の学習とゼロ・ショット翻訳がニューラル翻訳で可能であることを示している。」

「我々のモデルには、普遍的なインターリンガ表現が存在することを示唆する分析を示し、複数の言語を混在させた時に起きる、興味深い例を示す。」

多言語翻訳を単一モデルで

ニューラル機械翻訳（NMT）は、多くの大規模な環境で急速に採用されてきた機械翻訳に対するアプローチである。ただ、このようなシステムのほとんどは、単一の言語のペアのために構築されていた。

基本的なNMTアーキテクチャを大幅に変更することなく、単一のモデルを使用して複数言語のペアを処理するための十分に単純で効率的な方法は、それまでなかった。

この論文では、単一のモデルを使用して多言語間で翻訳を行うための簡単な方法を紹介している。この方法では、ターゲット言語を示す人工的なトークンを、入力シーケンスに追加するだけで、従来のNMTモデルアーキテクチャに、変更を加える必要はない。

実験結果

まず、多言語モデルは、単一言語ペアのモデルより、ハンディがあることを確認しよう。にもかかわらず、多対1のモデルは、単一モデルより翻訳の精度が向上する。

12種類の言語からなる多言語モデルは、278Mのパラメータを持つのだが、合計3.33Bのパラメータを持つ12のモデルを実行するのと、ほぼ同じ性能が達成できる。

多対一の実験結果

Model	Single	Multi	Diff
WMT German→English (oversampling)	30.43	30.59	+0.16
WMT French→English (oversampling)	35.50	35.73	+0.23
WMT German→English (no oversampling)	30.43	30.54	+0.11
WMT French→English (no oversampling)	35.50	36.77	+0.27
Prod Japanese→English	23.41	23.87	+0.46
Prod Korean→English	25.42	25.47	+0.05
Prod Spanish→English	38.00	38.73	+0.73
Prod Portuguese→English	44.40	45.19	+0.79

日本語、韓国語からの英訳は、他の言語の場合に比べて、
かなり、低い。

ゼロ・ショット翻訳

このアプローチの興味深い利点は、明示的な訓練データがない言語ペアの間でゼロ・ショット翻訳を実行できることである。

これを実証するために、2つの異なる言語ペア、
ポルトガル語 -> 英語と英語 -> スペイン語（モデル1）と、
英語 <-> ポルトガル語、英語 <-> スペイン語（モデル2）の4つの異なる言語ペアのデータで訓練されたモデルの、2つの多言語モデルを使う。

これらのモデルの両方が、スペイン語 -> ポルトガル語で、合理的に良質な ポルトガル語を生成できることを示す。

著者らは、これを、「私たちの知る限り、これは真の多言語ゼロ・ショット翻訳の最初のデモです。」と主張している。

意味の共通表現の存在
インターリンガの存在

モデルを複数の言語にまたがってトレーニングすることで、個々の言語レベルでのパフォーマンスが向上し、ゼロ・ショット翻訳が有効になることがわかるということが、この論文の結論なのだが、その意味は？

言語にかかわらず、ネットワークは、同じ意味を持つ文章が同じような方法で表現される何らかの共有表現を学習しているのか？
この質問に対しては、そうだという。これも、画期的。

モデルは、訓練された言語ペアを扱うのと同じ方法で、ゼロ・ショット翻訳を実行しているのか？　これについては、まだよくわからないという。

ビジュアル化による分析

研究の出発点は、アテンション・ベクトルのセット、すなわち、エンコーダとデコーダのネットワークを接続するレイヤ内の活性化の状況を見ることだ。

単一の文の翻訳は、一連のアテンション・ベクトルを生じさせる。この文脈において、共有表現に関する元々の問題は、異なる文のベクトルのシーケンスがどのように関連しているかを調べることで研究することができる。例えば、次のように。

ソースまたはターゲット言語に応じて、文章がクラスタリングされているのか？

あるいは、言語にかかわらず、同様の意味を持つ文章がクラスタリングされるのか？

インターリンガの証拠

いくつかの訓練されたネットワークは、実際に共有表現の強いビジュアルな証拠を示す。

たとえば、以下の図2は、英語 <-> JapaneseとEnglish <-> Koreanでトレーニングされた多対多モデルから作成されたものである。モデルの実際の動作を視覚化するために、意味論的に同一の言語間フレーズの74個のトリプルからなる小さなコーパスから始めた。

つまり、それぞれのトリプルには、同じ基本的な意味を持つ英語、日本語、韓国語のフレーズが含まれている。これらのトリプルをコンパイルするために、私たちは日本語と韓国語の翻訳と対になった、英語の文章のための正しい（Ground Truth）データベースを検索した。

英語

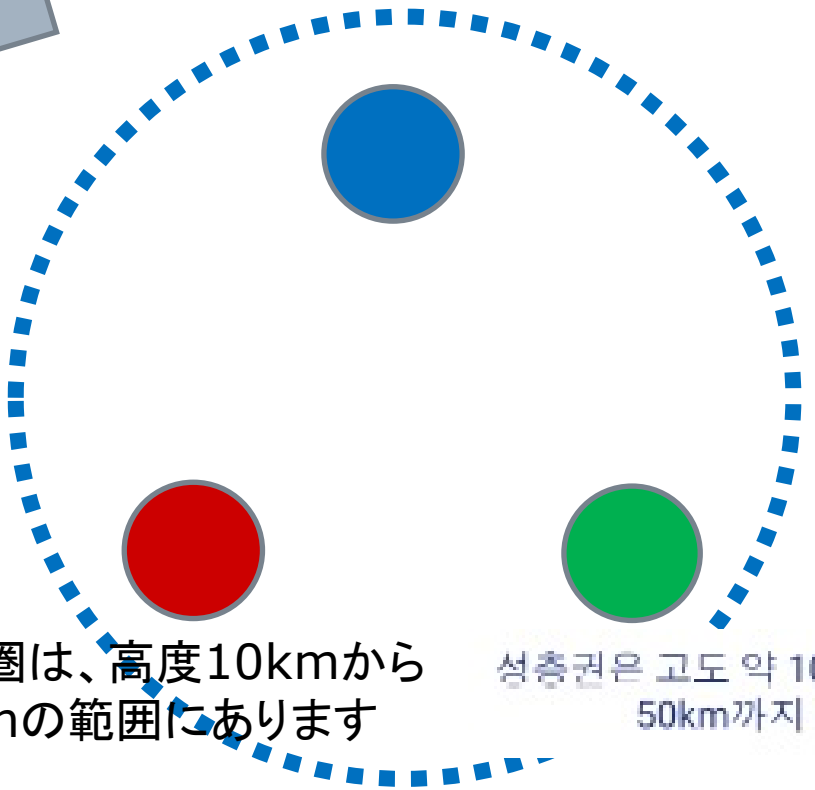
三つの言語で
同じ意味を持つ文を
一つのトリプルに
まとめる。

日本語

韓国語

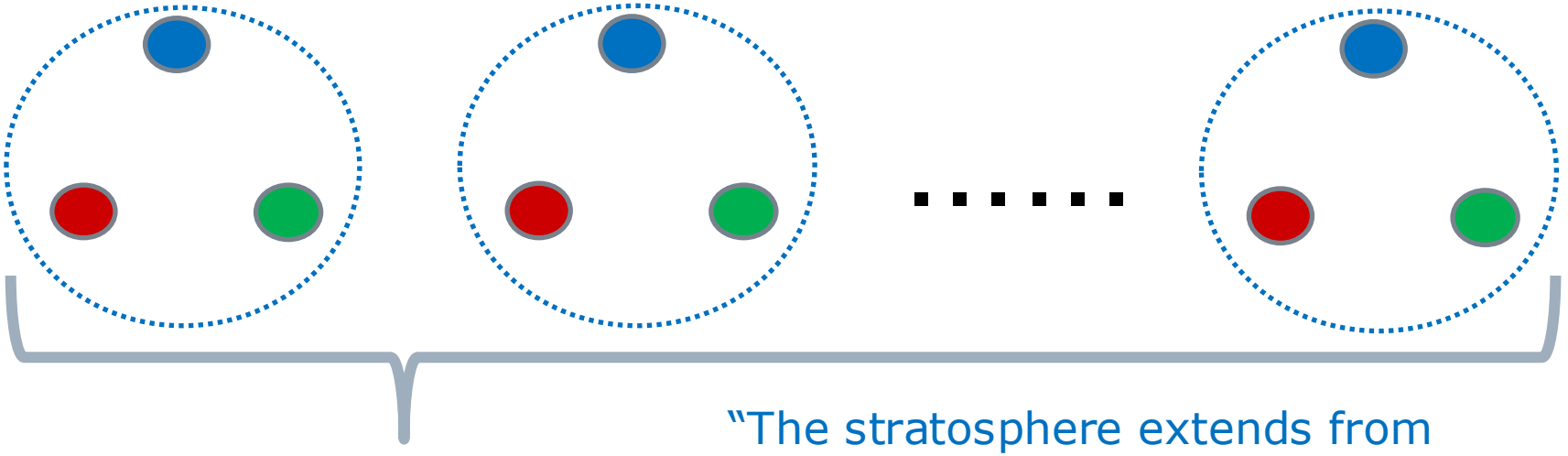
例えば

“The stratosphere extends from about 10km to about 50km in altitude.”



成層圏は、高度10kmから
50kmの範囲にあります

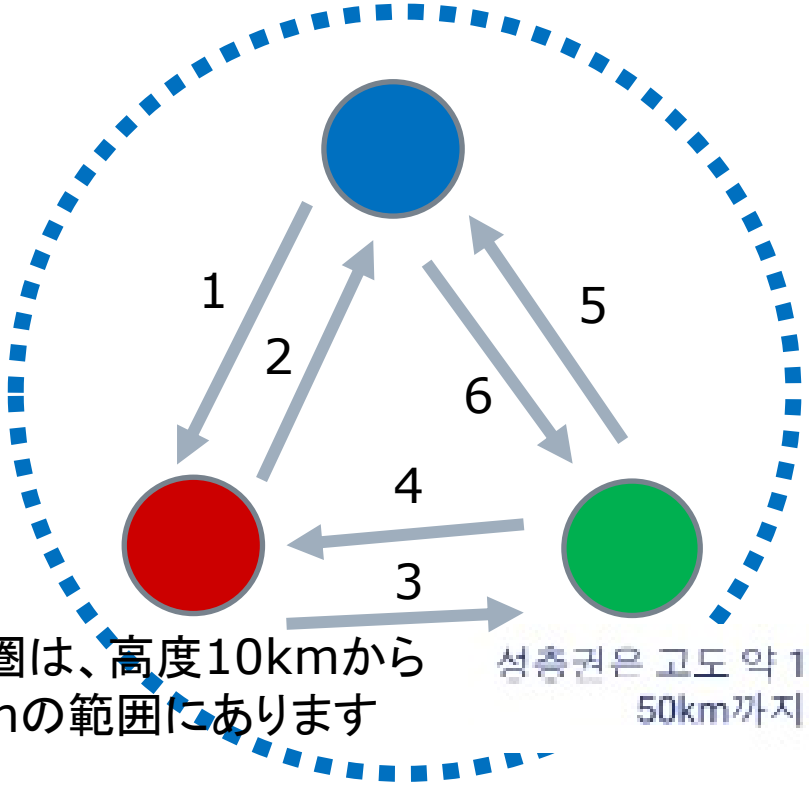
성층권은 고도 약 10km부터 약
50km까지 확장됩니다.



74個のトリプルからなる
コーパスを準備する。

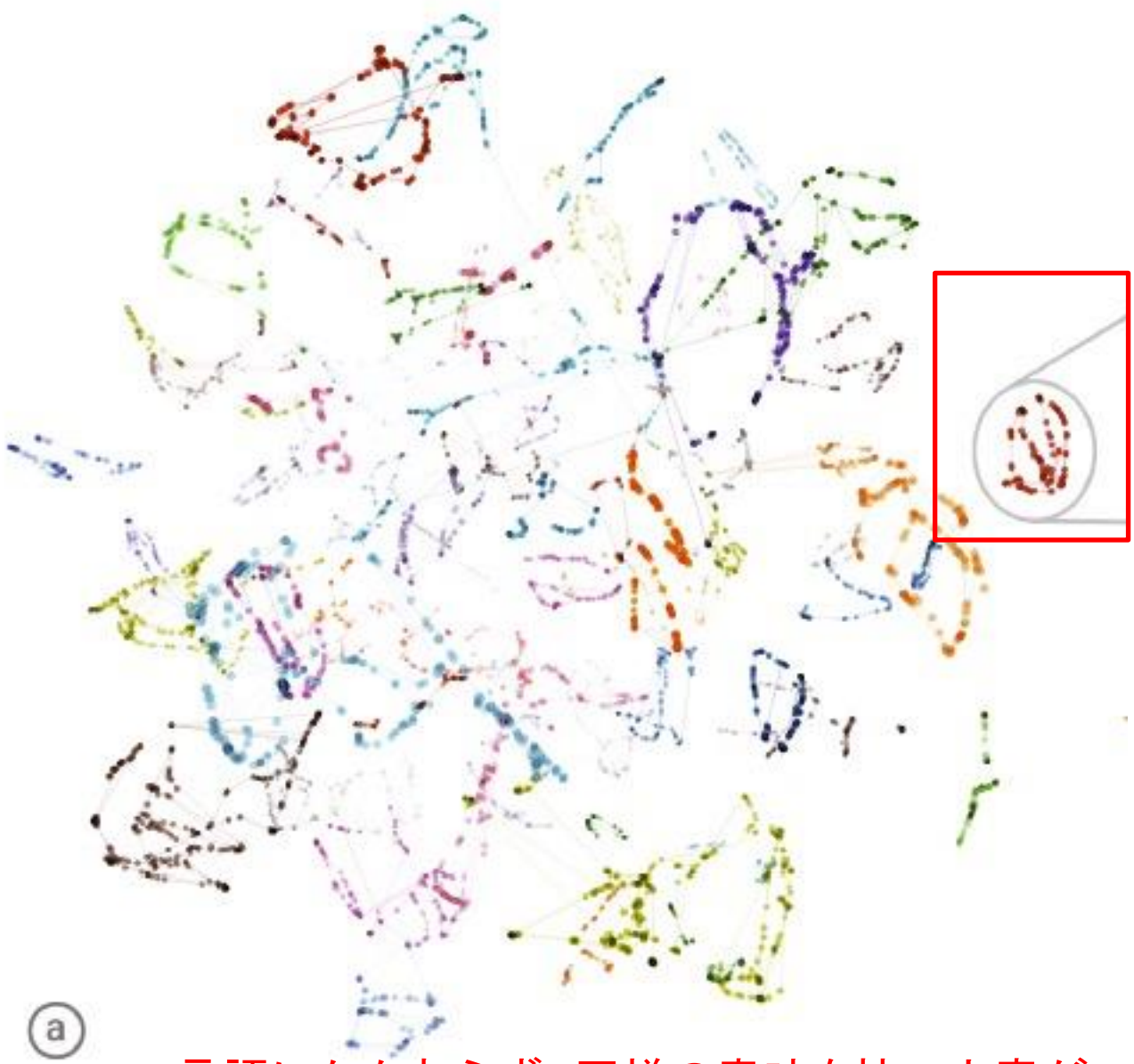
"The stratosphere extends from
about 10km to about 50km in altitude."

各トリプルの各センテンスを
他の2つの言語に翻訳する。
一つのトリプルについて、
6つの翻訳が生まれる。



このコーパスでは、
74 x 6 = 444 の
翻訳が生まれる。

成層圏は、高度10kmから
50kmの範囲にあります



言語にかかわらず、同様の意味を持つ文章がクラスタリングされている。

この 444の翻訳を行うのに、システムは9,978ステップ要した。

この時、発せられた9,978個のアテンションベクトルを、t-SNEを用いて二次元に投射したのが左の図。

同じトリプル内の文の翻訳で発せられるアテンション・ベクトルは、同じ色で表している。

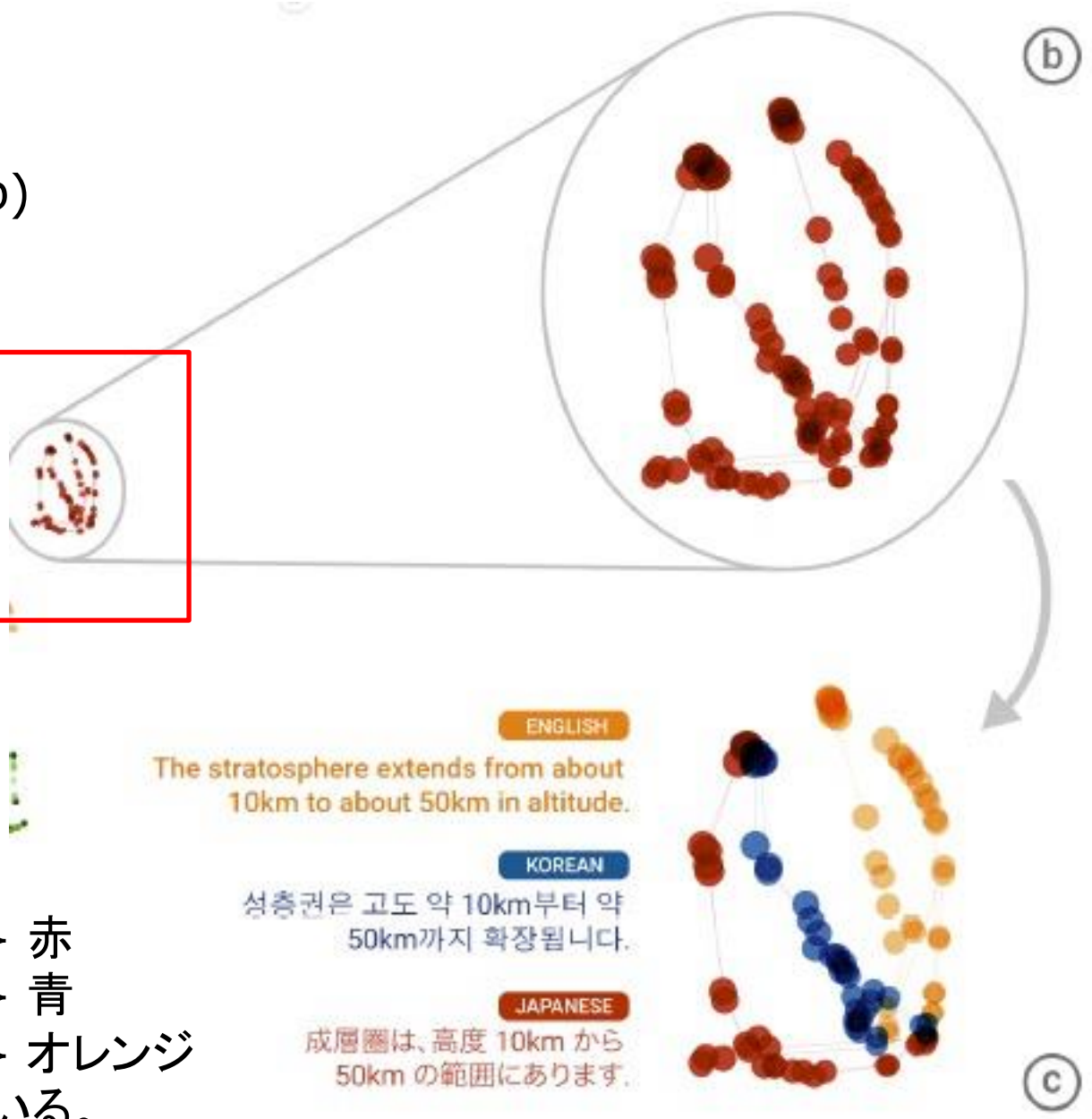
同じ色が、近くに集まってクラスターを形成しているのが見て取れる。（？）

この同じ色のクラスター(b)は、元の同じ意味を持つトリプルから生まれたものである。

同一の文から発せられるアテンション・ベクトルは、線で結ばれている。

(c)は、翻訳ソースが同じ言語を同じ色で塗り分けたもの。

日本語->{英語,韓国語} 赤
韓国語->{英語,日本語} 青
英語->{日本語,韓国語} オレンジ
は、クラスターを形成している。



意味の「同一性」 / 意味の共通表現の存在

- 言語が異なっていても、「同じ」意味を持つ語が存在する。
- 言語が異なっていても、「同じ」意味を持つ文が存在する。

機械翻訳技術から派生した大規模言語モデルの成功は、この「意味の共通表現」が存在し、それを機械が発見する能力を持つことによるものだと、考えることができる。